Fast Similarity Search for High-Dimensional Dataset

Quan Wang and Suya You
Computer Science Department
University of Southern California
{quanwang,suyay}@graphics.usc.edu

Abstract

This paper addresses the challenging problem of rapidly searching and matching high-dimensional features for the applications of multimedia database retrieval and pattern recognition. Most current methods suffer from the problem of dimensionality curse. A number of theoretical and experimental studies lead us to pursue a new approach, called Fast Filtering Vector Approximation (FFVA) to tackle the problem. FFVA is a nearest neighbor search technique that facilitates rapidly indexing and recovering the most similar matches to a high-dimensional database of features or spatial data. Extensive experiments have demonstrated effectiveness of the proposed approach.

1. Introduction

Similarity search is crucial to multimedia database retrieval applications, for example, searching for correspondences between objects or retrieving multimedia contents from databases. The similarity search involves finding the most similar objects in a dataset to a query object based on a defined similarity metric. To achieve a robust and effective querying, many current multimedia systems use highly-distinctive features as basic primitives to represent original data objects and perform data matching. While these feature representations have many advantages over original data including distinctiveness, robustness to noise, and invariance and tolerance to geometric and illumination distortions, they typically produce high-dimensional feature spaces that need to be searched and processed. Methods that search exhaustively over the high-dimensional spaces are time-consuming, resulting in painfully slow evolution of such multimedia databases. Efficient search strategies are needed to rapidly and robustly screen the vast amounts of data that contain features and objects of critical interest to users' applications.

This paper addresses the challenging problem of searching and matching high-dimensional feature sets (e.g. over 100 dimensions for each feature vector) for the applications of multimedia database retrieval and pattern recognition. Traditional tree-structure techniques hierarchically partition or cluster the entire data space into several subspaces and then use special tree structures to index objects. These types of approaches are suitable for nearest neighbor (NN) searching of datasets with low dimensionality, but their performance could rapidly degrade when directly adapting to high dimensionality. The limitations of simply modifying or adapting these techniques to high-dimensional datasets are severe. This is referred to "curse of dimensionality" [1] and places a practical limit on the partitioning based techniques. It has been shown in [2] and our experiments, that using the hierarchical partitioning and indexing structures for searching beyond a certain dimension becomes even worse than an exhaustive sequential-scan.

Recently, there have been great efforts in developing vector approximation (VA) techniques such as VA-File [3] intending to overcome the limitations of the tree-structure approaches. Instead of partitioning the input data space hierarchically, the vector approximation methods directly index the objects based on linear and flat structure. While the VA approaches have several inherent problems, they demonstrate better performance in high-dimensional feature retrieval, and do not suffer from the problem of dimensionality curse.

This paper presents an improved vector approximation method, called Fast Filtering Vector Approximation (FFVA), for rapidly searching and matching high-dimensional features from large multimedia databases. FFVA is an NN-search technique that facilitates rapidly indexing and recovering the most similar matches, i.e. k-NN, in a high-dimensional database of features or spatial data. Comparing with several existing techniques including exhaustive linear scan, KD-tree [4], Best-Bin-First [5], and VA-File [2, 3], the FFVA has demonstrated better

performances in terms of query exactness, data access rate, query speed, and memory requirement.

2. Previous work

KD-tree is a widely used NN-search algorithm with sub-linear complexity [4]. It hierarchically partitions the input data and indexes objects using special tree structure. KD-tree performs well for low-dimensional datasets if there are many tree branches that can be pruned and consequently only a small fraction of data needs to be processed. The KD-tree approach and its variances, however, become inefficient when the dimensionality of data increases.

Weber, et al [2] conducted a quantitative analysis of various NN-search methods in high-dimensional vector spaces. It shows that when the dimensionality of data excesses around 10, the searching and indexing techniques based on hierarchical data-partitioning, including KD-tree, R-tree [6], R*-tree [7], SR-tree [8], M-tree [9], and TV-tree [10] can be easily outperformed by a simple sequential-scan approach.

X-tree [11] has demonstrated good performance for datasets with medium dimensionality (e.g. 30-dimension). X-tree is constructed in such a way that the minimum bounding-hyperrectangle of two splitting sets is minimal. The concept of supper node is introduced intending to tackle the problem of dimensionality curse. Experiments showed that the number and size of the supper nodes increase with dimensionality.

Best-Bin-First (BBF) [5] performs an approximate searching over KD-tree structure to accelerate the search speed for high-dimensional datasets. The tree-nodes are examined in an ascending order according to the minimum distance from the query point to the bin containing the leaf nodes. Similarity search terminates after certain number of leaf nodes are examined.

Another line of research is to use linear and flat data structures to index objects [2, 3, 12, 13, 14, 15, 16]. VA-File suggested in [3] has shown to be an effective technique for high-dimensional databases. The method divides the original data space into rectangular cells quantized as bit-vector representations. Similarity search of a query object is performed by first screening the data using lower/upper-bound distances, and then refining the candidates. VA-File, however, suffer from the assumption of database uniform distribution, which is not always applicable to complex data, especially a heterogeneous multimedia dataset. There are several variant approaches intending to overcome the problem. Ferhatosmanoglu [12] improved the VA-File search by using non-uniform bit allocation and optimal statistic quantization. Weber [13] discussed the issue of selecting appropriate bit number for each dimension, and also suggested a formula for evaluating the bound error distribution. Furthermore, a parallel VA-File search technique with multiple workstations was investigated in [14].

3. FFVA for Similarity Search

FFVA is an improved version of VA-File method to achieve fast vector approximation and indexing. Figure 1 illustrates the structure of FFVA and its major components for an efficient nearest neighbor search.

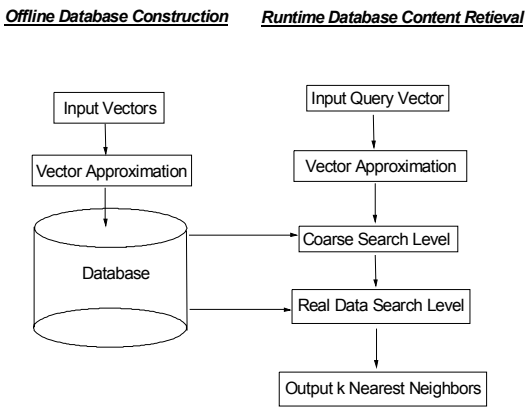

Figure 1: Structure of FFVA approach

The basic data structure of FFVA and standard VA-File method is a space-partition-table (SPT). Each dimension of input feature vectors is quantized as a number of bits used to partition it into a number of intervals on that dimension. In the whole vector space, each rectangle cell with the bit-vector representation approximates the original vectors that fall into that cell, resulting in a list of vector approximations of the original vectors. It is noteworthy that our FFVA method clusters the original vectors to the corners of SPT cells, thereby enable us to use a list of corners, instead of cells, to approximate the original vectors. Such strategy is efficient for the following fast lower bounds filtering.

There are two major levels involved in FFVA NN-search: 1) *coarse search level* to sequentially scan the approximations list and eliminate a large portion of data, and 2) *real data search level* to calculate accurate distances of resultant candidates and decide the final k-nearest neighbors.

In previous works, lower-bound (the Euclidean distances from the SPT cell's nearest corner to a query point) are used for the coarse search. Experiments show that the approximation quality and computation

cost of the bound-distances determinate the performance of the entire searching system [12, 16].

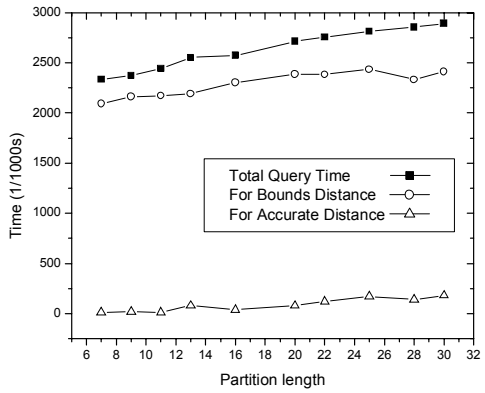

Figure 2: Query time of VA-File

Figure 2 shows the total time (sum of 900 queries) of a typical VA-file method query using real data sets (128-D features). According to the graph, the calculation of lower-bounds takes around 90% of the total querying time. This result is consistent with the experimental results in [17]. Therefore, a more efficient method for lower-bounds computation is essential for improving the entire searching performance.

In the coarse search level, only the vector approximations are accessed and ***block distance*** is used as similarity metric, which is calculated by employing the Manhattan distance of corresponding corner points of two SPT cells:

$$BD = \sum_{i=1}^{n} | v_{1i} - v_{2i} |$$

where v_{1i} and v_{2i} are the coordinates of two corresponding corner points in a n-dimension space.

Let "*max_bd*" represents the longest ***exact distance*** (squared Euclidean distance) from a query point to current k-nearest neighbors. Since in our experiments, all the vector coordinates are integers so the exact distance is strictly lower-bounded by the block distance, whenever we encounter an approximation whose block distance from the query point is larger than *max_bd*, it can be guaranteed that at least *k* better candidates have already been found. Therefore, we can eliminate data with block distances larger than *max_bd*. Updating *max_bd* is also fast, as we dynamically sort and maintain the k-NN structure.

Only those candidates with block distance no larger than *max_bd* will enter the real data search level. In this level, their original vectors are accessed in order to calculate their exact distances. If the exact distance turns out to be shorter than any of current k-NN distance, the k-NN as well as *max_bd* will be updated.

Table 1 outlines the algorithmic structure of the FFVA approach.

Table 1: FFVA algorithmic structure

Input: query and database vectors

Output: *kNNQ* containing k-nearest neighbors

/* Note: *kNNQ* is a list containing the k-nearest neighbors found so far, sorted according to ascending order of exact distance from a query point */

1. Calculate or load (if pre-computed) the approximations of database vectors: *aprov* and query point: *aproq*;

2. Initialize *kNNQ*;

3. *max_db* = maximum possible value;

4. For each approximation *aprov* of database vectors

```
{
    current_bd = block distance between aprov and
     aproq;
    if (current_bd < max_db)
    {
        Calculate the corresponding actual distance;
        Insert this vector into kNNQ, if it is closer
        to query point than the last element in
        kNNQ;
        Update max_db to be the distance from
        query point to the last element of k-NN
        found so far;
    }
}
```

4. Experimental results

In this section, we provide extensive performance evaluation and comparison of the proposed FFVA approach with four commonly used k-NN search techniques: exhaustive linear scan, standard KD-tree, BBF, and standard VA-File. The tests in this section cover: search accuracy, data access rate, query speed, and memory requirement.

Both synthetic and real data are used in our experiments. The real data sets are large number of high-dimensional feature vectors (128-dimension for each feature) generated from a range of real images containing various object and scenes. SIFT (Scale Invariant Feature Transforms) approach [18] was used

to extract the image features described as 128-dimension vectors for feature matching.

In our test, the partition length for VA-File and FFVA methods is 15. For BBF, the size of buffer for "best bins" is 25 and the limit number of examined nodes is 100.

During the similarity search, top two nearest neighbors (2-NNs) were returned. To evaluate the matching correctness, we used the *distance ratio* as evaluation criteria, that is: "the second closest neighbor should be significant far away from the closest one"[18]. A threshold value of 0.6 was used throughout the experiments.

4.1. Search accuracy

In the accuracy test, we randomly pick up 1000 vectors from a synthetic database served as query vectors (test set). Since the test set is a subset of the database, ideally a perfect match of a query vector should have zero distance.

Table 2: Matching accuracy of FFVA

Dimension	60	70	80	90	100	110	115
Correct matches	1000	999	1000	1000	998	1000	1000
Dimension	120	125	130	135	140	145	150
Correct matches	1000	1000	1000	1000	1000	1000	1000

Table 2 summarizes the number of correct matches (out of 1000 queries) of the proposed FFVA method to the synthetic database. Gaussian noise (variance = 0.3) is added to the uniform distributed vectors in the test set.

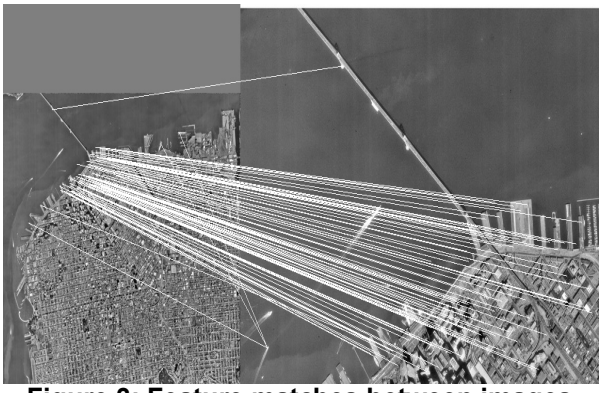

Figure 3: Feature matches between images

We also conducted tests on various real data sets. Overall, the matching performance is consistent with the results of above synthetic data. Figure 3 shows the feature matching discovered by FFVA between two aerial images with an overlapping area. White lines indicate the computed feature matches.

4.2. Data access rate

Data access rate is an important aspect to evaluate the effectiveness of an approach for very large high-dimensional database retrieval problem. KD-tree and BBF approaches utilize hierarchy tree structure to skip the nodes that are not along the searching path. FFVA and VA-File methods use compact vector approximations to avoid accessing the majority of the original database vectors.

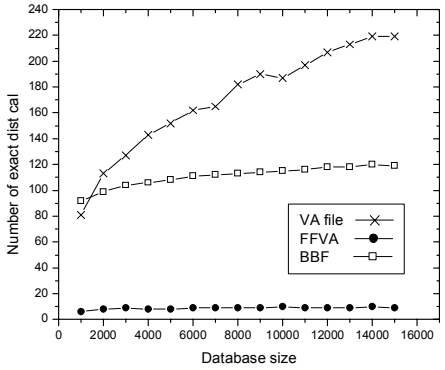

Figure 4: Data access rate test (results are the averages of 1000 queries)

Figure 4 illustrates the test results of data access rates for three methods. The FFVA NN-search method clearly demonstrates the best performance. Its lower-bound is much tighter than that of standard VA-File method. As a result the proposed lower-bound computation based on block distance is efficient in reducing the amount of necessary data access and distance calculation. In this experiment of synthetic data, less than 10 exact distance calculations are needed for FFVA to find the 2-nearest neighbors among 15,000 120-dimension feature vectors, while other two methods (BBF and standard VA-file) spend 10-20 times more for the exact distance calculations.

4.3. Query speed

We tested the query speed of various data sets. In this test, we assume that all the feature vectors and their data structures are loaded into main memory (i.e. full memory access) to perform NN-search. The total query time also include the times spent on the tree construction (BBF) and vector approximation (VA-file and FFVA) processes.

Figure 5 shows the testing results (total query time of 100 queries) of synthetic data with dimension fixed at 100. Under full memory access assumption, BBF demonstrates the best performance among the three approaches when the database excesses certain size.

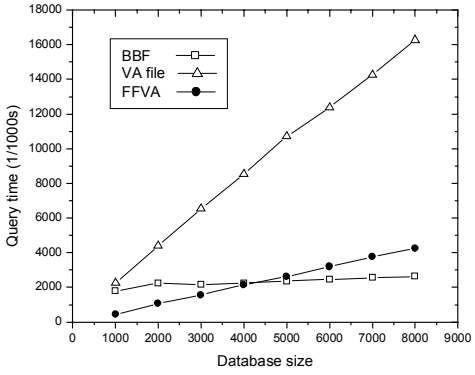

Figure 5: Query speed test (synthetic data)

Figure 6 shows the test results of real data. The test data are collections of 128-dimension image features extracted from image pairs using SIFT approach. The total query time is divided by the number of features in the query image to obtain the search time per query. An exact NN-search using hierarchy structure becomes even slower than exhaustive sequential-scan when the dimensionality is high [2]. Standard VA-file approach spends over much time for bounds-distances calculation when facing non-uniform real data, while FFVA and BBF demonstrate better and comparable performances.

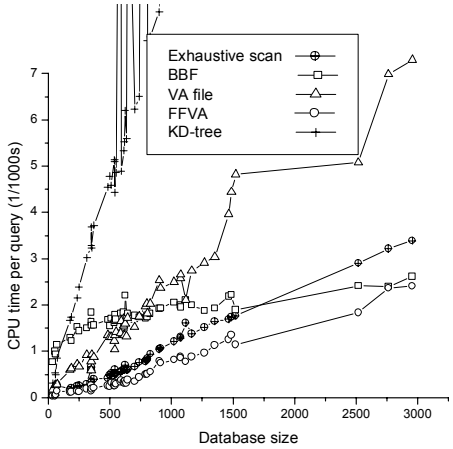

Figure 6: Query speed test (real data)

4.4. Memory requirement

Memory usage has to be considered in developing an effective algorithm for similarity search of large databases. BBF or similar tree-structure approaches typically needs to load and store entire data sets into system memory for online processing: constructing tree structure, iteratively tracing the tree branches, and searching for the optimal tree nodes. This strategy apparently is impractical for searching very large databases when the data size easily overwhelms the limit of available memory space.

One of the major advantages of FFVA and VA-File is their low memory requirement provided by the flat and linear SPT data structure. Figure 7 shows the memory usages of three methods, BBF, VA-file and FFVA to query a real feature database containing 3,000 feature vectors.

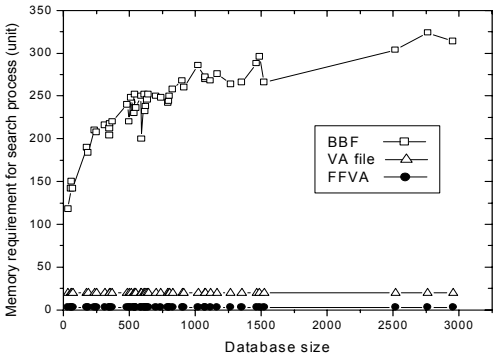

Figure 7: Test for memory requirement (note: one unit approximately equals to 512 bytes)

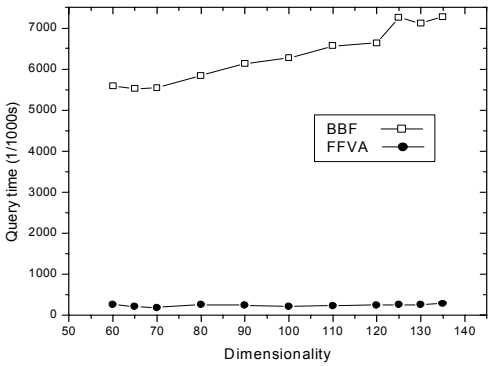

Figure 8: Test for query speed considering memory paging (database size: 100,000; page size: 2000 units; query times were averaged from 3 queries)

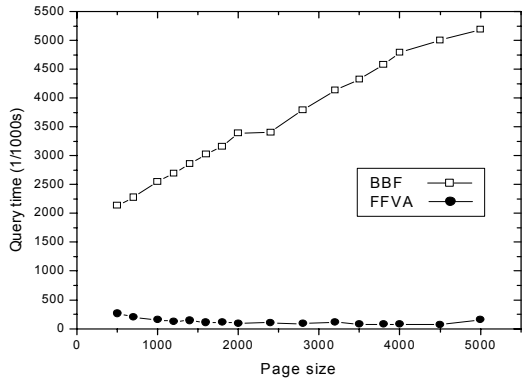

Figure 9: Test for query speed considering memory paging (database size: 50,000; feature dimension: 120; page size ranges from 240KB to 2.4MB)

Figure 8 and 9 show the relationship among query time, data dimensionality and memory blocks (i.e. memory pages). These results clearly indicate that the memory usages of FFVA are far more effective than that of BBF. Therefore, FFVA is suitable for the application of large database retrieval or the systems that have limited memory spaces such as mobile computing devices.

5. Conclusion

This paper presents a new similarity search method, Fast Filtering Vector Approximation (FFVA), for rapidly searching and matching high-dimensional features from large multimedia databases. FFVA is a nearest neighbor search technique that facilitates rapidly indexing and recovering the most similar matches to a high-dimensional database of features or spatial data. We evaluate extensively the proposed method. Comparing with several existing techniques, the method has demonstrated better performances in terms of query correctness, data access rate, query speed, and memory requirement.

References

[1] R. Bellman. Adaptive Control Processes: A Guided Tour. Princeton University Press, 1961.

[2] Roger Weber, Hans-J. Schek, Stephen Blott, A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Space, *Proc. 24th Int. Conf. Very Large Data Bases, VLDB,* 1998.

[3] S. Blott, R. Weber. A Simple Vector-Approximation Fi-le for Similarity Search in High-dimensional Vector paces. *Technical Report 19, ESPRIT project HERMES (no.9141)*, m-arch 1997.

[4] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209--226, 1977.

[5] Jeffrey S. Beis and David G. Lowe, Shape indexing using approximate nearest-neighbor search in high-dimensional spaces, *CVPR*, 1997.

[6] A. Guttman. R-trees: A dynamic index structure for spatial searching. *In Proc. of the ACM SIGMOD Int. Conf. On Management of Data*, pages 47-57, Boston, MA, June 1984.

[7] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. *In Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322-331, Atlantic City, NJ, 23-25 May 1990.

[8] N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. *In Proc. of the ACM SIGMOD Int.Conj.on Management of Data,* pages 369-380, Tucson, Arizon USA, 1997.

[9] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. *In Proc. of the Int. Conference on Very Large Databases*, Athens, Greece, 1997.

[10] K.-I. Lin, H. Jagadish, and C. Faloutsos. The TV-tree: An index structure for high-dimensional data. *The VLDB Journal*, 3(4):517-549, Oct. 1994.

[11] S. Berchtold, D . Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. *In Proc. of The Int. Conference on Very Large Databases*, pages 28-39, 1996.

[12] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, A. El Abbadi, Vector Approximation based Indexing for Non-uniform High Dimensional Data Sets, *CIKM: ACM CIKM* 2000.

[13] Roger Weber, Klemens Böhm, *Trading Quality for Time with Nearest-Neighbor Search*, Lecture Notes in Computer Science, 2000.

[14] R. Weber, K. Böhm, Hans-J. Schek, Interactive-Time Similarity Search for Large Image Collections Using Parallel VA-Files, *ICDE,* 2000.

[15] Stefan Berchtold, Christian Böhm, H. V. Jagadish, Hans-Peter Kriegel, Jörg Sander, Independent Quantization: An Index Compression Technique for High-Dimensional Data Spaces, *ICDE,* 2000.

[16] Ertem Tuncel, Hakan Ferhatosmanoglu, Kenneth Rose, VQ-Index: An Index Structure for Similarity Searching in Multimedia Databases, *ACMMM*, 2002.

[17] Paolo Ciaccia, Marco Patella, "PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces," *icde*, p. 244, 16th International Conference on Data Engineering (ICDE'00), 2000.

[18] David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, *IJCV*, 2004.