RESEARCH ARTICLE OPEN ACCESS

A New Bitmap Indexing Method for Complex Similarity Queries

*Guang-Ho Cha
(Department of Computer Engineering, Seoul National University of Science and Technology, Republic of Korea,
Correspondin Author : Guang-Ho Cha

ABSTRACT
A new indexing method for complex similarity queries are proposed in this paper. The efficiency of the new indexing method is realized by a specialized bitmap index that represents all objects in a database as a set of bitmaps. In order to provide the index with the flexibility in dealing with multiple features, we treat every dimension independently. The percentage of data accessed in the index is inversely proportional to the overall dimensionality of data, and thus the performance deterioration with the increasing dimensionality does not occur. To demonstrate the efficacy of our method we conducted extensive experiments and compared the performance with the linear scan by using real image datasets, and obtained a remarkable speed-up over the linear scan.
Keywords **-** Bitmap index, complex similarity query, dimensionality curse, multimedia database, image database

---

Date of Submission: 02-01-2018 Date of acceptance: 19-01-2018

I. INTRODUCTION

The similarity or nearest neighbor search problems are reasonably well solved in low dimensions for which efficient indexing methods have been presented. Recent research results show that in high dimensions, the concept of proximity may not be very meaningful. These results show that for certain classes of commonly used similarity functions such as the L_p-norm, the nearest and farthest neighbors are of the same relative distance to the query point for large classes of data distributions. The lack of relative contrast in terms of similarity is undesirable since it is not clear whether or not the nearest neighbor is meaningful.

Hinneburg et al. [1] address this quality issue of the nearest neighbor search and identify relevant features (dimensions) with respect to a given query. These identified features are used in order to determine the most similar objects. However, since it is very difficult to find the best combination of dimensions in the neighborhood of a query point, this approach is slower than a linear scan.

Aggarwal and Yu [2] stated that one of the reasons for the lack of discrimination between the nearest and farthest neighbors is the fact that for every pair of points there are dimensions with varying distances to the corresponding values in the target. When the dimensionality is high, even the most similar objects are likely to have a few feature values which are far separated because of sparseness and noise effects of the data. Therefore, the distance functions such as the Lp-norm that does the straightforward summation of distances over all dimensions may lack the relative contrast of the similarity among near neighbors.

We solve the high-dimensional indexing problem with the objectives of flexibility and efficiency. The flexibility means that the various features to describe the multimedia content can be treated in a flexible way. This property is very important to process the complex similarity queries with multiple different content descriptors. To this end, we develop a meaningful similarity function on which each dimension is treated independently. The efficiency means that the indexing method shows the behavior of improved performance with increasing dimensionality.

For multimedia data, we can imagine that there may be many diverse types of features to describe its content. The discriminating power of the content-based image retrieval (CBIR) system may increase as more information describing the media content is included in the metadata. In this paper, we assume that complex similarity queries are composed of Boolean combinations of feature descriptions. An example would be the query

(color = ‘red’) $\wedge$ (texture = ‘smooth’) $\wedge$ (shape = ‘round’).

In order to process this kind of complex query, we may assume that one subsystem deals with colors, and a completely different subsystem deals with textures, and a third subsystem treats shapes. Then the results from three different subsystems must be combined to produce the final result. Fagin presented an algorithm called A_0 that returns k answers for this kind of complex query $F(A_1, \ldots, A_m)$, where subsystem i evaluates the subquery A_i

and then the subresults are combined to return top k answers [3]. In order to handle this type of Boolean queries, Cha and Chung [4] also synthesize the subresults came independently from different types of subqueries. The main problem of this kind of approach is that the evaluation cost is expensive, for example, the cost of algorithm A_0 for evaluating $F(A_1, \ldots, A_m)$ is $O(N^{(m-1)/m} k^{1/m})$ when N is the number of objects in the database. We solve the complex similarity query problem with a single indexing method.

II. RELATED WORK

A variety of indexing methods for high-dimensional data have been proposed. Most of the work was motivated by the limitation of the state-of-the-art tree-structured indexes and they tried to break the dimensionality curse. Recently, most work trends toward the approximate nearest neighbor search beased on the principle that the approximate answers are enough for many applications. Moreover, in many multimedia applications, the concept of similarity may be subjective and it is also hard to articulate the query concept.

As stated in [5, 6, 7], it is very unlikely that traditional data space partitioning-based multidimensional index trees can prune some search space in high-dimesional data spaces, and thus most multidimensional index trees could not outperform the brute-force linear scan. The vector approximation-based approaches [5, 7] compress the data, scan them linearly, and try to filter the original data so that only a small fraction of them must be read. The drawback of this approach is that its performance cannot be better than scanning linearly the compressed data.

To the best of our knowledge, there have been no indexing methods that deal with complex similarity queries. Most methods treat the data with fixed numer of dimensions, and therefore they cannot adapt to the change of the number of featrures. However, it is very important for the effectiveness and efficiency of the CBIR to provide the functionality of querying with the variable number of features since users may use different vocabulary (i.e., different combinations of features) to pose a query. Our indexing method deals with the data with variable number of features very efficiently. Moreover, in our indexing method, the number of data accesses is inversely proportional to the overall data dimensionality.

III. BITMAP INDEXING

In high-dimensional space, the concept of proximity may not be very meaningful. In other words, most points are far away from one another due to the sparsity of data space. Thus, in order to restrict the range of proximity to a certain extent in terms of distance, we transform the k-nearest neighbor (k-NN) search problem to the range search problem.

3.1. Transformation to Projected Range Search

We transform the k-NN query to a projected range query in order to efficiently apply our indexing method to high-dimensional and complex similarity queries. In addition to the performance issue, the transformation to the projected range search implies limiting the similarity bounds to a certain extent.

In order to perform the range query efficiently, we first discretize the data into a set of intervals. Specifically, we assume that each dimension is divided into K intervals. This classification has an effect of similarity thresholding on each feature dimension. In other words, the two points classified into the same interval are defined to be similar within the interval. We employ the K-means algorithm [8] to classify the data into K clusters. It is a simple and efficient method but other more sophisticated partitioning algorithms may be used to this purpose.

After the data is classified into K intervals (i.e., clusters) on each dimension, for the jth interval $I_{i,j}$ on dimension i, we create a bitmap $b_{i,j}$. The bitmap $b_{i,j}$ contains the binary information that says which objects are lying on the interval $I_{i,j}$. For dimension i and two objects $X = (x_1, x_2, \ldots, x_d)$ and $Y = (y_1, y_2, \ldots, y_d)$, if both x_i and y_i belong to the same interval $I_{i,j}$, then two objects are considered to be similar on dimension i. For dimension i, let S_i be the set that contains the data objects lying on the interval on which the query point lie.

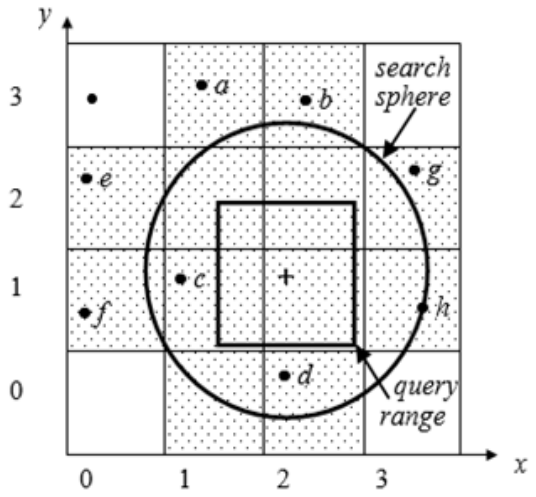

Fig. 1 Projected range query transformed from nearest neighbor query

With an example, we explain how the query range is defined. The range query extent for dimension i is the interval $I_{i,j}$ on which the query point lie. Considering Fig. 1, when the query point is

given by +, the intervals $I_{x,2}$ and $I_{y,1}$ are the range query extents for dimensions x and y, respectively. The candidates for the similar objects to the query point are determined by the intervals on which the query point lies. For dimension x, objects b and d are the candidates, i.e., S_x = {b, d}, and for dimension y, objects f, c, and h are the candidates, i.e., S_y = {f, c, h}, since they share the same intervals with the query point. To get the final candidates, we perform the intersection on Sx and Sy. If the number of objects common both to S_x and S_y is greater than or equal to k, the objects resulting from the intersection are the final candidates for the k-NN query. Otherwise, we extend the size of the query range to either side on dimension i by adjacent intervals of the $I_{i,j}$, and perform the intersection again with the extended query range. For example, in Fig. 1, since the number of objects both in S_x and S_y is less than k (= e.g., 3), the query ranges are extended to the interval covering $I_{x,1}$, $I_{x,2}$ and $I_{x,3}$ for dimension x, and the interval covering $I_{y,0}$, $I_{y,1}$ and $I_{y,2}$ for dimension y. This process is repeated until there are at least k objects.

3.2. Similarity Function

In order to provide the similarity function with flexibility, the similarity function needs to be defined so that it treats each dimension independently. The similarity threshold is predefined by the intervals of the range query transformed from the k-NN query. In other words, the objects within the defined query range are assumed to be within the similar range. For each dimension, we classify data objects into a set of intervals based on the K-means algorithm such that the objects in the same interval are more similar to each other than to objects in different intervals. For dimension i, the jth interval $I_{i,j}$ is identified and its lower and upper bounds l_{ij} and u_{ij} are found. For two objects x = (x_1, x_2, …, x_d) and y = (y_1, y_2, …, y_d), (x_{ij}, y_{ij}) ∈ [l_{ij}, u_{ij}] in dimension i, where the subscript ij denotes the object lies on the interval j in dimension i. In this situation the distance function D_i for two objects x and y in dimension i is defined as follows:

$$D_i(x_i, y_i) = \begin{cases} \dfrac{|x_i - y_i|}{u_{ij} - l_{ij}} & \text{if } x_i \text{ and } y_i \text{ lie on the same interval} \\ 0 & \text{otherwise} \end{cases}$$

The $u_{ij} - l_{ij}$ in the denominator normalizes the distances with respect to the different intervals. For two objects x = (x_1, x_2, …, x_d) and y = (y_1, y_2, …, y_d), the set of dimensions on which the two objects are similar includes those dimensions that have intervals that overlap the query range. Thus, for dimension i, if both x_i and y_i belong to the intervals within the query range, then the two objects are said to be in the similar range on dimension i. The overall distance between two objects x and y is given by:

$$D(x,y) = \left[\sum_{i=1}^{d} w_i D_i^{\,p} \right]^{1/p}$$

where w_i is the weight assigned to dimension i by user. Note that the overall distance value varies between 0 and Σw_i since individual distance value for dimension i lies in between 0 and 1. In our experiments, we set p = 1.

3.3. A New Bitmap Index

Our new bitmap index is based on the classified ranges of each dimension. The intersection and union of two ranges from other dimensions can be processed through the bitwise AND and OR operations, respectively.

3.3.1. Index Creation

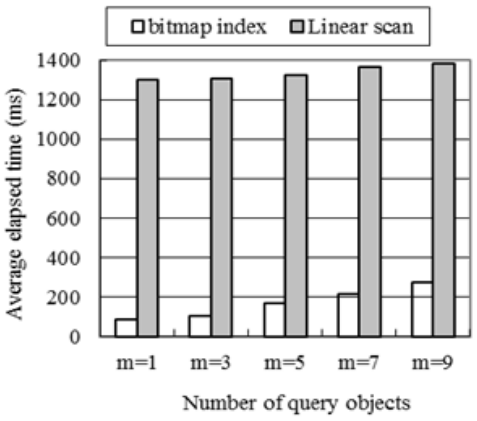

Fig. 2 Disk Accesses Experiments

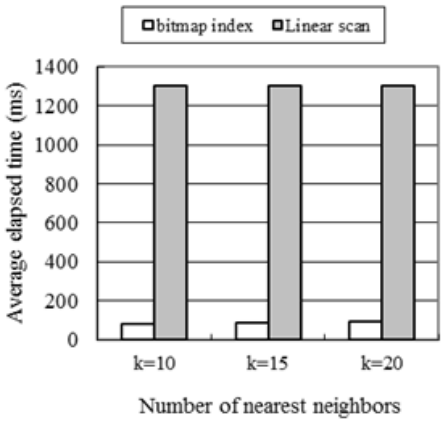

Fig. 3 Time Experiments

We create the bitmap index as follows:

(1) For each dimension i, $1 \le i \le d$, we classify the data into a set of K_i intervals. Let us denote the jth interval for dimension i by $I_{i,j}$.

(2) For the interval $I_{i,j}$, we create a bitmap $b_{i,j}$ and its lower and upper bounds [l_{ij}, u_{ij}]. The pth bit of the

bitmap $b_{i,j}$ has the value 1 if the object numbered p lie on the interval $I_{i,j}$.

(3) For the objects lying on the interval $I_{i,j}$, along with the bitmap $b_{i,j}$, we keep the actual coordinates for the dimension i in those objects.

(4) The bitmap index is simply an array of those bitmaps along with a list of actual coordinates and lower and upper bounds $[l_{ij}, u_{ij}]$'s for each interval $I_{i,j}$.

3.3.2. k-NN Search

When a user poses a k-NN query with feature vector $Q = \{Q_1, Q_2, ..., Q_d\}$, query weight $w = (w_1, w_2, \ldots, w_d)$, and k, the query Q is processed as follows:

(1) For the query Q and each dimension i, get the bitmap b_i for the interval on which Q_i lies, $1 \le i \le d$.

(2) Set all bits of the bitmap b to 1.
Let b_c be the number of 1-bits in b.

(3) for $1 \le i \le d$, do {
b = b AND b_i.
Let I_l be the left adjacent interval of b_i.
Let I_r be the right adjacent interval of b_i.
while ($b_c < k$) do {
b = b OR (bitmap of I_l) OR (bitmap of I_r).
Let I_l be the left adjacent interval of I_l.
Let I_r be the right adjacent interval of I_r.
}
}

(4) Get the final bitmap b.

(5) Obtain the set S of objects associated with the set bits in b.

(6) For the objects in S, compute their distance values D and return the k objects in increasing order of D.

IV. PERFORMANCE EXPERIMENTS

To demonstrate the practical effectiveness of our bitmap index, we performed an extensive experimental evaluation of our bitmap index and compared it with the linear scan. Since most multidimensional indexing methods cannot outperform the linear scan in high dimensions, the linear scan is usually used as the yardstick for performance comparison in high dimensions.

For our experiments, we used 12,861 images from an image dataset and COIL-100 (Columbia Object Image Library). To obtain feature vectors for experiments, we used MPEG-7 visual descriptors: dominant color, homogeneous texture, edge component histogram, and color structure. Theses descriptors are general descriptors that can be used in most applications. In all experiments, the number (k) of nearest neighbors to find were 10, 15, and 20. The page size used in the disk access experiments was 4 KB. 100 532-dimensional complex k-NN queries were processed in each experiment and the results were averaged.

4.1. Number of Disk Accesses

We computed the total number of disk accesses for the k-NN complex queries. Fig. 2 shows the total number of disk accesses to find k NNs. The number of disk accesses performed by the bitmap index is far smaller that those of the linear scan. The performance improvement shows that the bitmap index can be successfully employed for complex similarity search in high dimensions.

4.2. Elapsed Time

To demonstrate the practical effectiveness of the bitmap index, we also performed a number of timing experiments. Fig. 3 shows that the bitmap index achieves a remarkable speed-up over the linear scan.

V. CONCLUSION

We proposed a novel bitmap indexing method for complex similarity queries in high-dimensional spaces. The key design goal of our method was two-fold: (1) providing fast search for k NNs in high-dimensional multimedia databases and (2) supporting complex similarity queries. In order to achieve this goal, we used a similarity function where each dimension is treated independently. In this function the similarity is measured by the number and quality of similarity along the dimensions on which the query object and the target object are identified to be proximate. Based on this similarity function, we developed a new bitmap indexing technique that showed surprisingly pleasant performance. This efficiency come from the efficient bitwise AND and OR operations to find relevant objects with complex features. Up to now, there have been rare efforts to deal with complex similarity queries. It has been demonstrated that our bitmap index handles the complex similarity queries in a very efficient and natural manner. Moreover, the bitmap index does not restrict the number of features to be indexed in advance, and therefore the indexing of the variable number of features can be supported effectively.

Acknowledgements

This study was financially supported by Seoul National University of Science and Technology (2014-1703).

REFERENCES

[1] Hinneburg, C.C. Aggarwal, and D.A. Keim, "What is the nearest neighbor in high dimensional spaces?," Proc. VLDB Conf. 2000.

[2] C.C. Aggarwal and P.S. Yu, "The IGrid Index: Reversing the Dimensionality Curse for Similarity Indexing in High Dimensional Space," Proc. ACM SIGKDD, pp. 119-129, 2000.

[3] R. Fagin, "Combining Fuzzy Information from Multiple Systems," Proc. ACM Symp. on PODS, pp. 216-226, 1996.
[4] G.-H. Cha and C.-W. Chung, "Object-Oriented Retrieval Mechanism for Semistructured Image Collections," Proc. ACM Multimedia Conf., pp. 323-332, 1998.
[5] 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 Trans. on Multimedia, Vol. 4, No. 1, pp. 76-87, March 2002.
[6] H. Ferhatosmanoglu et al., "Vector approximation based indexing for nonuniform high dimensional datasets," Proc. ACM CIKM, pp. 202-209, 2000.
[7] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," Proc. of VLDB Conf., pp. 194-205, 1999.
[8] J. MacQueen, "Some methods for classification and analysis of multivariate observations," Proc. 5th Berkeley Symp. Math. Statist, Prob., 1:281-297, 1967.

Guang-Ho Cha. "A New Bitmap Indexing Method for Complex Similarity Queries." International Journal of Engineering Research and Applications (IJERA), vol. 08, no. 01, 2018, pp. 25–29.