PROCESSING CONCEPT QUERIES WITH OBJECT MOTIONS IN VIDEO DATABASES

Chia-Han Lin and Arbee L. P. Chen[*]

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: alpchen@cs.nthu.edu.tw

ABSTRACT

This paper proposes an indexing and query processing approach for content-based video retrieval. The concept hierarchy is used to provide high-level concepts for users to specify queries. The index buckets and suffix trees are proposed to index the static attributes and motions of video objects. A flexible query processing strategy is designed based on the index structure. Also, strategies for processing approximate queries and queries involving object motions are considered.

1. INTRODUCTION

Due to the great progress of computer technologies and the mature development of Internet, more and more multimedia data are distributed in the Internet. How to efficiently retrieve the desired multimedia data has become an important issue. Among various media, video contains the richest content, including image information, audio information and temporal information of image sequences. In a video database system, indexing and query processing should be considered for the content-based video retrieval. The research issues on video indexing include the representation of video content and the index structure used for efficient query processing. The research issues on query processing include the query specification and the query processing for retrieving the desired video clips.

Liu and Chen[8] propose a data structure named *3D-Llist* which is used to represent the spatial and temporal relationships of the video objects. In Dagtas et al.[2] the motion of the object is represented by *trail*, which is the picture of trajectory. The models for video indexing are proposed in [3]. Several similarity measures are proposed for retrieving videos with similar motion characteristics. The spatial translation invariance and the temporal scale invariance are also considered in the retrieving process. In our previous work[6], a video data model is proposed to represent the content of video data. The values of the attributes of video objects, which are color, object type, location and size, are recorded to represent the static properties of the video objects. Moreover, the motion property of single video objects and multiple video objects is automatically derived from the object trajectories.

In addition to the representation of video content, the consideration of query specification is also important in content-based video retrieval. The user may specify queries by a query language or query interface. Liu and Chen[7] propose a multimedia query language named Media SQL. A corresponding graphical query interface is implemented. Aghbari, et al.[1] propose a query interface for the user to specify the changes of frame contents, named *content trajectory*. The content trajectory is then used to retrieve relevant video clips. Kuo and Chen[5] propose a content-based video query language CVQL. The temporal and spatial relationships of video objects are used as query predicates. Moreover, a macro mechanism is employed to simplify the query specification. The query processing strategy of CVQL is also proposed. A prototype content-based video database system, including the modules of shot change detection, object detection and trajectory detection is described in Kuo and Chen[4]. Furthermore, a graphic query interface is designed for the user to specify queries. In [6], A query language V-SQL based on the proposed video data model is also presented. The static and motion properties can be flexibly specified using V-SQL. Based on the query language, a graphical query interface is implemented to help the user specify queries.

In this paper, a concept hierarchy is used to represent different levels of concepts for each attribute in the data model. An index structure based on the concept hierarchy is presented. Moreover, suffix tree is used as the index structure for motions. Query processing for various types of queries is also discussed.

The organization of this paper is as follows. The video data model is presented in Section 2. Section 3 presents the index structure for video objects. The query processing is presented in Section 4. Finally, Section 5 concludes this work.

2. VIDEO DATA MODEL

The video data model proposed in [6] is briefly presented in the following. A video scene is considered as the basic unit for video representation. The motions and other properties of the objects in the scene are used to represent the content of the scene. A scene can be expressed as a five-tuple (*sid, vid, VO, SOM, IOM*) where

sid : scene ID
vid : video ID of the video containing the scene

[*] Corresponding author.

VO : set of objects in the scene
SOM : set of single-object motions
IOM : set of inter-object motions

An object can be expressed as a seven-tuple (*oid, sid, T, X, Y, S, C*) where

oid : object ID
sid : scene ID of the scene containing the object
T : type of the object
X : the x-axis value of the location in the frame where the object first appears
Y : the y-axis value of the location in the frame where the object first appears
S : size of the object
C : dominant color of the object

There are three properties of single-object motions: *Velocity*, *Acceleration* and *Orientation*. Velocity can have four values: *high speed, medium speed, low speed* and *zero speed*. Similarly, Acceleration can have three values: *positive, zero* and *negative*, and Orientation can have eight values: *east, northeast, north, northwest, west, southwest, south* and *southeast*. Based on each property, a motion can be represented as a sequence of the property values. For example, based on Velocity, a motion can be represented as "high speed, medium speed, low speed, medium speed." There is only one property of the inter-object motions: *Distance*. Distance can have four values: *increasing*, *decreasing*, *constant* and *zero* denoting different situations of the distance between two objects. The single-object motion for each object and the inter-object motion for each object pair in the scene are used to express the motions in the scene.

3. INDEX STRUCTURE

In the proposed model, there are two kinds of properties of video objects by which users can specify queries. One is the static attributes and the other is the motions. In this section, we propose two index structures for these properties. A 5-D Grid index structure is used for indexing the static attributes and the suffix tree index structure for the motions.

3.1 Concept Hierarchy

The concept hierarchy is used to represent different levels of concepts for an attribute. The lowest level of the concept hierarchy is called the *elementary element* of the concept hierarchy. The other levels of the concept hierarchy are called *internal elements*. The *concept space* of an element is a set of values in the domain of the attribute. The concept space of an elementary element contains the value of the attribute. The concept space of an internal element is the union of its children's concept spaces.

Figure 1 shows an example of a concept hierarchy. In this example, each axis of location has eight values to represent the coordinates of video objects. The concept space of element 12 is {0, 1, 2, 3}, which is the union of its children element 8 ({0, 1}) and element 9({2, 3}). The concept space of the root of the concept hierarchy, i.e., element 14, contains all the values in the domain of the attribute, i.e., {0, 1, 2, 3, 4, 5, 6, 7}.

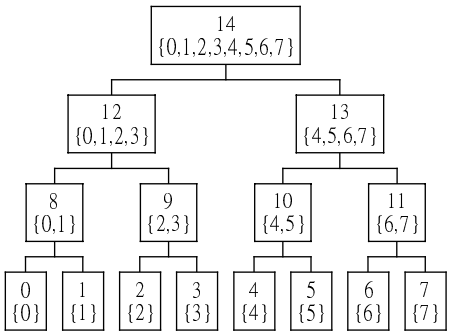

Figure 1. The concept hierarchy for X and Y.

3.2 Static Attributes

The index structure for the static attributes is constructed using the elements in the concept hierarchies. As described in Section 2, each video object is associated with five attributes. Each attribute is associated with a set of elementary elements. Video objects will be mapped into a five dimensional index space according to their attribute values.

Denote the attribute set of video objects as $A=\{A_1, A_2, A_3, A_4, A_5\}$. The *element set* of a concept hierarchy is defined as the set of all the elements in the concept hierarchy. Each attribute A_i is therefore associated with an element set, denoted E_i. An *index bucket* is denoted as $Bucket(b_1, b_2, b_3, b_4, b_5)$, where $b_i \in E_i$, i=1,...,5. Each index bucket contains objects whose attribute values fall in the range of the concept space of b_i, i=1,...,5.

Based on the elements in the concept hierarchies, the index space is partitioned into index buckets and the objects mapped into these index buckets. By this index structure, queries can be specified based on the elements of the concept hierarchies. It will be easy to map a query to a specific index bucket to retrieve the desired objects.

However, the number of index buckets can be as large as $|E_1|*|E_2|*|E_3|*|E_4|*|E_5|$. In order to reduce the storage space, the number of index buckets should be reduced, that is, only some elements are selected from the element set to form the index buckets. The selection of elements can be based on possible queries from users. In fact, only elementary elements are required for processing queries. However, the selection of frequently queried elements will enhance the performance of query processing and storage space.

The selected set of elements for each element set is denoted as SE_i, where $SE_i \subseteq E_i$, and $\{\text{all elementary elements in } E_i\} \subseteq SE_i$, i=1,...,5.

The following example shows some video objects and the index structure of these objects based on the concept hierarchy shown in Figure 1. Note that only attributes X and Y are considered.

Example 1 Assume there are three scenes in a video. The objects are expressed as (X, Y).

Scene1: Object 1-1: (2, 4)
Object 1-2: (0, 3)

Scene2: Object 2-1: (3, 4)
Scene3: Object 3-1: (2, 4)
Object 3-2: (6, 4)

The selected element sets are as follows:

SE_X = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 14}
SE_Y = {0, 1, 2, 3, 4, 5, 6, 7}

The index structure contains the following index buckets:

Bucket(2, 4) = {Object 1-1, Object 3-1}
Bucket(0, 3) = {Object 1-2}
Bucket(3, 4) = {Object 2-1}
Bucket(6, 4) = {Object 3-2}
Bucket(9, 4) = {Object 1-1, Object 3-1, Object 2-1}
Bucket(8, 3) = {Object 1-2}
Bucket(11, 4) = {Object 3-2}
Bucket(12, 4) = {Object 1-1, Object 3-1, Object 2-1}
Bucket(12, 3) = {Object 1-2}
Bucket(13, 4) = {Object 3-2}
Bucket(14, 3) = {Object 1-2}
Bucket(14, 4) = {Object 1-1, Object 3-1, Object 2-1, Object 3-2}

3.2 Motions

The motions can be represented as sequences of property values. We use suffix tree to index motions, which provides partial matching functions. For each Bucket(b_T, b_X, b_Y, b_S, b_C), a suffix tree is constructed for the motions of the corresponding objects. Moreover, for processing queries without the specification of attribute values, i.e., only motion specification is provided, we construct a suffix tree for all objects in the database (denoted Object_Motion(*, *, *, * ,*)). Figure 2 and Figure 3 show the index structures for motions in Example 2.

Example 2. For the video described in Example 1, the motions of the video objects are represented based on Velocity as follows:

Object 1-1: M→H→M→L
Object 1-2: L→M→H→M
Object 2-1: L→M→L→Z
Object 3-1: H→M→L→Z
Object 3-2: Z→L→M→L

Where H represents high speed, M represents medium speed, L represents low speed and Z represents zero speed.

4. QUERY PROCESSING

There are three types of queries: *concept query*, *approximate query* and *motion query*. Queries specified based singularly on the concept hierarchies are concept queries. Approximate queries request the system to extend the query results to some extent, and motion queries contain motion specifications.

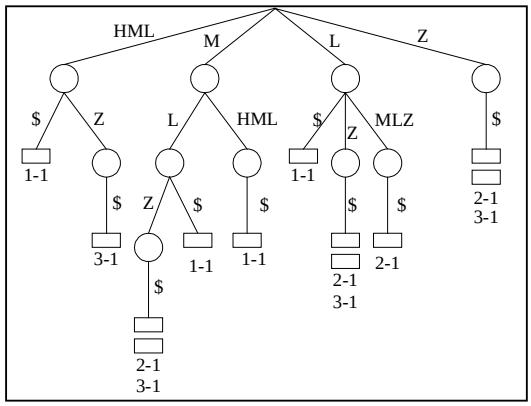

Figure 2. The suffix tree for Bucket(9, 4) and Bucket(12, 4)

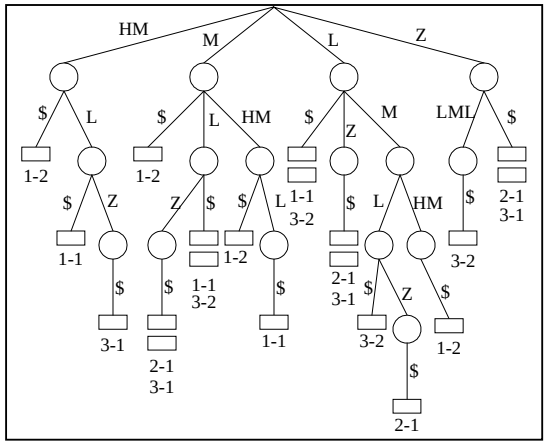

Figure 3. The suffix tree for Object_Motion(*, *)

4.1 Concept Query

A query can be specified based on different levels of concept hierarchies. When a high-level concept is specified, the video clips containing lower-level concepts will be retrieved as results. For an element e and a selected element set SE for a concept hierarchy, we define the *minimal cover set* (denoted MCS) of e as follows. If e∈ SE then MCS={e}, else MCS is the set of the descendents of e in SE, the union of these descendents' concept spaces equals to the concept space of e, and the set has the minimal cardinality.

A query can be represented as a 5-tuple (q_1, q_2, q_3, q_4, q_5) where $q_i \in E_i$, i=1,...,5. The minimal cover set of q_i is denoted as MCS_i. The query results are retrieved from the index structure, which can be expressed as follows:

$$QR = \{O \mid O \in Bucket(q'_1, q'_2, q'_3, q'_4, q'_5) \text{ where } q'_i \in MCS_i, i = 1,...,5\}$$

Figure 4 shows the algorithm to find the minimal cover set of an element e.

Algorithm minimal_cover_set

Input: An element e in E_i,
A selected element set SE_i
Output: The minimal cover set MCS of element e
Step1: MCS = {}
Step2: If e$\in$ SE_i
return {e}
Step3: for all children e' of e
if e'$\in$ SE_i
then add e' to MCS
else add minimal_cover_set(e') to MCS
Step4: Return MCS

Figure 4. Algorithm to find minimal cover set

Example 3. Assume the SE_X is {0, 1, 2, 3, 4, 5, 6, 7, 9, 11}, and the queries are Q1={11}, Q2={13}.

For Q1, since 11$\in$ SE_X, the MCS of 11 is {11}. For Q2, since 13$\notin$ SE_X, the MCS of 13 is {4, 5, 11}.

4.2 Approximate Query

Besides the concept query, a user may wish the system to extend the query results to some extent. These results may not match what the user specified, however, they are the results which the user may be interested in. In the proposed data model, location, size and color of video objects can be used to extend the results.

The similarity of locations is based on the distance. A user can specify a distance range for similar locations. The objects located in the range will be retrieved as results with different similarities. The similarity can be calculated using the distance between these two locations. Similar to the location approximation, the size approximation considers the objects with similar sizes of the objects specified in the query. The approximation of color is based on the color similarity table. The table records the similarity between different colors. The threshold of similarity is defined by the user. By referencing the similarity table, similar colors of the color specified in the query can be found.

4.3 Motion Query

If the motion queries also contain attribute predicates, the suffix trees of the corresponding index buckets will be traversed. On the other hand, i.e., only motions are specified, the suffix tree of Object_Motion(*, *, *, *, *) will be traversed to find the videos with matching object motions.

5. CONCLUSION

In this paper, the concept hierarchy is proposed for the concept query. The index structure based on the selected element set in the concept hierarchy is also proposed. To process a concept query, the objects in the corresponding index buckets will be retrieved. Our query processing considers the situation when the elements specified in the query are not in the selected element sets of the index structure. The suffix trees are used to index the motions of video objects in each index bucket.

In this approach, the simple suffix tree is used as the index structure for object motions. Our future work is to design an integrated index structure for static attributes and motions of video objects to enhance the processing performance and reduce the space of the index structure.

6. REFERENCE

[1] A. Aghbari, K. Kaneko and A. Makinouchi, “Modeling and Querying Videos by Content Trajectories,” IEEE International Conference on Multimedia & Expo, 2000.

[2] S. Dagtas, W. A.-Khatib, A. Ghafoor and A. Khokhar, “Trail-Based Approach for Video Data Indexing and Retrieval,” IEEE ICMCS, pp. 235-239, June 1999

[3] S. Dagtas, W. A. -Khatib, A. Ghafoor and R. L. Kashyap, "Models for Motion-Based Video Indexing and Retrieval," IEEE Transaction on Image Processing, Vol. 9, No. 1, pp.88-101, January 2000.

[4] Tony C. T. Kuo and Arbee L. P. Chen, “Indexing Query Interface and Query Processing for Venus: A Video Database System,” Proc. of Cooperative Databases for Advance Applications, 1996.

[5] Tony C. T. Kuo and Arbee L. P. Chen, “Content-Based Processing for Video Databases,” IEEE Transaction on Multimedia, Vol. 2, No. 1, pp. 1-13, 2000

[6] C. H. Lin and Arbee L. P. Chen, "Motion Event Derivation and Query Language for Video Databases," Storage and Retrieval for Media Databases 2001, Proceedings of SPIE Vol. 4315, pp.208-218

[7] C. C. Liu and Arbee L. P. Chen, “Vega: A Multimedia Database System Supporting Content-Based Retrieval,” Journal of Information Science and Engineering, Vol. 13, No. 3, pp. 369-398, 1997.

[8] C. C. Liu and Arbee L. P. Chen, “3D-List: A Data Structure for Efficient Video Query Processing,” to appear in IEEE Trans. on Knowledge and Data Engineering