UDC 004.422.635.3, DOI: 10.2298/CSIS1001139H

MFI-Tree: An Effective Multi-feature Index Structure for Weighted Query Application

Yunfeng He and Junqing Yu1

1School of Computer Science & Technology,
Huazhong University of Science & Technology, 430074 Wuhan, China
yjqing@hust.edu.cn

Abstract. Multi-Feature Index Tree (MFI-Tree), a new indexing structure, is proposed to index multiple high-dimensional features of video data for video retrieval through example. MFI-Tree employs tree structure which is beneficial for the browsing application, and retrieves the last level cluster nodes in retrieval application to improve the performance. Aggressive Decided Distance for kNN (ADD-kNN) search algorithm is designed because it can effectively reduce the distance to prune the search space. Experimental results demonstrate that the MFI-Tree and ADD-kNN algorithm have the advantages over sequential scan in performance.
Keywords: Multi-Feature Index Tree; KNN; Aggressive Decided Distance for kNN; Video Retrieval.

1. Introduction

With the development of multimedia and network technologies, it is much easier to generate, access and manipulate video data than ever before. Facing the massive video data, traditional retrieval methods are not efficient by using only metadata of video, such as the name of video and the creator. People care more about the content of video data, so content-based video retrieval (CBVR) [1] is becoming an active research area in the area of video databases. There are two main methods for video retrieval in CBVR: one is semantic retrieval which matches the retrieval keywords with the semantic keywords extracted from video data; the other is sample retrieval which calculates the similarity distances between the features extracted from the sample, such as an image, a piece of video or audio etc, with the features of video data. Semantic retrieval method is simple and effective, but unfortunately, the existed technologies of CBVR still suffer from the semantic gap because computers can not directly "calculate" the semantic meaning from low-level features of video data, such as color, shape, texture, motion and audio information.

For sample retrieval method, it is not easy for people to find the right sample he wants. Normally, people browse the video database and select an image or a video clip as a sample to retrieve. Therefore, browsing is quite important for sample retrieval method. In CBVR, video contents are usually described by multiple features, each of which is typically high-dimensional. For example, in MPEG-7 (Multimedia Content Description Interface) [2], a shot of video may be described by a 29-dimentional camera motion feature, and one keyframe of a shot may be described by a 12-dimentional color layout feature and a 31-dimentional homogeneous texture feature, etc. To support multi-feature queries, a high-dimensional index is needed to be built. Existed high-dimensional indexing technologies, such as M-tree [3] and VA-File [4], typically treat all different features homogeneously, which means the similarity distance is based on a static combination of feature weights. However, in sample retrieval, the weights of features are different for different people with different understanding to the sample. For example, an image sample is described by a color feature and a shape feature. Some people like its color and retrieve by using the weightages of (0.8, 0.2) for the color and the shape feature, while some people think the color feature is as the same important as the shape feature, and retrieve using the weightages of (0.5, 0.5). what's more, multiple high-dimensional features become ineffective with the dimension increasing.

In this paper, we propose a new indexing structure called Multi-Feature Index Tree (MFI-Tree) and a uniform similarity distance function is applied to ensure that the distance value of two objects is the one and only one in MFI-Tree building processing. MFI-Tree is a hierarchical tree structure which has two kinds of node, leaf node and cluster node. Leaf node represents a video data in the set, while cluster node represents an aggregate including some leaf nodes with a close distance. Division algorithm is important for the building and updating of MFI-Tree. Here, a new division algorithm which obtains several separate subsets is employed. To support K Nearest neighbors (kNN) queries, we propose a novel searching algorithm called ADD-kNN (Aggressive Decided Distance for kNN). To reduce the high-dimensional effect, ADD-kNN directly search cluster nodes in the last level of MFI-Tree. ADD-kNN is proved to be an efficient filter-and-refine approach which fast decreases the filtering value to avoid accessing data regions without objects belonging to the result-set.

The rest of the paper is organized as follows: related work is reviewed in section 2, while the structure of MFI-Tree and ADD-kNN searching algorithm is discussed in section 3 and section 4 respectively; in section 5, we make some experiments to evaluate the performance of the MFI-Tree and ADD-kNN algorithm; section 6 contains our conclusion and future work.

2. Related Works

The purpose of indexing is to improve the performance of queries. But for 30~50 dimension data, existed indexing techniques have failed to improve the performance of sequential scan due to the known "dimensionality curse" [5, 6]. To solve this problem, proposals in researches approximately belong to three categories: dimensionality reduction, one-dimensional transformation, and data approximation [7].

Dimensionality reduction method maps the high-dimensional space into a low-dimensional space. The low-dimensional space is composed by some most important dimensions based on the correlation analysis of different dimensions, which is easy to be indexed by existed indexing techniques. For example, the dimension of color layout feature in MPEG-7 is reduced from 192 to 12. One-dimensional transformation includes Pyramid-Technique, iDistance, iMinMax, etc. The Pyramid-Technique [8] divides D-dimensional data space into 2-dimensional pyramid areas which share the center point of the space as a top and then cuts each pyramid area into slices，each of which forms a data page. Then the D-dimensional space is mapped to 1-dimensional space. IDistance method transforms a high-dimensional point into a 1-dimensional distance value with reference to its corresponding reference point [9]. IMinMax method maps points in high dimensional spaces to single dimension value determined by their maximum or minimum values among all dimensions [10]. One-dimensional transformation is efficient， however, because of the information loss in transformation, many candidates which are not results are calculated. In data approximation method, indexing is built on small and approximate representations which represent original data, such as VA-FILE (Vector Approximation File) [4]. The VA-FILE uses small vectors to represent the original data point and then sequentially scan the vector files to obtain candidates. However, the performance of VA-FILE is limited due to sequential scan. What's more, VA-FILE does not adapt to highly skewed data. Extended from VA-FILE, OVA-FILE uses the ordered approximation file where the approximations close to each other in data space are placed in the close positions based on VA-FILE [11].

With the development of multimedia database technology, there are some research works on multi-feature indexing structure. In [12], a single M-tree index is constructed for all the features, and principle component analysis and neural network is used to reduce dimension. But neural network training process is undesirable for very large data sets and M-tree structure is degraded in performance for dimensionality larger than 20. In [7], a multi-feature indexing structure using dimensionality reduction and B+-tree is proposed. Each feature is represented by two components: one is a 2-dimensional vector obtained by transforming each feature into minimum and maximum of a distance range, and the other is a vector of bit signatures which are set by analyzing each feature's descending energy histogram. This representation can effectively prune away points that are impossible to speed up query processing. However, B+-tree indexing is not suitable for browsing.

In fact, multi-feature indexing structures are given great attention with the development of video retrieval technology. But the existed researches put more emphasis on the indexing structure for the solution of "dimensionality curse" but less emphasis on the indexing system. The MFI-Tree structure and ADD-kNN searching algorithm proposed in this paper are suitable for retrieval and browsing application in video database.

3. MFI-Tree Structure

3.1. Uniform similarity distance function

In multi-feature video retrieval application, the similarity distance between two video objects is different for different weightages corresponding to different results. Set $F=(F_1, F_2, \ldots, F_n)$ is a video data point described by n features, where F_i is the i th feature and it is comprised of d_i dimensions. Thus, the similar distance function between video data point P and point O is following:

$$\begin{cases} Dist(P,O) = \sum_{i=1}^{n} w_i Dist_i(P,O) \\ \sum_{i=1}^{n} w_i = 1 \quad w_i > 0 \end{cases}$$

where $Dist_i(P, O)$ is the normalized distance value between point P and point O on the i th feature, and w_i is the weight that describes the importance of the i th feature.

For video data, different features describe different content of video, and apply different distance function. The distance of different feature has different value range. To generate a distance representing all features, each feature distance has to be normalized. We normalize $Dist_i$ (P, O) into the range of [0,1] by the following normalization formula:

$$Dist_i(P,O) = \frac{Dist_i'(P,O)}{Dist_{i\max}}$$

Where $Dist_{i\max}$ is the maximal distance value of the i th feature on two video data points, and $Dist_i'(P, O)$ is the distance value between point P and point O on the i th feature. Here, the minimal distance value is not used because minimal distance value is easy to change when new object is inserted into the data set.

Note that the different weightages cause different distance value between P and O, but indexing building depends on the unique distance value between these two points. Therefore, a similar distance function for indexing building is applied. From normalization formula, it is easy to know that the distance value between two points is always less than the maximal distance value in the normalized distance value on n features. We can use the following formula to generalize the similar distance between two points:

$$Dist(P,O) = \max(Dist_i(P,O))$$

3.2. MFI-Tree Structure

MFI-Tree is a hierarchical tree structure which satisfies the need of video retrieval and browsing. Just like the M-tree, MFI-Tree has two kinds of node, leaf node and cluster node. Leaf node is used to save all video data points in video set, and figure 1 illustrates the structure of leaf node.

NodeID	FatherID	Feature F_1	…	Feature F_n	Dis

Fig. 1. The structure of leaf node in MFI-Tree

Where NodeID is the identification of current leaf node, and FatherID is the parent node ID of current leaf node, Feature F_i is the i^{th} feature value, and Dis is the distance value between current leaf node and the center object of the parent node.

Cluster node is used to save the cluster of some close leaf nodes. Different from the routing node of M-tree, the cluster node of MFI-tree is used not for retrieval, but for browsing to reduce the influence of the high-dimensional. The cluster node structure is shown in figure 2.

NodeID	FatherID	R	CenterID	BrowserID	Leaf-Num	isRoot

Fig. 2. The structure of cluster node

Where NodeID and FatherID are used to build hierarchical tree structure for browsing, R is the covering radius of current node, namely the maximal distance of all the distances between each son node and the center node. CenterID and BrowserID is the identification of the center object and browsing object of current cluster node. LeafNum is the number of son node in current cluster node, and isRoot is the flag to show whether current node is the last level cluster node or not. When cluster node generates, a virtual point which does not really exist in video data set is used to be center object for decreasing the covering radius and the overlap area of cluster nodes. The browsing objects which present current cluster in browsing application can be saved to speed up the response time.

3.3. Building of the MFI-Tree

The building process of the MFI-Tree can be treated as a process that a large data set divides continuously into several small data sets，which needs three steps to accomplish.

Step 1: to calculate the maximal distance values of each feature. This is one of the main characteristics different from other dynamic indexing structures. For the distance value is normalized by maximal distance value of each feature, the changed maximal distance values of each feature may disable the MFI-Tree. That is why the MFI-Tree is called a semi-dynamic indexing structure. The algorithm of finding the maximal distance values of each feature is shown in figure 3.

```
    FindMaxDistance
    Input:   X：video data set
             $i$：the number of feature order
    Output:  $dis$：the maximal distance on $F_i$
    Steps:
    1. Select an arbitrary object x0 in X；
    2. Visit all object in X and find object $x_1$ which has the
maximal distance on $F_i$ with $x_0$;
    3. Visit all object in X and find object $x_2$ which has the
maximal distance on $F_i$ with $x_1$;
    4. Visit all object in X and find object $x_3$ which has the
maximal distance $dis$ on $F_i$ with $x_2$;
    5. Output $dis$。
```

Fig. 3. Illustration of the maximal distance finding algorithm.

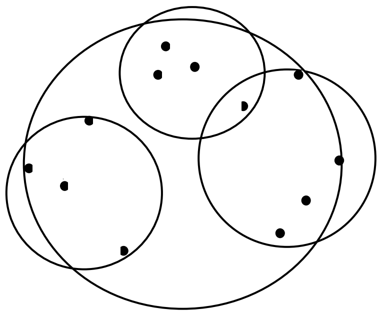

Fig. 4. Set division illustration in MFI-TREE.

Step 2: to divide data set. Considering browsing application, MFI-Tree does not divide data set into two subsets, but divide into several subsets based on the distribution of data sets. Figure 4 gives the illustration of the set division process. Firstly, we find the farthest pair of object A and B in data set using the algorithm like finding the maximal distance algorithm, and then object A and B are inserted into two new subsets separately. Secondly, calculate the minimal distance value between objects which is not distributed with the first object of each subset. The minimal distance value is denoted to d_{min}. If d_{min} is less than threshold value AddDis, it means the object is close enough to the corresponding subset and is inserted into it, such as object K, L, H and G in figure 4. If d_{min} is more than threshold value NewSetDis, it means the object is far away from all subsets, such as object C in figure 4, then a new subset is created and the object is inserted into the new subset. If d_{min} is more than AddDis and less than NewSetDis, the object does not execute insert processing, such as the object D, I and J in figure 4. Finally, calculate the minimal distance value between objects which is not distributed with the first object of each subset, and the object is inserted into the corresponding subset. The algorithm of dividing the data set is shown in figure 5.

```
DivideSet
Input: X: a data set to be divided
Steps:
  1. Set subset list SetList and object list ObjList are NULL;
  2. Find the farthest pair of object a0 and b0 in X, set A={a0} and B={b0}, delete a0 and b0 from X and insert A and B into SetList;
  3. When X is not NULL
    (1) Calculate the minimal distance $d_{min}$ between object with the first object in each subset and sign the subset;
    (2) If $d_{min}$<AddDis, move object into the signed subset;
    (3) If $d_{min}$>NewSetDis, create a new subset C, move object into C and insert C into SetList;
    (4) Else insert object into ObjList;
  4. When ObjList is not NULL
    (1) Calculate the minimal distance $d_{min}$ between object with the first object in each subset and sign the subset;
    (2) Move object into the signed subset;
  5. Generate new cluster nodes.
```

Fig. 5. Set division algorithm in MFI-Tree

After all objects in data set are inserted, all subset nodes are generated. For each subset, the center point of the farthest pair of objects is the center object of the subset, the maximal distance value between objects in the subset and the center object is the covering radius of the subset, and the object which has minimal distance to the center object is the browsing object of the cluster.

In the division algorithm, two important threshold values are used. NewSetDis is used for creating a new subset and is calculated by the following formula:

$$NewSetDis = \delta \cdot D_{\max}$$

where D_{max} is the maximal distance between two objects in the data set to be divided, and the δ is the degree of the division. Normally, new subsets are created when δ is more than 0.5. But for the distance function of feature is not Euclid distance and the distance value is the maximal distance value, δ is usually from 0.6 to 0.8 due to reducing the number of subsets. 0.7 is used in our experiments. AddDis is used for the inserting operation and is normally equal to half of the NewSetDis.

Step 3: to check new subset. If the subset meets the conditions of division, the subset can execute division, else stop. There are two conditions for set division. One is LeafNum, which is the number of the objects in set. The bigger LeafNum increases the calculation, but the smaller LeafNum increases the number of cluster nodes and reading times. The other is the covering radius R. The bigger R decreases the performance of pruning, while the smaller R increases the number of cluster node. Normally, LeafNum is depending on the organization of data. And the maximal R value is different for different application. In multi-feature indexing structure, most data points do not cluster into a small area because of using the maximal feature distance as object distance. The maximal of R is from 0.1 to 0.3, and in our experiments, this value is set to 0.3.

3.4. Insertion and Deletion Algorithm

When the MFI-Tree is generated, the insertion and deletion operations can be enabled if the maximal distance values of each feature are not increased.

In the process of inserting, the first is to find all cluster nodes in the last level of MFI-Tree, and to calculate the minimal distance between the insert object and the cluster node. The second is to insert object into corresponding cluster node and update the LeafNum and R of the ancestor nodes up to top. Finally, if the cluster node to be inserted meets the conditions of division, this cluster node executes division operation.

The deletion operation is similar to the insertion. The first step is to find and delete object from MFI-Tree, and then update the LeafNum value of ancestor nodes up to top. Lastly, if the LeafNum value of the cluster node is smaller than the conditions of division, the father node of this cluster node will delete all its son nodes, and then execute division operation.

4. ADD-kNN Searching Algorithm

Ordinarily, video content similarity queries contain some elementary types, such as Range Query, *k* nearest neighbor query (kNN query), etc. Using a range search, it is difficult to specify a maximal distance as the constraint without some knowledge of the data and distance function. An alternative way is to use kNN query which finds *k* nearest neighbors to the given query object.

There are two ways of approaching kNN query, range query and filtering. For the range query, query data set uses a given distance *R*. If the number of candidates is more than *k*, then *k* nearest neighbors are returned. Else, continually query data set uses increased *R* value until k results is found. Obviously, the performance of the mentioned methods relies on the value of *R*, the bigger or the smaller of the value can decrease the query's efficiency. The distance functions of high-dimensional features have the property of triangle inequality, that is:

$$\forall x, y, z \in D, d(x,z) \le d(x,y) + d(y,z)$$

Filtering method uses triangle inequality to prune away a lot of impossible objects, and then calculate the distance between the query object and candidates. This method can improve the performance by reducing the unnecessary calculation.

Comparing with the traditional indexing structure, MFI-TREE structure has an important characteristic. In the building processing of MFI-Tree, the distance between two objects is the maximal distance value in the normalized distance value on n features. But in query processing, the distance between two objects is calculated by given weightages. That is to say, the covering radius of cluster nodes in MFI-TREE is bigger than the real radius of cluster nodes calculated by given weightages in most cases. Because the covering radius is an important parameter for filtering methods, some useless cluster nodes cannot be pruned away easily by using bigger radius in query processing.

To avoid performance decreasing, a new kNN search algorithm, called Aggressive Decided Distance (ADD-kNN) is proposed. The main idea of ADD-kNN algorithm is to fast decrease the filtering value, and to effectively filter most of unnecessary objects. The ADD-kNN search algorithm can be characterized by using the following steps:

Step 1: create the query object according to the sample, features and its weightages.

Step 2: empty the query node list and result list, and set value D for filtering to zero.

Step 3: different from breadth-first search and depth-first search algorithms, our method traverses all the cluster node in the last level of MFI-Tree, that are all the cluster node whose value *isRoot* is equal to 1.

Firstly, the real distance value d with the given weightages is calculated between the query object and the center object of the current cluster node. Then, the minimal distance value D_{min} is calculated by using the following formula:

$$D_{min} = d - R$$

where R is the covering radius of the current node.

When value D_{min} is less than zero, the query object is in the current cluster, and the objects in cluster node are all candidates. The objects in cluster are inserted into result list one by one according to the real distance to the query object. If the number of the objects in result list is more than k and the real distance is more than filtering value D, the object is not inserted. After insertion operates, if the number of the objects is more than k, the filtering value D is equal to the k_{th} smallest distance value in the result list, and then the object whose distance value is more than D is removed from result list. If the number of objects is less than k, the filtering value D is equal to the biggest distance value of objects in result list.

When value D_{min} is more than zero, the query object is outside the current cluster, and the cluster node is inserted into query node list.

Step 4: when the query node list is not empty, traverse all nodes in query node list. If the number of objects in result list is more than k and value D_{min} of the node is more than the filtering value D, the node is removed from query node list. Else, the objects in the node will execute insertion like the step 3, and then that can be removed from query node list.

Step 5: output the result list.

According to the characteristics of MFI-Tree structure, ADD-kNN research algorithm calculates the real similar distance values which are necessary for reducing the filtering value, decreasing the spending of sorting and improving the performance of retrieval.

5. Experiments

The 12-dimensional color layout feature and the 80-dimensional edge histogram feature in MPEG-7 are used in our experiments because these features are compact and effective to describe image contents. The data set for experiments were conducted by using 400 pieces of video clips from 30 movies, including various movie types like action, comic, comedy and science fiction, etc. The color layout and edge histogram feature are extracted from 10000~30000 images, which are the keyframes derived from the movie clips per 20 frame, using the extraction algorithm and distance function mentioned in MPEG-7. All the experiments were performed on an IBM T61 portable computer. The operating system is windows XP and the database is Oracle9i.

5.1. Effect of data size

We generate MFI-TREE structures and M-tree structures for 10000, 20000 and 30000 keyframes respectively, and then use 6 images derived from test video clip as query objects to execute k-NN query, where k is equal to 20. The experimental results are illustrated in figure 6 (a) and (d). It can be easily found that the average similar distance calculating times of the MFI-Tree structure using ADD-kNN algorithm are less than the calculating times of M-Tree structure. The reason is that the covering radius of current node in M-tree structure which uses uniform similarity distance function is bigger, the k-nn research algorithm of M-tree cannot prune away most nodes, and accessing hierarchic structure of the M-tree structure cause its inefficiency.

5.2. Effect of weighted queries

We use 6 images derived from test video clips as query objects to execute k-NN query with different weightages on the set of 20000 keyframes. Figure 6 (b) and (e) shows the average retrieval time and the average distance calculating times when the weight of colorlayout feature changes from 0.1 to 0.9 while the sum of two feature weights is equal to 1. It is shown that the average retrieval time is the highest when the weight of colorlayout feature is 0.4, and the average distance calculating times is the highest when the weight of colorlayout feature is 0.3, for the retrieval time is decided by distance calculating times and database accessing times.

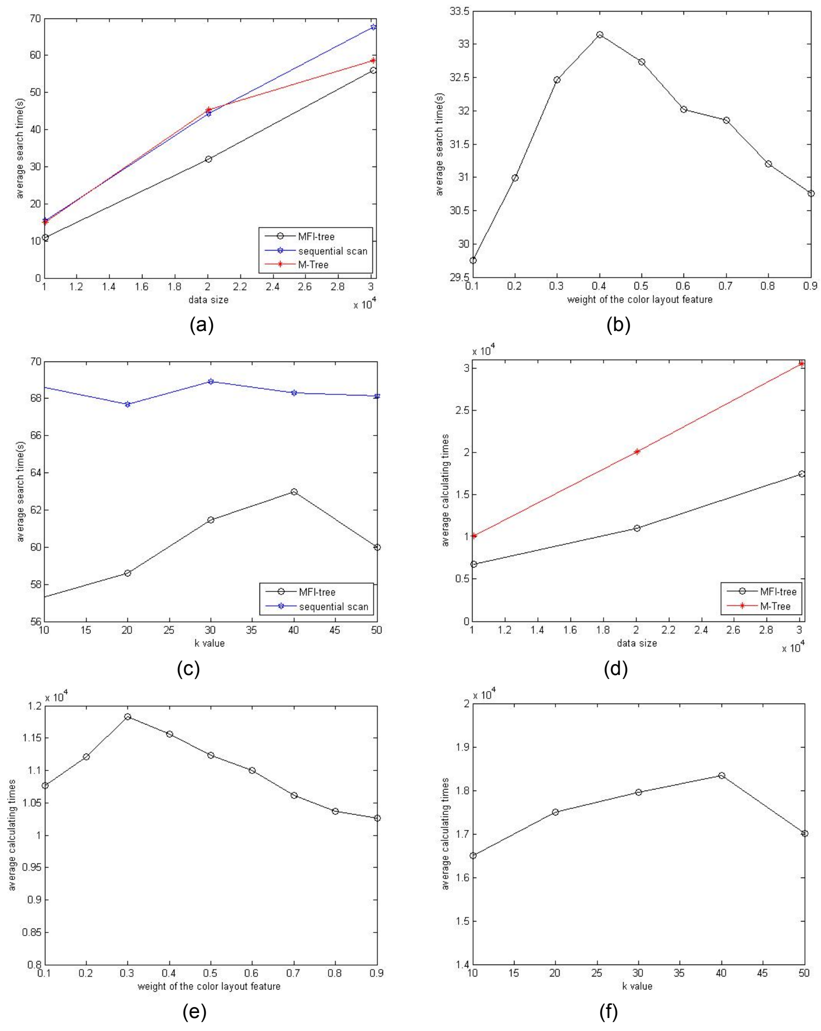

Fig. 6. Experimental results

5.3. Effect of *k* value

We executed kNN queries when the value of k changes from 10 to 50 on the set of 30000 keyframes. The results are illustrated in figure 6 (c) and (f). As we can see, ADD-kNN achieves the worst performance when k value is equal to 40. Actually, the maximal LeafNum value of the cluster nodes in indexing structure definitely influences query performance. Biggish value may lead to biggish covering radius, while less value may lead to more cluster nodes. So, the maximal LeafNum value is decided by the size of data set and the k value which is usually used in applications.

6. Conclusion

In this paper, we have proposed a multi-feature indexing structure——MFI-Tree and ADD-kNN search algorithm for the weighted query application of video retrieval. Based on uniform similarity distance function, MFI-Tree is built to index multi-features of video data. MFI-Tree is a hierarchical structure which can be used for browsing effectively. The ADD-kNN search algorithm directly search the last level cluster nodes in MFI-Tree structure to reduce the high-dimensional effect of the indexing structure, and fast minimize the filtering value to prune most unnecessary data away. The experimental results demonstrate that the MFI-TREE and ADD-kNN search algorithm are effective and efficient for weighted query application.

However, in the division operation of MFI-Tree, the overlap areas of the subsets still exist, which can influence the performance of retrieval. What's more, the features to describe the video content are not easy to be understood by people, so how to create an effective interface for weighted queries application will be one of our future work.

7. Acknowledgement

This paper is financially supported by the National Natural Science Foundation of China under Grant No.60703049; the "Chen Guang" Foundation for Young Scientists of Wuhan under Grant No. 200850731353.

8. References

1. Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain: Content-based multimedia information retrieval State of the art and challenges. ACM Transactions on Multimedia Computing, Communications and Applications, 22(1), 1–19 (2006)
2. ISO/IEC/JTC1/SC29/WG11 (MPEG). Text of ISO/IEC 15938-3 Multimedia Content Description Interface – Part 3: Visual. Final Committee Draft. Singapore, (2001)
3. Paolo Ciaccia, Marco patella, Pavel Zezula. M-tree: An efficient Access method for Similarity Search in Metric Spaces. In proceedings of the 23rd VLDB conference, Athens, Greece, 426-435 (1997)
4. R. Weber, H.J. Schek, and S. Blott: A Quantitative Analysis and Performance Study for Similarity Search Methods in high-Dimensional Spaces. In proceedings of the 24th VLDB Conference, New York, USA, 194-205 (1998)

5. Gislir. Hjaltason and Hanan Samet: Index-Driven Similarity Search in Metric Spaces. ACM Transactions on Database System, 28(4), 517-580 (2003)
6. Christian Bohm, Stefan Berchtold and Daniel A.Keim: Searching in High-Dimensional Spaces-Index Structures for Improving the Performance of Multimedia Databases. ACM Computing Surveys, 33(3), 322-373 (2001)
7. H.V. Jagadish, Beng Chin Ooi, Heng Tao Shen, and Kian-Lee Tan: Toward Efficient Multifeature Query Processing. IEEE Transactions On Knowledge and Data Engineering, 18(3): 350-362 (2006)
8. S. Berchtold, C. Bohm, H.P. Kriegel: The Pyramid-Technique: Towards Breaking the Curese of Dimensionality. In proceedings of Int. Conference On Management of Data, ACM SIGMOD, Seattle, Washington, 142-153 (1998)
9. C. Yu, B.C.Ooi, K.L.Tan, H.V.Jagadish: Indexing the Distance: An Efficient Method to KNN Processing. In proceedings of the 27th VLDB Conference, Roma, Italy, 166-174 (2001)
10. B.C. Ooi, K.L. Tan. C. Yu, S.Bressan: Indexing the Edges-A simple and Yet Efficient Approach to High-Dimensional Indexing. In ACM SIGMOD 19th Symposium on Principles of Database Systems (PODS), Dallas, Texas, 166-174 (2000)
11. Hong Lu, Beng Chin Ooi, Heng Tao Shen, and Xiangyang Xue: Hierarchical indexing structure for efficient similarity search in video retrieval. IEEE Transactions on Knowledge and data engineering, 18(11), 1544-1559 (2006)
12. A.H. Ngu, Q. Sheng, D. Huynh, and R. Lei: Combining Multivisual Features for Efficient Indexing in a Large Image Database. VLDB. 9(4): 279-293 (2001)

Yunfeng He is currently a PhD. Candidate and a member of faculty at HuaZhong University of Science and Technology in China. His research interest focuses on video data index and retrieval.

Junqing Yu is the corresponding author of this paper. He received his PhD in Computer Science from Wuhan University in 2002. He is currently an Associate Professor at the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests include digital media processing and retrieval, multi-core programming environment.

Received: May 14, 2009; Accepted: August 07, 2009.