Hierarchical Affinity Hybrid Tree: A Multidimensional Index Structure to Organize Videos and Support Content-Based Retrievals

Kasturi Chatterjee and Shu-Ching Chen
Distributed Multimedia Information Systems Laboratory
School of Computing and Information Sciences
Florida International University
Miami, FL 33199, USA
{kchat001, chens}@cs.fiu.edu

Abstract

Multimedia data, especially videos, have gained enormous popularity in the recent years. Data management techniques for traditional text-based data are inadequate to handle multimedia data efficiently due to their atypical characteristics. Thus, to have a robust data management framework for complex multimedia data like videos, comparable in efficiency and capability to the traditional data management approaches, components like multimedia data storage, index, and query engines need to be developed with dedicated abilities to handle the characteristics of multimedia data like multidimensional representation and semantic gap. In this paper, we investigate the design of the second component, i.e., a multimedia index, and propose a novel tree-based multidimensional hierarchical index structure called Hierarchical Affinity Hybrid-Tree (HAH-Tree) which addresses the critical issues of multidimensionality and semantic gap. The index structure accommodates different levels of video relationships during Content-Based Video Retrieval (CBVR) by utilizing a probabilistic approach called the Hierarchical Markov Model Mediator (HMMM), which is also responsible for managing the high-level semantic content of the video components. In addition, a computationally efficient k-Nearest Neighbor (k-NN) algorithm is proposed, which supports CBVR for different video units with a high precision level.

1. Introduction

Multimedia data, like videos, have gained popularity in the recent years due to the proliferation of internet technology and availability of cheap storage. The atypical nature of multimedia data, viz. its high dimensionality, the varied semantic interpretation and the gap between the low-level features and high-level semantic contents, necessitate dedicated research to be able to efficiently manage them. Traditional database management systems are not capable of handling multimedia data efficiently. Thus, a robust data management framework for complex multimedia data like videos requires components like multimedia data storage, index, and query engines to be developed which should be comparable in efficiency and capability to the traditional data management approaches, and should have dedicated abilities to handle multidimensional representation and semantic gap. Indexing is one of the pivotal issues in designing efficient data management frameworks as it is basically the bridge between the storage system and the query engine, which accesses the storage system to provide query results. Therefore, an index structure should be able to handle the characteristics of both the storage system as well as the retrieval requirements to be submitted to the query engine. For multimedia data, the storage system should accommodate the multidimensional representation of the multiple features of the multimedia data, and the retrieval system, comprising of the query engine, should be able to handle the issue of high level relationship and semantic gap associated with multimedia data. Hence, an efficient index structure for complex multimedia data like videos should have the capability to handle the multi-dimensional aspect of the data and should be able to efficiently bridge the semantic gap as well.

In the past, the concept of video indexing mainly dealt with the process of classifying the video content and assigning content-based labels to them for the ease and precision of retrieval processes. As pointed out in [1], three main issues arise while classifying the video content viz. granularity, modality, and type. There are different video indexing techniques like [2][3][4] etc. from the traditional video classification point of view. For example, [2] tends to index videos based on single modality; whereas [3][5][6] use a more advanced multimodal approach to index the videos. [4] proposes another content-based video indexing system which achieves the purpose of automatic management of the video data by syntactic and semantic features. Other similar video techniques were proposed in [7][8] etc. where concepts like virtual image and Dublin core metadata [7] were used and statistical frameworks [8] were engaged for modeling and segmenting video content into coherent space-time segments.

However, none of the above techniques attempted to address the issue of indexing the video data from the true database point of view. They classify the video data into units and design a way to identify useful information from them, but internally they need to perform exhaustive search of the entire database to locate the video objects of interest. This increases the computation overhead and has increasing negative effects on the overall retrieval performance, especially for large video retrieval systems. To develop a robust multimedia database management system, designing an index structure, just as efficient and useful as index structures like kDB-Tree [9], R-Tree [10] etc., is crucial. A set-based nearest neighbor approach applied on a multidimensional index structure, to index and retrieve videos based on their feature information, was proposed in [11]. Though [11] attempts to enable index structures like SR-Tree [12] to support multimedia object indexing, it has a major drawback. That is, the nearest-neighbor algorithm proposed by [11] does not consider the high-level semantic interpretation of multimedia objects, which might have huge degradation in the precision of the query results due to the semantic gap, an inherent characteristic of multimedia data.

To address the issue of efficiently managing video data, we propose a novel hierarchical tree-based multidimensional index structure, called Hierarchical Affinity Hybrid-Tree (HAH-Tree) which indexes the data, in a multidimensional space, based on feature-level information. We further propose a computationally-economic nearest-neighbor algorithm that enables content-based video retrieval by considering both the feature-level and semantic-level similarity. Additionally, the framework of HAH-Tree supports various levels of video-unit similarity search and retrieval like frame-level similarity search, shot-level similarity search, and entire video-level similarity search. To define and accommodate the high-level similarity among different levels of video objects and to bridge the gap between the feature and semantic information, we utilize the HMMM framework [13] and embed it seamlessly within the k-NN similarity search.

The rest of the paper is organized as follows: Section 2 describes the overall structure of HAH-tree. In Section 3, we briefly discuss video units, video features, and the HMMM model. It is followed by Section 4, which presents the k-NN algorithm facilitating the similarity search. Section 5 presents the experimental results. In Section 6, a brief conclusion and the scope of future work are given.

2. Hierarchical Affinity Hybrid Tree

Hierarchical Affinity Hybrid-Tree is an elaborate extension of the basic framework of Affinity Hybrid-Tree [14][15] which is an index structure developed to manage images efficiently. Videos are more complex to handle than images because they carry more information both at feature-level as well as at semantic level. Each video can be considered as an ensemble of a large number of images, called *frames*, which might carry a number of semantic information or events in them. For example, a piece of soccer video might have a goal event and a foul event among its frames/shots, either of which might be of interest to the user. Additionally, a video can be represented as different units like frames, shots, concepts etc, which is discussed in Section 3. These special characteristics of video data made the existing AH-Tree inadequate to handle it efficiently.

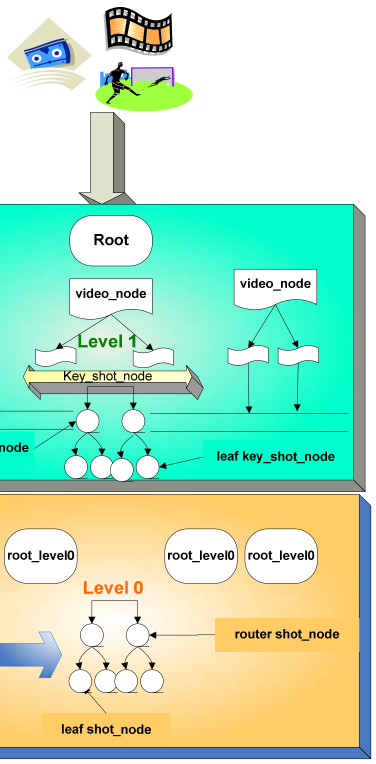

Figure 1: HAH-Tree Structure

As depicted in Figure 1, HAH-Tree is basically a two-level index structure, where Level 0 indexes the shots of the videos and Level 1 indexes the information about the concepts/events associated with individual videos. Hence, it is called *hierarchical*. One of the major differences of HAH-Tree from its predecessor AH-Tree, besides the hierarchical structure, is the type of indexing methodologies utilized. Whereas AH-Tree utilizes both feature-based and distance-based index structures within its framework, HAH-Tree utilizes only a distance-based index structure. The reason for excluding the feature-based index structure in HAH-Tree is that the feature-level index structure was utilized in AH-Tree mainly as a filtering mechanism to reduce the number of distance computations. However Level 1 in HAH-Tree already performs the required filtering process by building the index trees for the lower-level based on the conceptual information that identifies candidate video shots to a submitted query.

The top level or level 1 indexes the video-level information and stores the features of the *key shot*, which identifies a video, in a multidimensional tree index, called the *video_level_tree*. Generally, the *key shot* is the first shot of a group of shots identifying an event in a video. There can be multiple *key shots* in a video if there are multiple events associated with it. On the other hand, if no event has been identified in a video yet, the first shot of the video is set as the *key shot* by default. The lower-level or level 0, indexes the *shot level* information associated with each video event, comprising of multiple shots. Level 0 can be considered as a collection of several multidimensional index trees, one for each video/video event built with the shot nodes. The root of each such a *shot_level_tree* tree is linked to each video via the key shot of the video_level_tree. Another important characteristics of HAH-Tree is that each node is linked to its siblings, which ensures across-video or across-shot traversal during the retrieval process.

HAH-Tree has the following eight types of nodes:

- *root*: This node is the root of the HAH-Tree.
- *video_node*: This node stores the pointer to the object ids of the videos.
- *key_shot_node*: This node stores the information of the key shot for each video.
- *router_key_shot_node, leaf_key_shot_node*: These nodes store the feature-level information of the key_shots identifying the video and form the *video_level_tree*.
- *root_level0*: This node is the root of the individual tree-structures for each shot.
- *router_shot_node*, *leaf_shot_node*: These nodes store the feature-level information of each shot for each video shot and form the *shot_level_tree*.

Each router node of the HAH-Tree (both the *router_key_shot_node* as well as the *router_shot_node*) has an associated pointer which references the root of a subtree. All objects in the subtree should be within a specified radius *r* from the routing object. For the leaf nodes (both the *leaf_key_shot_node* and the *leaf_shot_node*), there are no associated covering radii and they store the pointer to the root of the *shot_level_tree* and the object id of the video object, respectively.

HAH-Tree is a balanced tree structure and is dynamic in nature. Insertion and deletion of a new video object can be achieved without re-shuffling the entire tree structure. To insert a new video object, one needs to traverse the HAH-Tree recursively to find the most suitable location to insert a leaf node which can accommodate it. The most suitable subtree, where a new leaf node can be inserted, is generally identified as one which will not increase its covering radius. If no such candidate subtree can be identified, the goal is to choose one, which on inserting a new leaf node pointing to a video object, will have the minimum increase in its covering radius. When a leaf node storing the pointer to the video object is deleted, the covering radius of the corresponding subtree should be updated and the pointer to the leaf node is set to null. The entire tree structure will not be updated for each deletion, but is updated after a certain number of deletion procedures for efficiency and optimization.

3. Video Representation

The novelty of HAH-Tree is the approach which seamlessly integrates and includes both the low-level features and high-level semantic interpretations of videos in its index and retrieval framework. Thus, it is imperative to understand the techniques which capture the required information from videos. It should be pointed out here that HAH-Tree is a very flexible and dynamic structure and should be able to accommodate different representations of low-level and high-level video information without imposing considerable overhead.

3.1. Video unit classification

Temporal segmentation of a video sequence into meaningful units is called video unit classification. There are various levels of video units that have been proposed viz. shot level, frame level, scene level, and clip level. Among them, shots are the most self-contained and well defined units. A shot-based approach categorizes a video sequence into a collection of frames, where each collection represents a continuous camera action in time and space while sharing a close high-level semantic as well as low-level feature similarity. In this paper, we used video shots as the lowest conceptual unit of videos. Video shot detection is mainly performed by adopting the three-level filtering architecture viz. pixel-histogram comparison, segmentation map comparison, and object tracking as discussed in [16]. Each video shot consists of a number of temporally related video frames, one of which called the key frame, serves as a representative of the shot. For the purpose of ease, in this work, we identified the first frame of each shot as the key frame, but other techniques can be used as well like selecting the frame which best describes the overall concept of the shot. The average of the low-level features of all the frames comprising a shot is used to represent each shot's feature vector.

3.2. Low-level features

There are two main approaches towards extracting the low-level features/visual descriptors from videos viz. unimodal and multimodal. The unimodal approach utilizes the features of a single modality such as visual, audio or textual; whereas the multimodal approach uses more than one modality for representation. In this work, we utilized multimodal features (visual and audio) as proposed in [17] for each shot. Some important shot-level visual feature descriptors utilized for indexing purposes in this paper are pixel change, histogram change, average volume, average energy, flux, etc.

3.3. High-level semantic video interpretation

In order to capture and utilize the high-level relationship among the different video units and bridge the gap between the low-level features and high-level semantic concepts attached to each video unit, a mathematical construct, called Hierarchical Markov Model Mediator (HMMM) [13] is used. It is represented by an 8-tuple $\lambda = (d, S, F, A, B, \Pi, O, L)$, where each element of the tuple is discussed in details in [13]. The element d represents the number of levels in an HMMM and the purpose and representation of the other elements vary within the level under consideration. In this research, we set d as 2 and are mainly concerned with the following three elements of the tuple viz. F, A, and B during the similarity searches. F represents the set of distinct features in level 0 and semantic concepts in level 1, A represents the affinity matrix which denotes the similarity measurement between video units as perceived by the users and collected over time, and B represents the low-level feature information for each frame at level 0 and concept matrices at level 1. The matrices F, A, and B are constantly updated, for each iteration, through a learning process by utilizing users' feedback.

4. Similarity Search

Similarity searches for video data are mainly based on two different similarity criteria viz. low-level feature similarity and high-level semantic or conceptual similarity. As discussed earlier, HAH-Tree is a distance-based index structure where multimedia data (here videos) are indexed based on a distance function like Euclidean or Manhattan in a metric space. When a query in the form of a video shot or a complete video is submitted, the k-NN search algorithm traverses the HAH-Tree and produces k most similar video objects to the user. During querying the HAH-Tree, the proposed k-NN search algorithm considers both the distance or (dis)similarity between the indexed nodes *(video_node, key_shot_node*, etc.) and the query object (also represented as feature vectors with the same data structure as the index tree nodes), as well as the high-level semantic relationships among them. A threshold value (affinity), specifying the minimum high-level similarity expected in the query result, is supplied with the query. This value is utilized to further prune the candidate nodes which have passed the distance criterion or the low-level similarity condition. It should be mentioned here that the k-NN search algorithm for HAH-Tree can handle different video units as queries. For example, HAH-Tree can be queried in frame-level, shot-level, or entire video-level. It totally depends on the video units chosen by the users to classify the videos. Frame-level queries may be issued to find frames similar to the submitted frame from within the same video or across multiple videos. This is useful when a video is large in size and users may be interested to find similar frames from within the video. Similarly, shot-level and video-level queries can be issued with the same efficiency. The k-NN algorithm presented in Table 1 discusses the video-level query processing. The frame-level and shot-level query processing are merely subsets of the presented algorithm and can be easily reproduced from it.

The k-NN algorithm, supporting CBVR, starts with extracting features from the key shots representing the videos and the frames constituting a shot of the submitted query object, and represents them as multidimensional feature vectors. Then, depending upon the video unit of the submitted query, it proceeds to the corresponding portion of the retrieval algorithm. If an entire video is submitted and is associated with a concept, at first the pointers of the videos with similar concepts are stored in a queue by querying the *video_nodes*. Next, the *video_level_tree* in Level 1 of HAH-Tree is searched for similar *key_shot_nodes* based on both the low-level feature similarity and high-level semantic closeness (obtained from the A matrix for level 1 of HMMM). d_{kKey} is the dynamic radius which stores the radius of the current kth nearest neighbor. If the examined *key_shot_node* satisfies both the conditions, it qualifies as a candidate node and is pushed into the priority queue. The value of d_{kKey} is also updated. Once the candidate key shots or in other words, the candidate video shots are short-listed, the same pruning criteria, applied for key shots, are used at the *shot_level_tree*. d_{kShot} is also dynamic in this case and the high-level semantic closeness is obtained from $A_{(ShotLevel0,ShotQuery)}$. Once the priority queue associated with the *shot_level_tree* is exhausted, the result set consisting of shots is sorted based on the temporal information (since the result set obtained is sorted based on the distance or (dis)similarity measurement). The users' feedback is collected for each iteration and the affinity matrix at each level is updated accordingly. This ensures that a constant learning loop is executed to improve the high-level semantic relationship captured metric.

Table 1: k-NN search algorithm supporting CBVR

```
1.  Extract the key shot ($key_{query}$) and the low-level feature descriptor vector ($B_{query}$) from the query video shot ($V_{query}$).
2.  For each video_node in HAH-Tree
3.      Obtain the key_shot_node.
4.      For each key_shot_node in video queue
5.          If ((d(key_shot_node, $key_{query}$) $\leq d_{kKey}$) &&
               ($A_{(KeyShotLevel1,KeyQuery)} \geq$ affinity ))
6.              Add key_shot_node to the key shot queue.
7.          endif
8.      endfor
9.  endfor
10. For each candidate key shot in key shot queue
    // search the corresponding shot-level tree at level 0
11.     If ((d(shot_node, $shot_{query}$) $\leq d_{kShot}$) &&
            ($A_{(ShotLevel0,ShotQuery)} \geq$ affinity ))
12.         Add shot_node to the shot priority queue.
13.     endif
14. endfor
15. Sort the shot priority queue for each candidate video with respect to the temporal information and generate the result set.
16. Collect users' feedback for each result set and update the A matrix at each level.
```

5. Experiments

In our experiments, 10 soccer videos were collected from different sources with total time duration of almost 2 hours. For each video, shot boundary detection was performed utilizing the concepts presented in [16]. For each shot, the key frame is set as the first frame and 20 multimodal features (as discussed in Section 3.2) are extracted. Ten queries comprising of video-level and shot-level were issued. Since, to the best of our knowledge, there is no comparable video indexing framework like HAH-Tree, we could not compare the performance with any other tree-based video indexing strategy. However, we compared our system with the traditional exhaustive video retrieval strategy which does not have any underlying index structure from the storage point of view and depends only on the video classification technique to provide search result. Essentially, the exhaustive search traverses the entire video shot-by-shot to provide the query results. The results presented in Table 2 and Figure 2 demonstrate tremendous improvement in computation time and the number of I/Os for HAH-Tree over the naïve system. The accuracy, a very subjective indicator for video retrieval, of HAH-Tree was satisfactory with an average value of 70%-80%. Though the exhaustive search framework may have better retrieval accuracy than that of the HAH-Tree, it is achieved at the cost of very high computation overhead. Please also note that due to the small video sizes used in the experiments, the improvement in the computation overhead for HAH-Tree may not seem very drastic as compared to the exhaustive search approach. However, for videos which consist of several thousands of shots, the improvement will be dramatic. The sample videos were chosen to be considerably smaller in sizes so that we can traverse the videos manually and check the accuracy.

Table 2: Experimental Results for HAH-Tree and Naive Video Retrieval Framework

Data Set	# of distance computations		# of I/Os	
	HAH-Tree	Naïve	HAH-Tree	Naïve
1	106	137	50	64
2	94	200	39	83
3	15	197	36	472
4	124	168	51	69
5	113	126	50	56

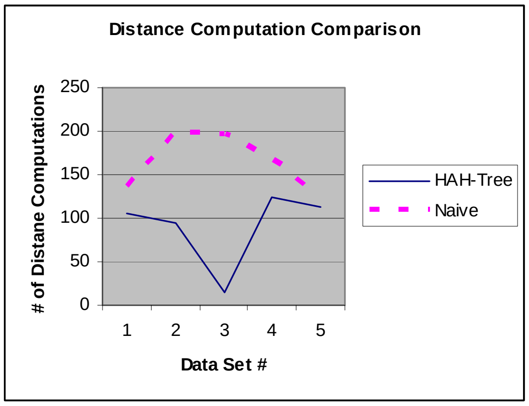

Figure 2: Graph Comparing the Distance Computations

6. Conclusion and Future Work

In this paper, a novel tree-based multidimensional index structure, called Hierarchical Affinity Hybrid Tree, was proposed to manage video data efficiently. We further proposed an innovative k-NN search algorithm for HAH-Tree which supports CBVR by amalgamating low-level feature similarity and high-level semantic closeness among videos. The proposed k-NN search algorithm is a very flexible structure and is capable of accommodating queries for different video units like frames, shots, and entire videos. The experimental results demonstrate encouraging outcome in terms of low computation overhead and a satisfactory accuracy for queries. We can conclude that HAH-Tree is a useful framework and has potential for future investigation and improvement.

As a part of our future work, we plan to include temporal relationship and event information in the index structure to improve its performance and broaden its domain.

REFERENCES

[1] C. Snoek and M. Worring, "Multimodal video indexing: A review of the state-of-the-art," *Multimedia Tools and Applications*, 25(1), pp. 5--35, 2005.

[2] C. Colombo, A. Del Bimbo, and P. Pala, "Semantics in visual information retrieval," *IEEE Multimedia*, 7(1), pp. 60-67, 2000.

[3] A.A. Alatan, A.N. Akansu, and W. Wolf, "Multi-modal dialogue scene detection using hidden markov models for content-based multimedia indexing," *Multimedia Tools and Applications*, 14(2), pp. 137-15, 2001.

[4] B. Gunsel, A.M. Ferman, and A.M. Tekalp, "Video indexing through integration of syntactic and semantic features," in *Proceedings of 3rd IEEE Workshop on Applications of Computer Visions*, Sarasota, USA, pp. 90-95, 1996.

[5] N. Babaguchi, Y. Kawai, and T. Kitahashi, "Event based indexing of broadcasted sports video by intermodal collaboration," *IEEE Transactions on Multimedia*, 4(1), pp. 68-75, 2002.

[6] S. Eickeler and S. Muller, "Content-based video indexing of TV broadcast news using hidden markov models," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, USA, pp. 2997-3000, 1999.

[7] P. Palma, L. Petraglia, and G. Petraglia, "The virtual image in streaming video indexing," in *Proceedings of International Conference on Dublin Core and Metadata for e-Communities*, Florence, Italy, pp. 97-103, 2002.

[8] H. Greenspan, J. Golberger and A. Mayer, "Probabilistic space-time video modeling via piecewise GMM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3), pp. 384-396, 2004.

[9] J. Robinson, "The k-d-b-tree: A search structure for large multidimensional dynamic indexes," in *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, Ann Arbor, United States, pp. 10–18, 1981.

[10] A. Guttman, "R-trees: A dynamic index structure for spatial Searching," in *Proceedings of the 1984 ACM SIGMOD InternationalConference on Management of Data*, Boston, Unites States, pp. 47–57, 1984.

[11] N. Katayama and S. Satoh, "Application of multidimensional indexing methods to massive processing of multimedia information," *Systems and Computers in Japan*, 31(13), pp. 31-41, 2000.

[12] N. Katayama and S. Satoh, "The SR-tree: an index structure for high-dimensional nearest-neighbor queries," in *Proceedings of 1997 ACM SIGMOD*, Tucson, pp. 369-380, 1997.

[13] S.-C. Chen, N. Zhao, and M.-L. Shyu, "Modeling Semantic Concepts and User Preferences in Content-Based Video Retrieval," *International Journal of Semantic Computing (IJSC)*, 1(3), pp. 377-402, 2007.

[14] K. Chatterjee and S.-C. Chen, "A Novel Indexing and Access Mechanism using Affinity Hybrid Tree for Content-Based Image Retrieval in Multimedia Databases," *International Journal of Semantic Computing (IJSC)*, 1(2), pp. 147-170, 2007.

[15] K. Chatterjee and S.-C. Chen, "Affinity Hybrid Tree: An Indexing Technique for Content-Based Image Retrieval in Multimedia Databases," in *Proceedings of the IEEE International Symposium on Multimedia (ISM2006)*, San Diego, CA, USA, pp. 47-54, 2006.

[16] S.-C. Chen, M.-L. Shyu, and C. Zhang, ''Innovative Shot Boundary Detection for Video Indexing,'' *Video Data Management and Information Retrieval*, Edited by S. Deb, Hershey, PA, USA: Idea Group Publishing, pp. 217-236, 2005.

[17] S.-C. Chen, M.-L. Shyu, C. Zhang, L. Luo, and M. Chen, ''Detection of Soccer Goal Shots Using Multimedia Features and Classification Rules,'' in *Proceedings of the 4th International Workshop on Multimedia Data Mining*, Washington, DC, USA, pp. 36-44, 2003.