A Multimedia Data Management Approach with GeM-Tree

Kasturi Chatterjee¹ and Shu-Ching Chen¹,²

¹Distributed Multimedia Information Systems Laboratory, School of Computing and Information Sciences, Florida International University, Miami, FL 33199
²State Key Laboratory of Software Engineering, Wuhan University, 430072, China
Email: {kchat001, chens}@cs.fiu.edu

Abstract—In this paper, we propose a multimedia data management framework using GeM-Tree. GeM-Tree is a multidimensional tree-based index structure which provides a generalized framework to organize and retrieve images and videos seamlessly. In addition to supporting different multimedia data types and diverse representations, the proposed data management framework supports varied multimedia retrieval strategies like content-based image and video retrieval, mixed multimedia data retrieval where cross-similarity between images and videos is considered, region-based retrieval, etc. The framework embeds a high-level semantic relationship between multimedia data objects with a construct called Hierarchical Markov Model Mediator via a novel affinity promotion technique to improve query result relevance manifold. Extensive experiments were conducted with different multimedia data types possessing varied representations, different retrieval approaches, and different data sizes. The encouraging results demonstrate a potential solution to the important and complex issue of managing a large volume of multimedia data.

Index Terms—multimedia data management, content-based retrieval, generalized index structure, varied feature distributions

I. INTRODUCTION

Today, information sharing and communications are the two major directions around which most of the popular web applications are designed and developed. These applications fall under various genres such as social networking, video and image sharing, and web searches, to name a few. As multimedia data like images and videos are way more expressive than text-based information, they have quickly become the preferred information medium to be shared or communicated. However, these extra information in multimedia data (the very cause of their popularity) comes at the cost of complex representation, fuzzy interpretation, and costly maintenance. Thus, it calls for an extensive framework to manage multimedia data with robustness and efficiency comparable to the text-based traditional database management system. Multimedia data is different from traditional text-based data as it is represented with a multidimensional feature vector and have semantic information attached to it. Hence, traditional database management frameworks cannot accommodate multimedia data efficiently and dedicated multimedia data management frameworks are desired.

One of the major requirements for a successful multimedia database management framework is an efficient index structure. The atypical nature of multimedia data asks for index structures which are customized to cater to the specific needs. The multidimensional aspect of multimedia data were addressed in [1] [2] [3] and the various retrieval requirements involving semantic relationships were addressed in [4]. These index structures were specifically designed to handle images and cannot organize videos efficiently. Videos are comprised of several units like frames, shots, etc. and in turn carry semantic information in these units. Internally, these units are organized in a hierarchy where the inter-relationships among the different units need to be stored. [5] was developed to address these issues effectively. These frameworks were structures dedicated to a particular multimedia data type and do not have the capability to serve as a common platform for both images and videos and their different information need.

For the complete development of a database management system, the index structure needs to be embedded to the database kernel successfully. To do so, a number of other components like the Query Optimizer, Query Processor, SQL Compiler/Interpreter, etc. are affected and need to be modified too. Integrating a single type of index structures into the database kernel is itself quite difficult and complicated as pointed out in [6] [7]. Thus, attempting to introduce multiple index structures into the database kernel is not the desired approach. Additionally, even if one succeeds to technically integrate multiple index structures into a single database kernel, it may give rise to conflicting issues. For example, defining a particular set of rules for a Query Optimizer, pertaining to a particular index structure for a certain multimedia object, might have an undesired effect on the performance of another index structure that co-exists in the same database kernel. Therefore, it can be concluded that for a successful multimedia database management framework, a generalized multimedia index structure is necessary.

An index structure has two major functions, namely to organize the data objects and to support the popular retrieval strategies via its k-NN search routine. The popular categories of multimedia data are images and videos. However, there are several retrieval strategies for multimedia data such as content-based retrieval and region-based retrieval, to name a few. The ideal solution is to design an index structure with a plug-in
like architecture which will support different data types along with different retrieval techniques without the need for the user to perform any major modification to the basic framework. In [8], a generalized index structure capable of handling both images and videos seamlessly was proposed. In this paper, we extend its usability in terms of supporting different multimedia retrieval strategies and its handling of high level semantic relationships efficiently from within a single framework. Thus, while [8] laid down a basic generalized multimedia indexing unit, this paper attempts to extend it to a more complete multimedia data management framework. We introduce a cluster-based feature representation of multimedia data in addition to the histogram-based feature representation into the index structure and use the Earth Mover’s Distance (EMD) [9] to calculate the (dis)similarity among them. Though EMD was used earlier in VP-tree [10] to build the VP-EMD tree [11], it is unable to support effective video indexing and retrieval. Furthermore, VP-EMD tree is not a balanced structure and do not support the concept of high-level semantic relationships among data objects. On the other hand, in our framework, the high-level semantic relationship is captured via a stochastic model Hierarchical Markov Model Mediator, and is introduced into the index structure via a novel promotion algorithm. Further, an efficient k-NN based similarity search algorithm is proposed, which can handle different image and video retrievals seamlessly while considering different feature representations. It should be mentioned that the GeM-Tree is a balanced index structure as it was built from a bottom-up approach similar to the M-Tree [2].

The rest of the paper is organized as follows. In Section II, we introduce the Earth Mover’s Distance and its various usefulness in multimedia data management. It is followed by Section III which discusses the various feature representations via signatures for histogram-based and cluster-based representations. It also describes the node structures and the different functions associated with them. Section IV gives a detailed idea about the different aspects of the k-NN based similarity search. It is followed by Section V, where an extensive study of the behavior of the GeM-Tree is presented with different data types, varied data representations and different retrieval strategies. A brief conclusion and scope of future work are presented in Section VI.

II. EARTH MOVER’S DISTANCE

The Earth Mover’s Distance (EMD) naturally extends the concept of distance measurement between single elements to that between sets or distributions [9]. It was derived from the transportation problem viz. the Monge-Kantorovich Problem [12] which determines the minimum cost of transporting goods from a set of m sources or suppliers to a set of n destinations or demanders. Signature/feature matching can be extended to the transportation problem by defining one signature as the supplier and the other as the consumer. The cost to transfer goods between one pair of supplier and consumer is termed as the ground distance when signature or feature distributions are considered and the solution is the minimum amount of ‘work’ needed to transform one distribution to another.

The main advantages of using EMD are twofold. First, it can be used to determine the distance between variable-length feature sets, and second, it accommodates partial matches between feature distributions of data objects. The first property makes it a likely choice for our system where different feature type representations, sometimes involving variable length features, are necessary to support the different types of multimedia retrieval strategies. For example, in region-based image retrieval approaches, an image is typically represented with the features of important regions. Often, the number of distinguishing regions vary from image to image, and hence the feature distributions of the images will be of variable length. The second property of EMD is particularly beneficial when only parts of an image need to be matched. This is useful for object-based retrieval strategies where parts of an image, consisting of particular objects, need to be matched.

Let m be the number of points, $D^{K,m}$ represent the distribution in a $K \times m$ dimensional space with $K$ as the dimension of each distribution $x^m$, and $R^m$ and $R^{K \times m}$ represent the real number space in $m$ and $K \times m$ dimension, respectively. To use an EMD function, a multimedia object is represented as a signature or a finite distribution $P$ as follows.

$$P = (x_1, w_1), (x_2, w_2), \ldots, (x_m, w_m)$$

$$\equiv (X, W) \in D^{K,m}. \quad (1)$$

$$X = [x_1, x_2, \ldots, x_m] \in R^{K \times m}. \quad (2)$$

$$W = [w_1, w_2, \ldots, w_m] \in R^m. \quad (3)$$

Given two distributions $P_1 = (X, w^m_1) \in D^{K,m}$ and $P_2 = (Y, w^m_2) \in D^{K,n}$, where $P_1$ and $P_2$ are $P$, a flow between $P_1$ and $P_2$ is a matrix defined as:

$$F = (f_{ij}) \in R^{m \times n}. \quad (4)$$

The main approach is to find a flow between $P_1$ and $P_2$ that minimizes the overall cost of the work done in order to displace values between $P_1$ and $P_2$ as presented in Equation (5).

$$Work(P_1, P_2, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}. \quad (5)$$

where $d_{ij}$ is the ground distance defined in Equation (11).

Four conditions need to be satisfied by $f$ in Equation (5) as presented in Equations (6) to (9).

$$f_{ij} \geq 0; \quad 1 \leq i \leq m, 1 \leq j \leq n. \quad (6)$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{x_i}; \quad 1 \leq i \leq m. \quad (7)$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{y_j}; \quad 1 \leq j \leq n. \quad (8)$$
\[
\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = \min_{} (\sum_{i=1}^{m} w_{xi}, \sum_{j=1}^{n} w_{yj}); \quad (9)
\]

An optimal flow \( F \) is calculated using the solution technique of the transportation problem [12], and EMD is defined as given in Equation (10), where \( d_{i,j} \) is the ground distance expressed in Equation (11).

\[
EMD(P_1, P_2) = \frac{\left(\sum_1^m \sum_1^n d_{ij} f_{ij}\right)}{\left(\sum_1^m \sum_1^n f_{ij}\right)}.
\quad (10)
\]

\[
d_{i,j} = d(x_i, y_j).
\quad (11)
\]

EMD is a metric, i.e., it follows the laws of symmetry, positivity, and triangular inequality when the total weights of the distributions are equal. \( \sum_1^m w_{xi} = \sum_1^n w_{yj} \), and the ground distance \( d_{i,j} \) is a metric [9].

### III. GeM-Tree

Generalized Multimedia Tree, as the name suggests, is designed with a vision to provide a general framework to organize different multimedia data types with varied representations efficiently. As discussed in Section II, EMD is metric if the ground distance is metric and the summation of weights in each feature distribution is equal to one another. We used a metric ground distance, specifically Euclidean Distance function \( (L_2) \), and made the total weights in each feature distribution equal to make the EMD metric. The multimedia data objects are indexed in a \( R^n \) metric space, where \( n \) is the total number of features used to represent a multimedia object (an image or any video unit). The most essential step in indexing the multimedia data objects is to first represent them effectively, so that the information required during different retrieval methods are preserved. To be able to provide a common platform, there should be a common structure for the feature representations of images as well as videos. At the same time, since each media type is different and has a different information need, the common feature representation structure should be able to accommodate it. Additionally, even for similar multimedia data types, different retrieval types have a need for different data representations. To cover all the above stated needs, a novel Multimedia Data Signature is proposed. It has two main representations, viz. fixed-length and variable-length multimedia data signatures. Both the signature types are designed in a manner so that the underlying distance function (i.e., EMD) can handle them and can be a metric.

#### A. Fixed-Length Multimedia Data Signatures

For traditional content-based retrieval, where the entire multimedia object (an image or a video frame) is treated as a single class/region, the fixed length multimedia data signature is used to represent them. Both visual and audio features are used in the feature distributions. In this paper, for images, nineteen visual features consisting of color and texture information are utilized. For videos, the same features used for images are used, and additionally some combined values (average, maximum, or minimum) over a number of frames are utilized to represent a video shot. Nineteen additional features are extracted from those video frames comprising a video shot to capture the sequential relationship that exists among them. Techniques described in [13] are applied for shot boundary detection and grouping a bunch of consecutive video frames as a shot. It is worth pointing out here that any feature set (visual or audio) can be represented collectively with \( F_A \), \( F_B \) and \( F_C \) without any loss of generality. It completely depends upon the application, data types and the user preference. For example, while representing videos, if the retrieval application addresses frames as the lowest level of video units, \( F_A \) stores the low-level features of individual frames. However, if the retrieval applications do not require frame-level information, \( F_A \) can have all ‘zero’ values. In this paper, we use ‘frame’ as the lowest level of video unit. For the rest of this section, we discuss the signature parameters particularly with respect to the retrieval application and dataset used in our system.

For the sake of clarity, the feature distributions for each multimedia data object are divided into three sub-distributions as follows.

\[
F_A = \{x_1, x_2, \ldots, x_i\}, \quad (12)
\]

\[
F_B = \{y_1, y_2, \ldots, y_j\}, \quad (13)
\]

\[
F_C = \{object_id, v_{id}, s_{id}\}, \quad (14)
\]

The feature vector representing the distribution of each multimedia data object is a union of the three sub-distributions and is represented as given in Equation (15).

\[
F = \{(F_A \cup F_B \cup F_C), F_{vid}\}, \quad (15)
\]

where \( F_A \) is the color and texture features for each image and each frame in a video shot, \( x_i \) is the value of the color/texture feature normalized in a 0-1 scale. \( F_{vid} \) has the visual and audio features for a video shot. Examples of the visual features for video shots are the average percentage of the changed pixels between consecutive frames in a shot, the mean value of the frame-to-frame histogram difference in a shot, etc. [13]. For audio features, a sampling frequency of 16,000 \( H_z \) is used and the audio track is divided into clips. Each audio feature is calculated at the frame-level and synchronized with the visual features. Audio features used in this framework are divided into three basic types: volume, energy, and spectrum flux. Some audio features used to represent the framework are volume mean, RMS value of energy, mean of spectrum flux, etc. \( F_C \) captures two important information about the multimedia data: the unique identification number and the hierarchical relationship (if any). The hierarchical relationship is only meant for video data where it exists between the different video units. \( v_{id} \) stores the \( object_id \) of the video data object of which a particular frame or a shot is a part, and \( s_{id} \) stores the \( object_id \) of the shot of which a frame is a part. For images and entire video objects, both the fields are set to ‘zero’. For video shots, \( v_{id} \) is set to the \( object_id \) of the video object to which the video shot belongs; whereas the \( s_{id} \) field is set to ‘zero’.
Similarly, for video frames, \( s_{vid} \) is set to the object\(_{vid} \) of the video object, and \( s_{vid} \) is set to the object\(_{vid} \) of the video shot to which the particular video frame belongs. \( F_{wt} \) is a set of cardinality 1 as only one feature/distribution class is utilized, and the value is set to 1 for all the Data Signatures.

### B. Variable-Length Multimedia Data Signatures

For variable-length Multimedia Data Signatures, each image/video frame is modeled as a Gaussian Mixture distribution in the feature space. Each image/video frame is divided into homogeneous regions, and regions are represented by Gaussian Mixtures [14]. In our proposed framework, the RGB color space is used, but any feature space can be utilized without any loss of generality. The Expectation Maximization (EM) algorithm [15] is used to determine the maximum likelihood parameters of a mixture of \( k \) Gaussians in a 3-D feature space (RGB). EM algorithm is generally used when there is a missing data. In this case, the missing data is the region where each pixel in the image/video frame belongs. To choose \( k \) (i.e., the number of clusters/regions in each image/video cluster), a goodness of fit measure is considered. There are three popular goodness of fit measures: (i) The Akaike Information Criterion [16], (ii) The Bayes Information Criterion [17], and (iii) Minimum Description Length (MDL) [18]. Here, the Akaike Information Criterion is adopted to get the \( k \) value that will divide the image/video frames into an optimum number of regions.

To set the weight \( F_{wt} \) for each region, the fraction of image pixels that belong to the particular cluster is determined. Thus, the summation of \( F_{wt} \) for each image/video shot is always equal to one. The mean of each cluster/region is used to represent the feature vector for that particular region of the image/video frame. For the dataset we used, \( k \) ranges from 2 to 5. Thus, each region of an image/video frame is represented as \( F_{A_{cluster}}=(0.35,0.45,0.39) \), \( F_{A_{cluster}}=(0.54,0.62,0.23) \), \( F_{A_{cluster}}=(0.54,0.62,0.23) \), and \( F_{A_{cluster}}=(0.43,0.27,0.15) \). The feature distribution is thus \( F_{id}=\cup_{i=2}^{5}(F_{A_{cluster}}) \). For videos, the hierarchical relations and the weights are represented as \( F_{B}=[0,0,0,0] \), \( F_{C}=[5,0,0] \), and \( F_{wt}=[0.23,0.37,0.25,0.16] \), respectively. Figure 2 represents the different stages of the clustering method using Gaussian Mixture Models with Expectation Maximization Algorithm. This example uses \( k=5 \) (i.e., the number of clusters is 5). The top-leftmost is the original picture. The top-right figure shows the dominant colors or the cluster colors. The bottom-left figure represents the clustered image and the bottom-right depicts the cluster index. Thus, the image is represented into five clusters and the mean color values of the pixels present in each cluster forms the feature distributions of the image. It should be noted that videos are collections of frames which can be treated as images and clustered in the same way to generate the distributions for each video frame.

![Figure 1. Clustering using Gaussian Mixture Models with k=5.](image)

### C. GeM-Tree Nodes

There are two basic node types in GeM-Tree, namely the intermediate nodes storing the pointers to the subtrees, and the leaf nodes storing the pointer to the multimedia data objects stored in the database. Each node also stores a covering radius. For intermediate nodes, it is the distance between the root and the farthest child of the subtree it is pointing to. For leaf nodes, the covering radii is essentially ‘zero’. Additionally, each node has place holders for storing the high-level semantic relationship of the multimedia data objects with respect to the query object and with each other. Since there are four types of multimedia data objects handled in this framework, these nodes can be sub-categorized according to the type of multimedia data objects they are storing. Thus, we call them image\(_{intermediate} \) and image\(_{leaf} \) nodes, frame\(_{intermediate} \) and frame\(_{leaf} \) nodes, shot\(_{intermediate} \) and shot\(_{leaf} \) nodes, and video\(_{intermediate} \) and video\(_{leaf} \) nodes, respectively.

New multimedia data objects are inserted to GeM-Tree with an effort to achieve the following two criteria, namely to keep the increase of the covering radii minimum and to store similar object types close to each other. The first criterion needs to be primarily satisfied, and in case there is a tie, the nodes are picked up so that the object types of the majority objects stored in it match the object type of the incoming one. For a detailed discussion of the insertion criteria, refer to [8]. As all multidimensional tree-based index structures descended from the basic B+ Tree, the insert and search operations have a complexity of \( O(\log n) \) for \( n \) entries in the worst case [19].

### IV. Similarity Search

Though the number of feature distributions present in each multimedia data signature can vary, the number of features in each distribution is essentially the same. Since images do not have the video-related features, \( F_{B} \) is set
to all zeros. The typical representation ensures that not only the intra-multimedia object similarity is determined correctly, but also the inter-multimedia object similarity (e.g., between an image and a video shot) is calculated correctly. [8] presents a detailed explanation of how the similarity between multimedia data objects is preserved with the proposed data signatures and the underlying distance functions.

Retrieval based on content is the most popular approach of multimedia retrieval. While enabling an index structure to support such retrieval strategies, two important aspects need to be considered. First, the metric properties of the indexed space should not be violated while attempting to get the query results (if it is done, it can no longer be guaranteed that the obtained results are indeed the closest ones to the submitted query). Second, the high-level semantic relationships should be considered during retrieval (the semantic-gap issue typical to multimedia data will not provide satisfactory results otherwise). The k-NN approach like branch-and-bound [20] ensures the correctness of the search process in terms of the underlying indexed space. Thus, the main challenge in introducing similarity searches to GeM-Tree is to tailor the original k-NN search algorithm to accommodate content-based retrievals and to introduce high-level relationships while preserving the metric properties. Additionally, since it involves different multimedia data types, different retrieval scenarios need to be considered which are specific to it.

A. High-Level Semantic Relationships

A mathematical construct called Hierarchical Markov Model Mediator (HMMM) [21] is used. It is represented by an 8-tuple $\lambda = (d, S, F, A, B, \pi, O, L)$, and each element of the tuple is discussed in details in [21]. In this framework, $d$ is set to 3 and $A$ matrix is used to store the semantic relationship between multimedia data objects at each level. To determine the high-level similarity between data objects at different levels (i.e., between different multimedia data types), the hierarchy is traversed accordingly. For example, if one wishes to find out the semantic relationship between a frame and a shot, the hierarchical model is first traveled upwards to get the similarity between the shot under consideration and the shot to which the frame belongs.

B. Affinity Promotion in GeM-Tree

As discussed in [4], the affinity relationships cannot be introduced into any distance-based index structure during building a tree, in order to satisfy the metric condition of triangular inequality of the metric search space. They need to be promoted from the leaves to the intermediate nodes before each query.

The main idea behind the affinity promotion is to ascertain that there is no false dismissal and no unnecessary sub-tree traversal. The tree is basically traversed from bottom-up during the affinity promotion technique. When a query is submitted, at first the object type of the query is determined to find out if the submitted query is an image, a shot, or a video. Then the appropriate affinity matrix is loaded and the leaf nodes are examined to obtain the affinity relationship between each one of them and the query object. If the examined leaf node holds an object that is of the same type as the type of the query object, the corresponding affinity value from the affinity matrix is stored. If the object type of the examined leaf is different from the object type of the query, the HMMM model is traversed to obtain the affinity value between them as discussed in Section IV-A. Once all the leaf nodes have a particular affinity value with respect to the query object, the maximum of the affinity values among the sibling leaf nodes is determined. This becomes the affinity of the parent node to which the set of leaves belong.

The affinity is stored in the appropriate place holder according to the object type of the submitted query. This process continues till the root of the GeM-Tree is reached. It ensures that if there is at least one object with the required semantic closeness to the query object, this fact gets reflected in the affinity value stored at its parent node. Thus, false dismissals can be avoided during the similarity search. Additionally, it also ensures that if there is no child object satisfying the object type and/or the affinity matching, the corresponding parent node can be confidently pruned without any further consideration. This will save huge computation overhead during the similarity searches. A pseudo-code for the affinity promotion technique is presented in Table I.

C. k-NN Search

The k-NN algorithm for GeM-Tree supporting content-based retrieval of multimedia data is presented in Table II. It supports traditional image retrieval, inter- and intra-level video data retrieval for multiple video units, along with cross-multimedia data type retrievals. Additionally, it supports the approaches like region-based and object-based retrieval with little or no modifications of the presented algorithm. The main modification will be in the
data signature representation whereby the regions/objects should be captured efficiently. This should be followed by modifying the distance function during the similarity search. For region-based or object-based retrieval, the distance function should compute the similarity between regions of images or object boundaries. However, in this paper, we concentrate on basic content-based retrieval of images and videos along with concept-based retrieval where the cross-similarity between them is considered. At first, the affinity promotion is done as discussed in Section IV-B. For each intermediate node in the GeM-Tree, the feature similarity and the affinity value (if similar object types) with respect to the query object are checked. If both these values of the candidate node are greater than those of the nodes examined so far (called the threshold similarity and affinity values), it is stored in a priority queue of the possible nodes for future recursion. The priority queue is updated and so are the dynamic threshold distance and the affinity values. The process continues in a recursive manner. Now, if the object type of the candidate node does not match with the query object type, the hierarchy relationship is traversed upwards/downwards to find an affinity value between the two data objects as discussed in Section IV-A. If there is no available hierarchical relationship between the query object type and the object type of the candidate intermediate node and thus no available affinity relationship, the search procedure is continued depending on the feature-level similarity only. For the leaf nodes, the same steps are undertaken with the difference of adding the candidate objects to the result set without further recursion if they satisfy both the low-level and high-level similarity requirements. Thus the k-NN algorithm of GeM-Tree is flexible and can accommodate different kinds of video unit classifications. In the k-NN algorithm presented in Table II, a shot is used as the lowest unit of a video.

The multimedia data signature is designed so as to keep the image and video related features separately and easily accessible. This helps in using the GeM-Tree as a dedicated index structure for only images or videos. For video accesses, using only the \( F_B \) portion during the distance computation will return only similar video objects. For image accesses, considering only the \( F_A \) portion of the signature and pruning off any candidate having a non-zero \( F_B \) will provide the required result.

V. EMPIRICAL STUDY

An extensive study of the computation cost in terms of distance calculations and the relevance of query results in terms of accuracy is performed for Gem-Tree for different multimedia data types, viz. images and videos. Three different types of queries are executed, namely queries involving only images, queries involving only videos, and queries involving both images and videos. The results obtained from GeM-Tree are compared with a distance-based index structure for only images \([4]\) (labeled as I in Tables III to V), a distance-based index structure for only videos \([5]\) (labeled as II in Tables III to V), and with a sequential search approach (labeled as III in Tables III to V). Since the sequential search method essentially computes the distances between every pair of multimedia data objects present in the system, it has the highest accuracy and is used as a bench mark to determine the relevance of the query results obtained from the frameworks with index structures. Its distance computation is also presented to show that the high accuracy comes at the cost of a high computation overhead. Also, no matter what object is being searched, the sequential search goes through the entire dataset to generate the results. Thus, the number of distance computations is always the same for method III.

We used about 1000 images from the COREL database and about 5 videos with a couple of hours of duration. We performed the experiments on two different Data Signature types. Table III and Table IV use a fixed-length feature distribution where each image/video frame is considered as a single class/region. Nineteen color and texture features are extracted from them and the \( F_A \) part of the Signature is formed. \( F_B \) is formed from the nineteen video related features, and \( F_C \) captures the hierarchical relationships among the video shots. The results for 10 queries of each category are averaged. The first query type consists of querying the database, consisting of mixed type multimedia objects, for only images. The second type consists of querying the same database for only videos, and the third type comprises of cross-queries, where any multimedia object (images and videos) similar to a submitted query needs to be retrieved. To indicate

<table>
<thead>
<tr>
<th>TABLE II. k-NN Search Algorithm for Gem-Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>k-NN GeneralSearch</strong> ( (Q, N, k) ) {</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

© 2010 ACADEMY PUBLISHER
that a particular index structure is incapable of handling a particular query type, the corresponding location in the table is marked with an ‘X’. It can be seen that the computation cost for GeM-Tree in all the three types of queries is slightly higher than those of index structures for only images (I) and only videos (II). This is because GeM-Tree indexes more types of multimedia data objects as the underlying database consists of both images as well as videos. The accuracy of GeM-Tree was slightly lower than those of method I and method II because of the same reason. Since the underlying database has both media types, while retrieving only one type of media, the search parameters are made stringent and only nodes satisfying the object type of the query are considered. These nodes might have object types, matching the query object, as their children. This is possible as it has been pointed out before that similarity of the low-level feature content is given a higher priority over object types while organizing the multimedia data objects in GeM-Tree. Thus, there can be some false dismissal which might affect the overall accuracy to a little extent. Such a problem can be easily overcome by making the query parameters more loose whereby a node having a different object type than the query object should be considered if it has at least one child with the same object type as the query. However, this might result in a slight increase of the computation cost.

However, it should be noted that though GeM-Tree has slightly lower performance in comparison to dedicated image-only and video-only index structures, it has the added capability to answer concept-based queries involving both images and videos.

### TABLE III.
**DISTANCE COMPUTATIONS DURING QUERYING THE INDEX TREES FOR FIXED-LENGTH FEATURE DISTRIBUTIONS**

<table>
<thead>
<tr>
<th>Query</th>
<th>Distance Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Image</td>
<td>GeM 98, 80</td>
</tr>
<tr>
<td>Only Video</td>
<td>90% X 91%</td>
</tr>
<tr>
<td>Cross Query</td>
<td>80% X X</td>
</tr>
</tbody>
</table>

### TABLE IV.
**ACCURACY FOR FIXED-LENGTH FEATURE DISTRIBUTION**

<table>
<thead>
<tr>
<th>Query</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Image</td>
<td>GeM I II III</td>
</tr>
<tr>
<td>Only Video</td>
<td>90% X 91%</td>
</tr>
<tr>
<td>Cross Query</td>
<td>80% X X</td>
</tr>
</tbody>
</table>

### TABLE V.
**DISTANCE COMPUTATIONS DURING INDEX TREE FORMATIONS FOR VARIABLE-LENGTH FEATURE DISTRIBUTIONS**

<table>
<thead>
<tr>
<th>Data</th>
<th>Distance Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images</td>
<td>GeM 145 X X</td>
</tr>
<tr>
<td>Only Video</td>
<td>240 X X</td>
</tr>
<tr>
<td>Both</td>
<td>960 X X</td>
</tr>
</tbody>
</table>

The second set of experiments were performed on a variable-length feature distribution. As discussed in Section III-B, each image/video frame is represented as clustered regions using Gaussian Mixture Models and Expectation Maximization techniques. We conducted a preliminary test on the HSI and the RGB color spaces and used various cluster sizes ranging from 2 to 5. Using the Akaike Information Criterion to determine the goodness of fit, we found that the optimum cluster size for most of the image/video frames was 4. Figure 2 presents the relationship between the distance computations and the number of clusters during constructing the GeM-Tree. It can be seen that the computation overhead increases with the increase of the number of clusters, which is obvious as with the increase of the clusters, the number of instances of each feature corresponding to each multimedia data object also increases. Thus, the total number of feature distributions representing the entire data object increases and the number of distance computations, necessary during the tree formation, increases as well. Hence, a careful choice of the number of clusters should be made. If the number of clusters is large, though the subsequent similarity measurements will be more precise, it is at the cost of an increased computational overhead. Similarly, if in order to reduce the computation overhead, too few clusters are chosen, the data object will not be represented properly. This will further lead to poor relevance of query results.

Table V presents the comparison of the computation cost for variable-length data signatures between the different index structures. It uses about 500 multimedia objects consisting of images and videos. Among them, about 200 are images and 300 are video frames. It can be observed that both index types I and II are incapable of handling variable length feature distributions (indicated by ‘X’) as they do not use EMD as the distance function. Thus, compiling observations from Table III, IV and V, it can be concluded that GeM-Tree has added capabilities over dedicated multimedia index structures with a comparable computation cost. It should be also pointed out that Table V represents the performance of GeM-Tree during the tree formation stage and thus the results presented should
not be compared with Table III and IV, which represent the performance during the query stage and uses different datasets and representations.

VI. CONCLUSION

In this paper, a common and flexible framework for organizing multimedia data is presented. It uses a distance-based index structure, GeM-Tree, which uses EMD as the distance function. A novel multimedia data signature is proposed which is capable of preserving the pertinent information for different multimedia data objects like images and videos with a single extensible structure. The proposed framework is capable of handling similarity queries involving a single multimedia data type like images/videos or mixed concept-level queries. High-level semantic relationships among the multimedia data objects are efficiently embedded into the k-NN search without violating the underlying indexed metric space to achieve query results with higher relevance. Experimental results demonstrate satisfactory performance both in terms of number of distance computations and relevance of query results.

REFERENCES


Kasturi Chatterjee received her Master’s degree from the School of Computing and Information Sciences (SCIS), Florida International University (FIU), Miami, in December 2006. She is currently pursuing her Ph.D. in Computer Science from FIU. Her research interests include multimedia database management systems, multimedia information retrieval and multimedia data mining.

Shu-Ching Chen is a Full Professor in the School of Computing and Information Sciences (SCIS), Florida International University (FIU), Miami since August 2009. Prior to that, he was an Asstistant/Associate Professor in SCIS at FIU from 1999. He received Master’s degrees in Computer Science, Electrical Engineering, and Civil Engineering in 1992, 1995, and 1996, respectively, and the Ph.D. degree in Electrical and Computer Engineering in 1998, all from Purdue University, West Lafayette, IN, USA. His main research interests include distributed multimedia database management systems and multimedia data mining.