

(Invited Review Paper)

Content-based Video Data Retrieval

ARBEE L. P. CHEN, CHIH-CHIN LIU, AND TONY C. T. KUO

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.

(Received June 4, 1998; Accepted September 21, 1998)

ABSTRACT

Many multimedia applications, such as the World-Wide-Web (WWW), video-on-demand (VOD), and digital library, require strong support of a video database system. In this paper, we survey various approaches to content-based video data retrieval. Compared with traditional keyword-based data retrieval, content-based retrieval provides a flexible and powerful way to retrieve video data. We discuss three approaches in depth. First, since video segmentation is an elementary task in video index construction, we discuss an approach to detect shot changes for video segmentation, which is based on the processing of MPEG compressed video data. Second, since a video object consists of a sequence of frames, the techniques for retrieving video objects based on the similarity of key frames are addressed. Third, we discuss a video query model and video query processing techniques based on the symbols in a video object. The notion of two-dimensional strings is extended to obtain three-dimensional strings (3D-Strings) for representing the spatial and temporal relationships among the symbols in both a video object and a video query.

Key Words: video databases, content-based retrieval, video indexing, shot change detection, video query processing, key frame matching, approximate query processing, motion track, temporal and spatial relationships, 3D-Strings

1. Introduction

In recent years, progress in the development of hardware and storage devices has made the use of digital multimedia, including video, images, graphics, animation, and audio, very common. Many multimedia applications, such as the World-Wide-Web (WWW), video-on-demand (VOD), and digital library, require strong support from a multimedia database system. In addition to the storage management for the multimedia objects, a main service that a multimedia database system should provide is to enable its users to easily and efficiently retrieve multimedia objects according to their content. As multimedia applications becomes more and more popular, the need for content-based multimedia data retrieval is getting more important (Chen *et al.*, 1995; Chiueh, 1994; Chou *et al.*, 1996; Dimitrova and Golshani, 1995; Gudivada and Raghavan, 1995; Jagadish, 1991; Liu and Chen, 1997, 1999; Myron *et al.*, 1995; Smoliar and Zhang, 1994; Wai and Chen, 1997; Weiss *et al.*, 1994; Yoshitaka, 1994). To provide the ability of content-based multimedia data retrieval, a multimedia query language or query interface should be developed by means of which users can

specify their queries. Furthermore, various techniques should be developed to efficiently process multimedia queries.

Most of the previous research on modeling and querying multimedia objects has focused on image databases (Chang *et al.*, 1988; Chiueh, 1994; Jagadish, 1991; Liu and Chen, 1996a; Myron *et al.*, 1995; Petrakis and Orphanoudakis, 1993). Unlike the QBIC project (Myron *et al.*, 1995), one can retrieve image objects according to their color, texture, or shape characteristics. Another approach to querying image databases is to construct an iconic index for the image objects (Chang *et al.*, 1987, 1988); image queries can then be posed against the symbols and their spatial relationships. Based on the R-tree data structure, a query language called PSQL was proposed (Roussopoulos *et al.*, 1988). Although PSQL is suitable for geographical data, such as maps, it is not suitable for general images, such as photographs and medical images.

Audio objects can be classified into two groups, music and sound objects, according to whether they have associated *stuffs*. For audio objects, Wold *et al.* (1996) proposed an approach to retrieve them based on their content. In their approach, an N-vector is

constructed according to the acoustical features of an audio object. These acoustical features include *loudness*, *pitch*, *brightness*, *bandwidth*, and *harmonicity*, which can be automatically computed from the raw data. The N-vector of acoustical features is then used to classify sounds for similar searching. However, the acoustical features are at a level too low for human beings. The most straightforward way for a naive user to query music databases is to hum a piece of music as the query example to retrieve similar music objects. This approach was adopted in Bakmutova *et al.* (1997), Balaban (1996), Chen and Chen (1998), and Ghias *et al.* (1995). Ghias *et al.* (1995) proposed an approach to transform a music query example into a string which consists of three kinds of symbols (“U”, “D”, and “S”, which represent a note is higher, lower, or the same as its previous note, respectively). The problem of music query processing is then transformed into that of approximate string matching. However, using only three kinds of symbols is too rough to represent the melody of a music object. Moreover, the proposed string matching algorithm does not take music characteristics into consideration. To develop a content-based music database system, we have started a project called *Muse* (Chou *et al.*, 1996; Chen and Chen, 1998; Liu and Chen, 1996b; Liu *et al.*, 1999). In this project, we have proposed three different methods for content-based music data retrieval. In Liu *et al.* (1999), we treated the rhythm, melody, and chords of a music object as a music feature string and developed a data structure called 1D-List to efficiently perform approximate string matching. Similarity measures in the approximate string matching algorithm are designed based on the music theory. In Chou *et al.* (1996), we considered music objects and music queries as sequences of chords. An index structure was developed to provide efficient partial matching ability. In Chen and Chen (1998), we propose an approach to retrieving music objects by means of rhythm.

Compared with other media types, such as text, image, and audio, video contains richer information (Gibbs *et al.*, 1993; Tonomura and Abe, 1994). However, this richness results in the lack of a generally accepted representation of video content. As many features can be used to represent the content of a video object, many approaches to content-based video data retrieval have been proposed recently. In this paper, we provide a survey of recent research results on content-based video data retrieval and address three important techniques for video indexing and video query processing.

The rest of this paper is organized as follows. In Section II, we summarize features which can be used to represent video objects. The indexing techniques

which support video data retrieval based on the cinematic structure of video objects are explained in Section III. A method for key frame matching is discussed in Section IV. The model and query processing techniques for video queries based on the spatial/temporal relationships of symbols in video objects are discussed in Section V. Finally, Section VI concludes this paper and provides some challenging research goals.

II. Features for Content-based Video Indexing

In this section, we analyse features which can represent the content of video objects.

- (1) Image characteristics of key frames: A video object consists of a sequence of image frames. We can use some representative frames (*key frames*) as an abstract image of a video object. Thus, image characteristics, such as color, texture, shape, and sketch, can be specified as query conditions to retrieve video objects which contain image frames which have similar characteristic measures (Myron *et al.*, 1995).
- (2) Cinematic structure: the raw data of a video object forms a stream of frames. However, a video object intrinsically has a hierarchical structure. The basic unit in the hierarchical structure is a *shot* (Smoliar and Zhang, 1994). Shots in the same place form a *scene*. Several scenes form a *sequence* (Allen, 1983). In general, a video object consists of a set of sequences. We can pose a video query against the structure of video objects, for example, to retrieve all the shots in the second scene of the first sequence of a video object.
- (3) Symbol containment: a *symbol* is an object appearing in a multimedia object (Kuo *et al.*, 1996; Liu and Chen, 1999). Assuming that the symbols contained in a multimedia object can be identified by means of suitable pattern recognition techniques or manually, we can retrieve multimedia objects containing some user-specified symbols.
- (4) Symbol motion tracks: In a video or animation object, the appearance of a symbol forms a three dimensional curve (3D curve). We call this curve the *motion track* of the symbol. Using a graphical query tool, users can draw a 3D curve to retrieve multimedia objects which contain symbols with similar motion tracks (Chen *et al.*, 1995).
- (5) Spatial and temporal relationships between symbols: there may exist spatial or temporal

relationships between the symbols in a multimedia object, and we can pose a multimedia query against these relationships. For instance, we can retrieve video objects which contain a frame with an eagle flying above a tree and a later frame with a lake.

III. Query by Cinematic Structure

For content-based retrieval, retrievals based on the image features of video frames are more efficient and practical. By measuring the similarities among video frames, a hierarchical cinematic structure (Davenport *et al.*, 1991), including the *shots*, *scenes*, *episodes* of a video, can be constructed as an index. Users can browse the hierarchical cinematic structure to retrieve cinematic units of interest. For example, users can specify the following query to play the first shot of the third scene of the second episode of the current video object (Liu and Chen, 1997):

```
SELECT S3.play()
FROM current Video V, V.Episode E1, E1.Scene S2, S2.Shot S3
WHERE E1.EpNum = 2
AND S2.ScNum = 3
AND S3.ShNum = 1
```

A shot is a sequence of frames which represents continuous action in time and space. The contents of the frames belonging to a shot are similar. Therefore, shot change detection can be performed through similarity measurement of continuous frames. Most of the previous works on shot change detection were based on the processing of uncompressed video data. Color histogram comparison and pairwise pixel comparison are two straightforward approaches (Tonomura and Abe, 1990; Nagasaka and Tanaka, 1991; Chua and Ruan, 1995) to similarity measurement. The color histogram approach summarizes the color distribution of a frame and computes the differences between it and the color distributions of its adjacent frames. When the difference exceeds a predefined threshold, a shot change is detected. Without considering the spatial distribution of colors, two different frames with the same color histogram will be treated as being very similar. In the pairwise pixel comparison approach, the values of the pixels are compared pixel by pixel. The sum of the differences of the values is computed. A shot change is detected when the sum exceeds a predefined threshold. The pairwise pixel comparison approach can easily lead to misdetections since it is very motion sensitive.

To improve the quality of detection, misdetections and the loss of detection should be avoided. A

misdetection may occur due to a quick variance of video contents, such as the effect caused by an electronic camera flash. A loss of detection may occur due to the similarity of video contents in consecutive shots. Otsuji and Tonomura (1993) and Ueda *et al.* (1991) considered the case of large differences in the contents of continuous frames due to fast motion (but not shot change). Using a filter for detecting such situations, misdetection in fast motion video frames can be reduced. Shot change detection in special applications (Brown, 1995; Philips and Wolf, 1996; Swanberg *et al.*, 1993; Zhang *et al.*, 1993), such as news programs, can achieve better results. This is because the detecting algorithm can focus on the characteristics of the applications, and knowledge can be provided to assist detection.

Adjero and Lee (1997) presented a method to dynamically adjust the threshold for detecting shot changes. Moreover, a window size (in terms of the number of frames) was defined to avoid falsely detecting more than one shot change in a short period of time. All these detection methods for uncompressed video data suffer from the following drawbacks: (1) processing is time consuming; (2) since video data are often stored in a compressed format, such as MPEG (Gall, 1991), video data should be decompressed in advance before processing.

Lin (1996), Chang (1997), Arman *et al.* (1993), and Yeung *et al.* (1995) proposed shot change detection algorithms based on compressed data. Arman *et al.* (1993) proposed an approach that computes the Discrete Cosine Transform (DCT) coefficients for each frame. These DCT coefficients of each frame are stored as a set of vectors. The inner product of the vectors of two continuous frames is computed to measure their similarity. When the similarity degree falls in a range where a shot change cannot be determined, the color histogram comparison approach has to be performed. In MPEG coded video data, a frame can be referenced by or can reference to other frames. The reference ratios can be computed for similarity measurement among frames. In Chang and Lee (1995), and Meng *et al.* (1995), both references and DCT coefficients were used to detect shot changes. Shot changes with the effect of *dissolve* were considered.

Kuo and Chen (1996) extended a method we proposed before to detect shot changes for MPEG coded video data. This approach analyzes references among MPEG coded frames. It is more efficient than that of Chang and Lee (1995) and Meng *et al.* (1995) since only the references of frames have to be evaluated. A function is used to quantize the evaluation results to shot change probabilities such that a shot change can be easily recognized.

1. MPEG Compressed Data Analysis

MPEG is a standard for video compression which achieves a high compression rate. It is popular for use in many applications. Video data are often stored in MPEG format. Shot change detection algorithms which perform image processing on raw video data are not suitable for MPEG coded video. Additional processing for decompressing compressed video into raw video has to be performed first. Therefore, it is more efficient to directly detect shot changes in MPEG compressed video. In order to improve the compression rate, in MPEG uses motion compensation technology to reduce the codes of similar image patterns among adjacent frames. Therefore, similarity matching is performed in the process of encoding. In the following, we will introduce the MPEG data format and discuss the information which can be used for shot change detection.

A. MPEG Data Format

In this section, we will introduce the information needed for shot change detection in MPEG coded data. The MPEG coding algorithm uses DCT to compress raw video data. Additionally, it uses block-based motion compensation to reduce temporal redundancy. By means of motion compensation, codes of similar blocks can be reduced by referencing to the image contents of adjacent frames. The more blocks a frame references, the more similar these two frames are. Therefore, by analyzing the references among coded frames, similarity can be determined.

In MPEG coding, a frame is divided into macroblocks. Each macroblock is a 16 by 16 image as a basic coding unit. A macroblock can be coded by DCT or references to its adjacent frames when it matches the similar image patterns of these frames. A macroblock coded by DCT is called an *intra-coded* macroblock. Macroblocks referencing to similar image patterns are called *forward-prediction coded*, *backward-prediction coded* or *bidirectional-prediction coded* macroblocks when they reference to the image patterns of the preceding frame, subsequent frame, or both preceding and subsequent frames, respectively. A reference to the preceding frame is called a *forward reference*, and on to the subsequent frame is a *backward reference*.

By means of referencing patterns of macroblocks, three types of frames, called the *I* frame, *P* frame and *B* frame, can be defined. All the macroblocks in an *I* frame must be intra-coded macroblocks. That is, the *I* frame is independently coded. It can be decompressed without referencing to other frames. Macroblocks of the *P* frame may have forward references to its pre-

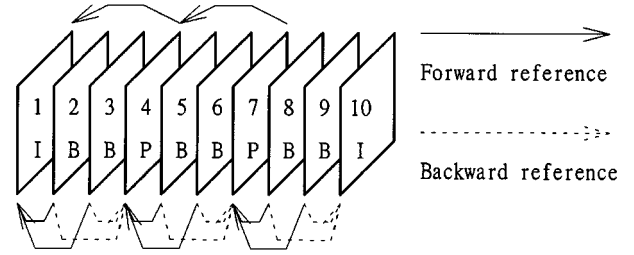


Fig. 1. Example of frame references.

ceding *I* or *P* frame. That is, the macroblock is a forward-prediction coded macroblock when a similar image pattern is found in the preceding *I* or *P* frame. The macroblock is intra-coded when a similar image pattern can not be found in the preceding *I* or *P* frame. A *B* frame may have references to its adjacent *I* or *P* frames. Bidirectional references are allowed. The macroblock in a *B* frame can be a bidirectional-prediction coded, forward-prediction coded, or backward-prediction coded macroblock.

In MPEG coded video, the number and sequence of *I*, *P*, and *B* frames are pre-determined. In general, there may be a number of *P* and *B* frames between two *I* frames, and a number of *B* frames between two *P* frames or between an *I* and a *P* frame. An example is shown in Fig. 1 to illustrate the structure of MPEG coded frames. The ratio of the numbers of *I*, *P*, and *B* frames (called the *IPB-ratio*) is 1:2:6. An *I* frame is followed by two *P* frames and six *B* frames in a sequence.

B. References among Video Frames

For *P* frames and *B* frames, macroblocks may reference to adjacent frames. We can compute the number of macroblocks for each type of reference to measure the similarity with the adjacent frames. We define two types of *reference ratios* (RRs) as follows:

(1) *Forward reference ratio*

$$(FRR) = R_f / N,$$

where R_f is the number of the forward-prediction coded macroblocks of a frame and N is the number of total macroblocks of the frame.

(2) *Backward reference ratio*

$$(BRR) = R_b / N,$$

where R_b is the number of the backward-prediction coded macroblocks of a frame and N is the number of total macroblocks of the frame.

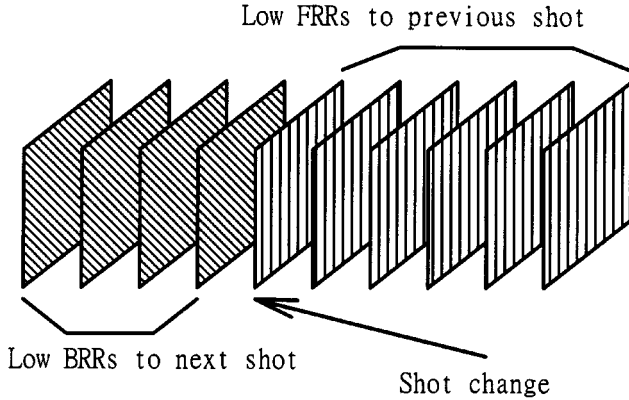


Fig. 2. The varieties of reference ratios when a shot change occurs.

The range of both FRR and BRR is between 0 and 1. A P frame has an FRR. A B frame has both an FRR and a BRR. When the FRR is high, this indicates that the frame is similar to its preceding frame. When the BRR is high, this indicates that the frame is similar to its subsequent frame. The RR is regarded as high when it exceeds a given threshold. An I frame has no FRR or BRR. Therefore, to measure the similarity between an I frame and its adjacent frames, we have to evaluate the FRR or BRR of these adjacent frames.

In a video sequence, the contents of continuous frames are similar when the shot does not change. Therefore, the reference ratios of these frames are high. When a shot change occurs, the contents of the frames are not similar to those of the preceding frames. The reference ratios are then low.

In the next section, we will propose an approach to detecting shot changes which evaluates the reference ratios of MPEG coded frames. Since only the information of the reference ratios of frames has to be computed, there is no need to decompress each coded frame. A large amount of time can, thus, be saved. For example, a video sequence contains 10,000 continuous frames. Each frame is a 256 by 256 image. That is, a frame contains 256 macroblocks. To compute the reference ratio of a frame, 256 add operations must be performed. This approach is more efficient than color histogram based approaches and the approach which computes the DCT coefficients of frames.

2. Shot Change Detection

A. Shot Change Occurrence Analysis

A shot change often causes the contents to be different from those of the previous shot. Therefore, frames of the previous shot may have low BRRs to the next shot. On the other hand, frames of the next shot

may have low FRRs to the previous shot, as shown in Fig. 2.

A shot change may occur in any type of frame. In the following, we will consider situations in which shot changes occur at I frames, P frames and B frames, respectively.

- (1) A shot change occurs in an I frame: Because I frames are encoded independently of other frames, they do not have forward and backward references. What we need to take into account is the B frames between this I frame and the preceding I or P frames. These B frames use this I frame as a backward reference for encoding. They cannot easily find similar image patterns from this I frame, so their BRRs must be low. We do not consider the FRRs of these B frames since they are not relevant to this I frame. The B frames between this I frame and the subsequent P frame need not be considered since they are not relevant to shot change detection.
- (2) A shot change occurs at a P frame: The B frames between this P frame and the preceding I or P frame behave in the same way as previously described. The difference in this case is that P frames have forward references. Since this P frame is the shot change frame, it cannot easily find similar patterns from the preceding I or P frame. The forward reference will be low.
- (3) A shot change occurs at a B frame: This B frame itself will have a low FRR. If there exist B frames between this B frame and the preceding I or P frame, their BRRs must be low. If there exist B frames between this B frame and the next I or P frame, their FRRs must be low, too. Furthermore, if the first non-B frame in the following sequence is a P frame, the FRR of this P frame must be low.

Consider the MPEG video sequence shown in Fig.

3. If a shot change occurs at I frame 13, the B frames 11 and 12 will have low BRRs. If a shot change occurs in P frame 10, the BRRs of B frames 8 and 9 will be low, and so will the FRR of P frame 10. The situation is different when a shot change occurs at B frame 5 or B frame 6. If B frame 5 is the shot change frame, P frame 7 and B frames 5 and 6 will have low FRRs. If a shot change occurs at B frame 6, the BRR of B

I	B	B	P	B	B	P	B	B
1	2	3	4	5	6	7	8	9
P	B	B	I	B	B	P...		
10	11	12	13	14	15	16...		

Fig. 3. An example video sequence.

...	I	B	B	P	B	B	P	B
...	0	1	2	3	4	5	6	7

B	P	B	B	I	...
8	9	10	11	12	...

Fig. 4. An example of computing the shot change probability.

frame 5 will be low, and so will the FRRs of P frame 7 and B frame 6.

B. The Mask Matching Approach

From the above analysis, to detect whether a frame has a shot change, the FRRs and/or BRRs of this frame and its adjacent frames have to be examined. In this section, we will present a *mask matching* approach to detection of possible shot changes. This approach examines MPEG coded video frame by frame. For each video, a set of masks is defined. The RRs of the frames specified in the masks are evaluated. Different types of frames have to be matched with different masks. When a frame is matched with the mask, it is detected as a shot change frame. Since there are I, P, and B frames, the types of masks are denoted as I_frame masks, P_frame masks, and B_frame masks, respectively.

A *mask* denotes the qualification for detecting shot changes in frames. It consists of two parts. One is the type of this mask. The other is a sequence of *mask frames* which have to be examined. A mask frame M_i can be denoted as follows:

$$M_i = FR,$$

where $F \in \{I, P, B\}$, $R \in \{f, b\}$. F denotes the frame type of this mask, and R denotes the RR, which should be low (f for FRR and b for BRR). High RRs are not used to detect the occurrences of shot changes.

A mask M can be denoted as

$$M = \{mask_type; (M_1, M_2, \dots, M_n)\},$$

where $mask_type \in \{I, P, B\}$, M_i are mask frames.

To denote a sequence of frames, the mask frame beginning with an '@' indicates the current frame. For example, as shown in Fig. 4, there are four masks for the video with the IPB-ratio 1:2:6. Mask M_1 is for the I frame and M_2 is for the P frame. Because of the IPB-ratio 1:2:6, the B frame may encounter two different situations: in one, it is preceded by an I or a P frame and followed by a B frame, and in the other, it is

preceded by a B frame and followed by an I or a P frame. Therefore, there are two masks, M_3 and M_4 , for the B frame. M_3 indicates that (1) the current B frame should have a low FRR, (2) its subsequent B frame should have a low BRR, and (3) its subsequent P frame should have a low FRR. If the subsequent frame is an I frame, it can be skipped.

We will use the previous example shown in Fig. 3 to demonstrate this kind of checking. To check I frame 13, the M_1 mask is applied. By checking the mask frames of M_1 , the preceding two B frames should have low BRRs when I frame 13 has a shot change. That is, B frame 11 and 12 have low BRRs.

In mask matching, to determine whether a frame has a low reference ratio, the reference ratio has to be compared with a predefined threshold. Different types of videos may have different thresholds.

C. Implementation

Some experiments were done to verify the validity of the mask matching approach. In the experiments, we designed a function to transform the results of mask matching into *shot change probabilities*. The probability was low when a frame was similar to its adjacent frames. This function will be introduced in the following section.

Our approach takes advantage of the concept of mask matching to detect whether a frame has a shot change or not. To implement this concept, the results of mask matching are quantized to a value which indicates the shot change probability. The shot change probability function P is as follows:

$$P = 1 - \frac{RR_{f_1}^2 + RR_{f_2}^2 + \dots + RR_{f_n}^2}{RR_{f_1} + RR_{f_2} + \dots + RR_{f_n}}, \quad (1)$$

where $f_1, f_2, \dots, f_n \in$ the mask frames of the current frame and RR_{f_i} is the corresponding RR of mask frame f_i . If $\forall RR_{f_i} = 0, 1 \leq i \leq n$, then P is set to 1:

$$M1 = \{I; (Bb, Bb, @I)\};$$

$$M2 = \{P; (Bb, Bb, @Pf)\};$$

$$M3 = \{B; (@Bf, Bf, Pf) \text{ or } (@Bf, Bf, I)\};$$

$$M4 = \{B; (Bb, @Bf, Pf) \text{ or } (Bb, @Bf, I)\};$$

Masks of the video with IPB-ratio 1:2:6.

The shot change probability is between 0 and 1. The larger the value is, the more possible it is that a

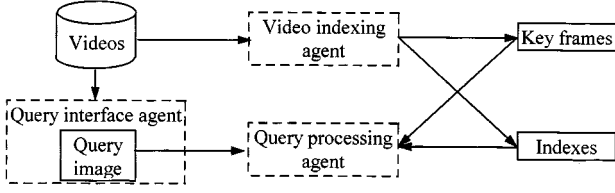


Fig. 5. System overview of the video retrieval system.

shot change will occur at the frame. The second term in Eq. (1) is the weighted sum of the corresponding RRs of mask frames. Based on the weighted sum, if one of the RRs is much larger than the others, then the result of the weighted sum will approach the larger RR. Therefore, the shot change probability will be low if there exists a mask frame with a high RR. For example, consider the video stream shown in Fig. 4. The mask used to detect P frame 6 was {P; (Bb, Bb, @Pf)}.

Suppose the BRR of B frame 4, BRR of B frame 5 and FRR of P frame 6 are all 0.2. The probability that a shot change will occur at P frame 6 is computed as $(1-0.2)=0.8$. This indicates that P frame 6 is very probably a shot change frame.

We will use Fig. 5 to illustrate another example. Suppose the BRR of B frame 4 is 0.8, the BRR of B frame 5 is 0.2 and the FRR of P frame 6 is 0.2. The shot change probability can be computed as $(1-0.6)=0.4$ by applying (2.1). The probability that a shot change will occur at P frame 6 is low in this case.

After all the shot change probabilities are computed, a threshold is defined to get the final result. As long as the shot change probability of a frame is larger than the threshold, it is regarded as a *shot change frame*.

IV. Query by Means of Key Frame Matching

This section will discuss techniques for the retrieval of videos based on key frames. In this approach, some representative *key frames* are extracted as an index. Users can query the videos using an example image, and the system compares the query image and the key frames to find possible results. Parsing the video into shots (i.e., shot change detection) is the first step in constructing such indexes for querying video data. Similarity measurement is based on comparison of the histograms of dominant colors. Moreover, the average luminance value of the image is applied as a filter to avoid matching on all key frames. Experiments have been performed to show how this approach is used.

1. Approach Overview

This subsection will present a system overview of our approach. It contains three agents. They are *video indexing agent*, *query interface agent*, and *query processing agent*, as shown in Fig. 5.

- (1) The video indexing agent constructs key frames for incoming videos. A key frame is a dominant frame of a shot, where a shot presents a sequence of continuous video frames. The image contents of the key frames are stored for similarity matching when a query is posed. When the image contents of the key frames are stored, the average luminance of each key frame is computed and stored as an index of a filter.
- (2) The query interface agent enables users to specify queries and invokes the query processing agent for query results. A query is simply specified by giving a sample image (called a *query image*). The threshold of the similarity degree can be adjusted for different matching criteria. Video clips containing the key frames which are similar to the query image are shown as query results.
- (3) The query processing agent analyzes the query image and compares it with the key frames for most similar matching. It can be divided into two steps. First, we compute the average luminance of the query image, and key frames with similar average luminance are chosen for exhaustive matching. It avoids exhaustive matching on all key frames. Second, matching is performed between query images and the selected key frames. The matching algorithm will be presented in the next section. The query results are returned to the user via the query interface agent.

2. Similarity Matching of Key Frames

Similarity matching is based on comparison of the color histograms of dominant colors of different image blocks. The color information is first analyzed. The steps are shown in Fig. 6.

For an image, we count the number of pixels of each color. The top n colors with the maximal number of pixels are selected as dominant colors. Furthermore, the image is divided into several square subimages. The color histogram of the dominant colors is then

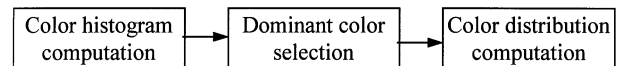


Fig. 6. Steps for color information analysis.




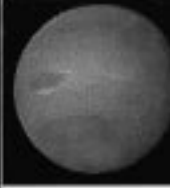

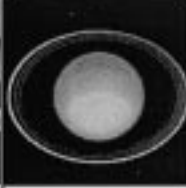
Rank	1	2	3	4	5
					
Similarity	0.811849	0.78121	0.774455	0.762532	0.746094
Class	Astronomy	Astronomy	Astronomy	Person	Astronomy

Fig. 7. Similarity matching between an astronomy image and other images.

computed for each image block. It considers the color distribution feature in the color histogram computation. In the following, we will present the algorithm for comparison between a query image and key frames.

We will first define a set of parameters that will be used in our presentation.

- (1) C_d : the number of dominant colors. The default value is 64.
- (2) n : the block size of a subimage. Each subimage is an $n \times n$ image block. The default value is 16.
- (3) R : the range of colors that will be treated as identical. For example, suppose $R=128$ in the 256 gray scale domain; the gray scales of 0~127 are treated as the same color, and so are the scales 128~255. When R is specified, the domain of colors is reduced.

These parameters can be adjusted for different applications. Next, we will present the algorithm for similarity matching between two images.

- (1) Input the query image and a key frame.
- (2) Transform the colors of the two images into the new domain based on R .
- (3) Compute the global color histogram to select the top C_d dominant colors for the query image.
- (4) Divide each image, including the query image and key frame, into a set of subimages, where each subimage is an $n \times n$ image block.
- (5) Compute the similarity for two corresponding subimages:
 - (i) Let sim be the similarity degree of the two subimages,

$$0 \leq sim \leq 1.$$

Let $diff$ be the summation of the differences of the numbers of pixels for each dominant color of the two subimages.

Let $P1(i)$ be the number of pixels for dominant color Di of subimage 1, $P2(i)$ be the

number of pixels for dominant color Di of subimage 2.

(ii) $diff=0$

For $i=1$ to C_d

$$diff=diff+|P1(i)-P2(i)|$$

$$sim=1-diff/(n \times n \times 2).$$

Note: When the number of pixels of each dominant color does not vary, $diff$ is equal to 0, and sim is equal to 1.

The cost of similarity matching between two images is high since the pixel values have to be computed online. To reduce the processing time, exhaustive matching between all key frames and the query image should be avoided.

We provide an *average luminance value* (ALV) filter to prune key frames which are not query results. It is based on a comparison of the ALV of the images to be pruned. The ALV is the average of the luminance of all the pixels in the image. Therefore, the range of the ALV of an image is 0~255. For all key frames, the ALVs can be stored in advance. When processing a query, the ALV of the query image is computed. Key frames which have similar ALVs are the possible results of the query. Additional similarity matching has to be performed between these key frames and the query image. On the other hand, key frames which do not have similar ALVs are pruned. A threshold can be defined to determine whether two ALVs are similar or not.

3. Experiment Results

The experiments were designed to demonstrate how our approach. The experiments can be divided into two parts. First, we performed the similarity matching algorithm on 100 images, which were classified into five classes. The results show that images

Rank	1	2	3	4	5
					
Similarity	0.750855	0.747762	0.744954	0.743571	0.731934
Class	Painting	Painting	Painting	Painting	Painting

Fig. 8. Similarity matching between a painting image and other images.

which belonged to the same class may have had a high degree of similarity. Second, we chose three videos and randomly queried the image contents of each video. The results show that the queried images could be matched to the key frames which belonged to the same shot of the query images.

We chose 100 images in five classes as our experimental images. The five classes were animal, painting, astronomy, person, and plant. Each class had 20 images. We randomly selected one image from each class and compared it with all the images. Figures 7 and 8 show the results of the classes astronomy and painting. The left most one is the query image. The other images ranked as 1~5 in similarity. We found that similar images usually belonged to the same class.

The above experiments showed good results for matching of similar types of images.

V. Query by Spatial and Temporal Relationships among Symbols

Chang *et al.* (1987, 1988) proposed the concept of a 2D-string structure for representing the contents of images. In this approach, each object in an image is represented by a symbol, and the orders of all the symbols along the x-axis and y-axis are stored in two strings. The notion of 2D-string structure can be extended with some modifications to take into account the characteristics of video, and the 3D-String structure for representation a video query has been defined (Liu and Chen, 1998). The problem of video query processing is then transformed into a problem of three-dimensional pattern matching. Many string matching algorithms (Aho and Corasick, 1979; Baeza-Yates and Gonnet, 1992; Boyer and Moore, 1977) and pattern matching algorithms (Baker, 1978; Bird, 1977; Fan and Su, 1993) have been proposed in the past. However, they are not suitable for 3D-Strings since the relationships between the symbols in a 3D-String are much

more complex than those between symbols in a string or pattern.

In this section, a video query model based on the content of video and iconic indexing is discussed. We extend the notion of two-dimensional strings to three-dimensional strings (3D-Strings) to represent the spatial and temporal relationships among the symbols in both a video and a video query. The problem of video query processing is then transformed into a problem of three-dimensional pattern matching. We have developed an efficient three-dimensional pattern matching mechanism. To process a video query, we first construct a 3D-String to represent the spatial and temporal relationships between symbols in the video query. Then, the symbol objects of a video object to be evaluated in the video database are retrieved and organized as a 3D-List according to the 3D-String. The 3D-List is a compact graph representation of the spatial-temporal relationships between symbol objects in a video object. Then, the 3D-List refinement algorithm is applied to the 3D-List to reduce the number of symbol objects in the 3D-List. Finally, the refined 3D-List is traversed to determine whether the video object is an answer to the video query.

1. Video Index Tool and Video Index Structures for Symbol Objects

The video index tool shown in Fig. 9 is a graphical interface for building a video index of video objects. It consists of a video index window and a video playback window. The video index window consists of a set of VCR buttons, an icon list, and an index display area. It allows users to interactively select interesting symbol objects of a video object and build corresponding indices for these symbol objects. First, the video object to be indexed is selected and played. When an interesting symbol object appears, the user presses the VCR button to pause the playback, chooses an icon represent-

ing the symbol object from the icon list and puts it at the position where the symbol object appears in the video playout window. The corresponding symbol object will appear in the index display window. This step is repeated for interesting symbol object in this video frame. The object identifiers of the video object, the icon that the symbol object belongs to, and the symbol object itself, the x, y coordinate values of the central point of the symbol object, and the number of the frame at which the symbol object first appears in the video object are stored in a table called a *video index table*. Table 1 shows an example of a video index table. The video object V0001 contains ten symbol objects. These symbol objects are associated with three icons.

2. A Motivative Example

Before we formally describe the concept of 3D-Strings for representing video queries and the data structure 3D-List for processing video queries, in this subsection, we will use an example to illustrate our approach.

Assume a video query Q contains three icons, A , B , and C . Icons A and B are at the same place in the x-axis, and icon C is on the right side of icon B (and icon A). This information can be denoted as the *string* $A \equiv B \Rightarrow C$. This notation can also be used to represent the relative positions between symbol objects in a video object. For example, assume that a video object V has 16 symbol objects, $a_1, a_2, a_3, b_1, b_2, c_1, c_2, c_3, d_1, d_2, d_3, e_1, e_2, f_1, f_2$, and f_3 (where a_1, a_2 , and a_3 belong to icon A , b_1 and b_2 to icon B , etc.), and that their relative positions are denoted as $a_1 \equiv a_2 \Rightarrow b_1 \Rightarrow f_1 \Rightarrow d_1 \Rightarrow f_2 \Rightarrow a_3 \Rightarrow b_1 \Rightarrow a_2 \Rightarrow c_1 \Rightarrow d_2 \Rightarrow e_1 \Rightarrow c_2 \Rightarrow d_3 \Rightarrow c_3 \Rightarrow f_3$. We say that the video object V is an answer to the video query Q if V contains three symbol objects, a_i, b_j , and c_k , such that $a_i \equiv b_j \Rightarrow c_k$.

To decide whether V is an answer to Q , a straight-

Table 1. An Example Video Index Table

<i>video_oid</i>	<i>icon_oid</i>	<i>symbol_oid</i>	<i>x</i>	<i>y</i>	<i>frame</i>
V0001	E	S0001	0	2	4
V0001	E	S0002	2	0	15
V0001	E	S0003	1	1	20
V0001	T	S0004	0	1	1
V0001	T	S0005	0	2	25
V0001	T	S0006	2	0	40
V0001	H	S0007	2	1	1
V0001	H	S0008	1	1	31
V0001	H	S0009	2	2	22
V0001	H	S0010	0	2	40

forward method is to match the two associated strings. However, due to the complexity of the relationships between the symbol objects, this method is very inefficient. For example, to find whether a_1 and b_2 match $A \equiv B$, we have to check $\Rightarrow f_1 \equiv d_1 \Rightarrow f_2 \Rightarrow a_3 \equiv b_1 \Rightarrow a_2 \Rightarrow c_1 \equiv d_2 \Rightarrow e_1 \Rightarrow c_2 \Rightarrow d_3 \Rightarrow c_3 \Rightarrow f_3$ to find the symbol object that matches $\Rightarrow C$.

Instead of directly matching the two strings, we use the string associated with a video query as a *template* and use a data structure to see whether the symbol objects of a video object can fit the template. Only those symbol objects of a video object that are associated with the icons of a video query need to be retrieved and checked. In this example, only $a_1, a_2, a_3, b_1, b_2, c_1, c_2$, and c_3 are retrieved, and three sets, $\{a_1, a_2, a_3\}$, $\{b_1, b_2\}$, and $\{c_1, c_2, c_3\}$, are formed. The next step is to check whether there exist three symbol objects selected from $\{a_1, a_2, a_3\}$, $\{b_1, b_2\}$, and $\{c_1, c_2, c_3\}$, respectively, that match $A \equiv B \Rightarrow C$. Instead of checking $3 \times 2 \times 3 = 18$ combinations of these symbol objects, we have developed a data structure to efficiently perform the matching process. First those symbol objects associated with the same icon are arranged according to their sequence in the string as shown in Fig. 10(a). Then a_3 and a_2 are linked since they are at the same position, which implies that they can be treated as a symbol object for checking. Similarly, c_2 and c_3 are linked. The number of combinations is reduced to $2 \times 2 \times 2 = 8$. Checking the relative position between a_1 and b_2 , we find they match $A \equiv B$. a_1 and b_2 are, thus, linked as shown in Fig. 10(b). a_1 and b_2 need not be checked since b_2 and b_1 are not at the same position (they are not linked). Similarly, a_3 and b_2 need not be checked since a_3 and a_1 are not at the same position. Also, a_2 and b_2 need not be checked since a_2 and a_3 are not at the same position. The remaining checking process is as shown in Fig. 10(c)-(f). Totally, only six checkings are needed.

The discussion above only considers the relation-

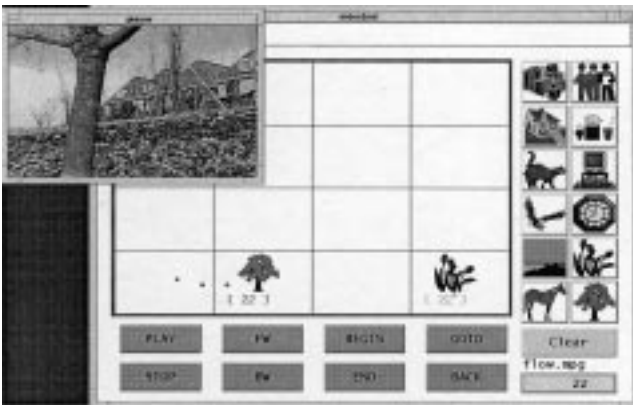


Fig. 9. The video index tool.

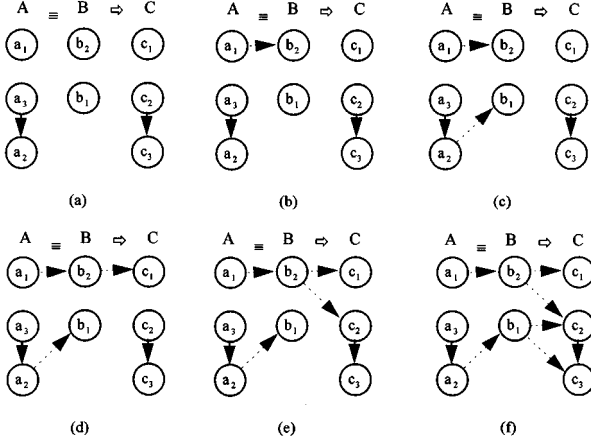


Fig. 10. A motive example for video query processing.

ships between symbol objects in the x-axis. When we evaluate a video object against a video query, the relationships between the symbol objects in the x-axis, y-axis, and time-axis should all be checked. An efficient algorithm should be developed to combine the one-dimensional results into the three-dimensional results. This is another important problem we have to deal with when we process a video query.

3. The Representation of Video Queries

In this subsection, we will describe the video query model and introduce the notion of 3D-Strings, which are used to represent the spatial and temporal relationships between the icons in a video query.

Symbol objects which represent the same kind of real world entities are grouped into an icon. The icons in the video database system form an icon hierarchy. Each icon has a graphical notation. A user can operate the query interface tool to select the icons of a video query, to place these icons at some locations on the screen to specify their spatial relationships, and to attach to each icon a time interval to specify the period of time during which the icon appears. We define the position of an icon by combining the geometrical location and the temporal location of the icon.

Definition V.1. (Position of an Icon) Assume that the resolution of the screen showing the query interface tool is $X_{\max} \times Y_{\max}$ pixels. The position of an icon in a video query is defined as a triple (x, y, t) , where x and y are the coordinate values in the x-axis and y-axis of the central point of the icon, and t is the number of the frame at which the icon first appears in the video query.

According to this definition, we only recognize

the central point of the icon. Information about the shape or size of the icon is omitted to reduce the complexity of query processing. We will relax this limitation in the future.

Because the resolution of the screen showing the query interface tool and the resolution (size) of each video object in the video database can be different, a *uniform resolution model* should be provided as a basis for performing similarity comparison between video queries and the index of video objects. Thus, the screen area is divided into $X_{\text{rank}} \times Y_{\text{rank}}$ grids of equal size, where X_{rank} and Y_{rank} are user specified. This concept can be extended to the time dimension. The total time intervals of a video query and each video object are both divided into T_{rank} time intervals. The position of an icon in the uniform resolution model, also called the *rank* of the icon, can be defined as follows.

Definition V.2. (Rank of an Icon) Assume that the position of an Icon I is (x, y, t) , that the screen resolution of the query interface tool is $X_{\max} \times Y_{\max}$ pixels and is divided into $X_{\text{rank}} \times Y_{\text{rank}}$ grids, and that the total time interval of a video query is set to T_{\max} frames and is divided into T_{rank} time intervals. The rank of the icon I is defined as the triple $(R_x(I), R_y(I), R_t(I))$, where $R_x(I) = \lfloor x \times X_{\text{rank}} / X_{\max} \rfloor$, $R_y(I) = \lfloor y \times Y_{\text{rank}} / Y_{\max} \rfloor$, and $R_t(I) = \lfloor t \times T_{\text{rank}} / T_{\max} \rfloor$.

For any two icons in a video query, basically, there are two kinds of spatial/temporal relationships between them, i.e., *adjacent relationships* and *appositional relationships* as defined in the following.

Definition V.3. (Adjacent Relationships) For any two icons I_1 and I_2 in a video query, I_1 is adjacent to I_2 in the x-axis with distance n , denoted by $I_1 \mid_n I_2$ if and only if $R_x(I_1) - R_x(I_2) = n$. Similarly, I_1 is adjacent to I_2 in the y-axis or t-axis with distance n if and only if $R_y(I_1) - R_y(I_2) = n$ or $R_t(I_1) - R_t(I_2) = n$, respectively.

Definition V.4. (Appositional Relationships) For any two icons I_1 and I_2 in a video query, I_1 is appositional to I_2 in the x-axis, denoted by $I_1 \equiv I_2$, if and only if $R_x(I_1) = R_x(I_2)$. Similarly, I_1 is appositional to I_2 in the y-axis or t-axis if and only if $R_y(I_1) = R_y(I_2)$ or $R_t(I_1) = R_t(I_2)$, respectively.

The adjacent relationship and the appositional relationship form the *normal spatial-temporal relationship set* $\{\mid_n, \equiv\}$.

Having defined adjacent relationships and appositional relationships, we can further define the *1D-String* and the *3D-String* notation for representation of the spatial-temporal relationships between the icons of

a video query.

Definition V.5. (*Normal 1D-Strings*) A normal 1D-String of length k is a string of the form $I_1\alpha_1I_2\alpha_2I_3\ldots\alpha_{k-1}I_k$, where each I_i is an icon and each α_j is in $\{|_n, \equiv\}$.

Definition V.6. (*Normal 3D-Strings*) A normal 3D-String of length k is a triple (X, Y, T) , where X , Y , and T are 1D-Strings of the form $I_1\alpha_1I_2\alpha_2I_3\ldots\alpha_{k-1}I_k$, $I_1'\beta_1I_2'\beta_2I_3'\ldots\beta_{k-1}I_k'$, and $I_1''\gamma_1I_2''\gamma_2I_3''\ldots\gamma_{k-1}I_k''$, respectively. In these strings, each I_1 , I_1' , and I_1'' is an icon, each α_j , β_j , and γ_j is in $\{|_n, \equiv\}$, and $\{I_1, I_2, I_3, \ldots, I_k\} = \{I_1', I_2', I_3', \ldots, I_k'\} = \{I_1'', I_2'', I_3'', \ldots, I_k''\}$.

Example V.1. Assume that A, B, C , and D are four icons. $(A\equiv B|_1 C|_2 D, B|_1 C\equiv A|_2 D, C|_2 D\equiv B\equiv A)$ is a 3D-String since $\equiv, |_1$, and $|_2$ are in $\{|_n, \equiv\}$, and $\{A, B, C, D\} = \{B, C, A, D\} = \{C, D, B, A\}$. $(A\equiv B|_1 C, B|_1 C|_2 D, C|_2 D\equiv A)$ is not a 3D-String since $\{A, B, C\} \neq \{B, C, D\} \neq \{C, D, A\}$.

The adjacent relationships and the appositional relationships are a straightforward representation of the spatial-temporal relationships between the icons of a video query. There may exist other representations for them. For example, a user specifies an eagle icon above a tree icon, but he or she may not know how far the distance is between them or may not care about their relative positions. Therefore, we further define the precedent relationships and the unknown relationships for these situations.

Definition V.7. (*Precedent Relationships*) For any two icons I_1 and I_2 in a video query, I_1 is precedent to I_2 in the x-axis, denoted $I_1\Rightarrow I_2$, if and only if $R_x(I_1) < R_x(I_2)$. Similarly, I_1 is precedent to I_2 in the y-axis or t-axis if and only if $R_y(I_1) < R_y(I_2)$ or $R_t(I_1) < R_t(I_2)$, respectively.

Definition V.8. (*Unknown Relationships*) For any two icons I_1 and I_2 , there is an unknown relationship between I_1 and I_2 , denoted $I_1? I_2$, if and only if I_1 and I_2 appear in the same video query and their relative position is unknown.

The adjacent relationships, the appositional relationships, the precedent relationships, and the unknown relationships form the *extended spatial-temporal relationship set* $\{|_n, \equiv, \Rightarrow, ?\}$.

Definition V.9. (*Video Query*) A video query Q is a triple $(I, R, Q\text{-type})$, where I is the set of all icons

referred to in the query, R is the set of the ranks of icons in I , and $Q\text{-type}$ is a number whose value is 0, 1, or 2, denoting the way to translate the spatial-temporal relationships between the icons in I . If $Q\text{-type}$ is 0, all the spatial-temporal relationships between icons in I are translated into unknown relationships. If $Q\text{-type}$ is 1, the adjacent relationships between icons in I are translated into precedent relationships. If $Q\text{-type}$ is 2, the adjacent relationships and the appositional relationships between icons in I are retained.

We will next define extended 1D-Strings and extended 3D-Strings for representing video queries.

Definition V.10. (*Extended 1D-Strings*) An extended 1D-String of length k is a string of the form $I_1\alpha_1I_2\alpha_2I_3\ldots\alpha_{k-1}I_k$, where each I_i is an icon and each α_j is in $\{|_n, \equiv, \Rightarrow, ?\}$.

Definition V.11. (*Extended 3D-Strings*) An extended 3D-String of length k is a triple (X, Y, T) , where X , Y , and T are 1D-Strings of the form $I_1\alpha_1I_2\alpha_2I_3\ldots\alpha_{k-1}I_k$, $I_1'\beta_1I_2'\beta_2I_3'\ldots\beta_{k-1}I_k'$, and $I_1''\gamma_1I_2''\gamma_2I_3''\ldots\gamma_{k-1}I_k''$, respectively. In these strings, each I_i , I_1' , and I_1'' is an icon, each α_j , β_j , and γ_j is in $\{|_n, \equiv, \Rightarrow, ?\}$, and $\{I_1, I_2, I_3, \ldots, I_k\} = \{I_1', I_2', I_3', \ldots, I_k'\} = \{I_1'', I_2'', I_3'', \ldots, I_k''\}$.

A video query can be transformed into an extended 3D-String. The transformation requires two steps. First, a video query is transformed into a normal 3D-String according to the ranks of the icons in the video query as stated in Algorithm V.1 and Algorithm V.2. Second, the normal 3D-String is transformed into an extended 3D-String according to the query type as stated in Algorithm V.3 and Algorithm V.4.

Algorithm V.1. (*Build_1D_String(Q, X)*)

/ input : a video query Q */*

/ output : a normal 1D-String */*

0: *begin*

1: $n \leftarrow 1$

2: $X \leftarrow \phi$

3: *for* $i=1$ *to* R_x

4: *begin*

5: *if do not exist any icon I with $R_x(I)=i$*

6: $n \leftarrow n+1$

7: *else*

8: *begin*

9: *pick an icon with $R_x(I)=i$*

10: *if* $X=\phi$

11: $X \leftarrow I$

```

12:      else
13:           $X \leftarrow X + \text{"|}_n\text{"} + I$ 
14:      end if
15:       $n \leftarrow 1$ 
16:      for each icon  $I'$  with  $R_x(I')=i$  and  $I' \neq I$ 
17:           $X \leftarrow X + \text{"}\equiv\text{"} + I$ 
18:      end
19:  end if
20: end
21: end

```

Algorithm V.1 constructs a normal 1D-String from a video query in the x-axis. The icons in the video query are sorted according to their R_x values in the rank. If the R_x values of two adjacent icons are the same, an appositional relationship " \equiv " is inserted between them (as shown in line 17). Otherwise, an adjacent relationship " $|_n$ " is inserted (as shown in line 13), where n is the difference between their R_x values. Similarly, this algorithm can be applied in the y-axis or t-axis by changing the R_x values to R_y or R_t values, respectively.

Algorithm V.2. (*Build_3D_String*($Q, (X, Y, T)$))
 /* input : a video query Q */
 /* output : a normal 3D-String (X, Y, T) */
 0: begin
 1: *Build_1D_String*(Q, X)
 2: *Build_1D_String*(Q, Y)
 3: *Build_1D_String*(Q, T)
 4: end

Algorithm V.2 constructs a normal 3D-String from a video query by applying Algorithm V.1 in the x-axis, y-axis, and t-axis.

After a normal 3D-String is constructed from a video query, the next step is to change the spatial-temporal relationships in the normal 3D-String according to the query type specified by the user. This can be done by applying the following two algorithms.

Algorithm V.3. (*Transform_1D_String*($X, Q\text{-type}$))
 /* input : a normal 1D-String $X=I_1\alpha_1I_2\alpha_2I_3\ldots\alpha_{k-1}I_k$
 and the query type $Q\text{-type}$ */
 /* output : an extended 1D-String $X'=I_1\beta_1I_2\beta_2I_3\ldots\beta_{k-1}I_k$ */
 0: begin
 1: begin case
 2: case $Q\text{-type} = 0$
 3: for $i=1$ to $k-1$
 4: $\alpha_i \leftarrow \text{"?"}$
 5: case $Q\text{-type}=1$
 6: for $i=1$ to $k-1$
 7: if $\alpha_i = \text{"|}_n\text{"}$

Table 2. Transformation Rules for Normal 1D-Strings

Normal 1D-String	Type 0 Extended 1D-String	Type 1 Extended 1D-String	Type 2 Extended 1D-String
$A _n B$	$A ? B$	$A \Rightarrow B$	$A _n B$
$A \equiv B$	$A ? B$	$A \equiv B$	$A \equiv B$

```

8:           $\alpha_i \leftarrow \text{"}\Rightarrow\text{"}$ 
9:      end if
10:  end case
11:   $X' \leftarrow X$ 
12:  return  $X'$ 
13: end

```

Algorithm V.4. (*Transform_3D_String*((X, Y, T), $Q\text{-type}$))

/* input : a normal 3D-String (X, Y, T) and the query type $Q\text{-type}$ */
 /* output : an extended 3D-String (X', Y', T') */
 0: begin
 1: $X' \leftarrow \text{Transform_1D_String}(X, Q\text{-type})$
 2: $Y' \leftarrow \text{Transform_1D_String}(Y, Q\text{-type})$
 3: $T' \leftarrow \text{Transform_1D_String}(T, Q\text{-type})$
 4: return (X', Y', T')
 5: end

Algorithm V.4 constructs an extended 3D-String from a normal 3D-String by applying Algorithm V.3 in the x-axis, y-axis, and t-axis. Table 2 summarizes the transformation of the spatial-temporal relationships used in Algorithm V.3.

4. Video Query Processing

To check whether the symbol objects of a video object and the spatial and temporal relationships between them satisfy a video query, we introduce a data structure called the 3D-List and its related algorithms. The formal definition of the 3D-List and the details of these algorithms can be found in Liu and Chen (1998).

VI. Other Approaches to Content-Based Video Data Retrieval

In addition to the three approaches discussed in the previous sections, other approaches have been proposed which are based on various video features.

1. Query by Keywords

As in the traditional alphanumeric database, keywords can be used to stored the semantic descrip-

tions of a video object. This approach was adopted by Gibbs *et al.* (1993), in which the keywords associated with video objects are stored as an attribute of the class *VideoValue*. Based on the object model, Oomoto and Tanaka (1993) considered a video object as a sequence of video frames and represented the content of a video object as a collection of attribute/value pairs which are attached to the video object.

2. Query by Algebraic Operators

Weiss *et al.* (1994, 1995) viewed a video object as a three-dimensional box and used algebraic operators such as *concatenation*, *union*, and *parallel*, to compose video objects into a complex video object. Users can specify temporal/spatial relationships between component video objects of a complex video object as the query predicates of a video query.

3. Query by Categories

Smoliar and Zhang (1994) modeled the contents of video objects in two ways. First, according to their topics, video objects are classified into classes, and these classes form a tree structure composed of topical categories. Users can browse video objects in categories of interest. Second, each video shot is represented as a movie icon, called a *micon*, which consists of a volume of pixels. By employing a horizontal or vertical slice on the *micon*, the movement of a symbol in the video object can be traced.

4. Query by Intervals

The OVID system (Oomoto and Tanaka, 1993) uses an interval as the basic unit of video objects. An interval is associated with an object identifier and a set of attribute/value pairs, which describe its meanings. Users can retrieve video objects based on a description in attribute/value form. They also proposed a video query language *VideoSQL*, in which a new inheritance mechanism based on the interval inclusion relationship between video objects is applied to enable users to specify video queries.

5. Query by Spatial/Temporal Relationships of Video Objects

In Little and Ghafoor (1993), a set of temporal operators was designed for video queries. However, the temporal relationships can be evaluated between frame sequences only. Temporal relationships of content objects are not considered. Lee and Kuo (1993) considered sixteen primitive types of motion

for specifying the tracks of content objects in queries. However, the spatial relationships between content objects were not considered. In Yu and Wolf (1997), the spatial/temporal semantics of video data were studied. The Conceptual Spatial Object (CSO), Conceptual Temporal Object (CTO), Physical Object (PO) and a set of predicate logics were defined to express queries. Since spatial and temporal semantics are only captured by CSOs and CTOs, semantics that are not defined in CSOs and CTOs cannot be applied in queries.

6. Query by Motion Tracks of Symbol Objects

Since video objects have temporal characteristics, the appearance of a symbol forms a three dimensional curve, called the *motion track* of the symbol. Using a graphical query tool, users can draw a 3D curve to retrieve the multimedia objects which contain symbols with similar motion tracks. For example, we can retrieve video objects that contain a spiraling eagle whose motion track is like the curve specified. Wai and Chen (1997) proposed an approach to approximate matching of the motion tracks of symbol objects.

VII. Conclusion

In this paper, we have surveyed important approaches to content-based multimedia data retrieval. First, for video queries based on cinematic structures, a mask matching approach has been proposed to detect shot changes in an MPEG coded video stream. It takes advantage of reference ratio variances of macroblocks between MPEG coded frames to detect shot changes. In this approach, the processing time is reduced by directly evaluating MPEG coded data. In the implementation, a function is employed to quantize the results into shot change probability values. The results have been illustrated and discussed. Moreover, a conversion function has been presented for elimination of misdetection and loss of detection of shot changes.

Second, techniques for video query by key frame have been presented. In this approach, a set of key frames is automatically extracted from the compressed video. A query image is given to retrieve the video which contains frames similar to the query image. The similarity measurement is based on a comparison of the histogram of dominant colors. Moreover, the average luminance value of the image is applied as a filter to avoid matching all key frames.

Third, we have discussed the 3D-string approach for content-based video data retrieval. Based on the well known representation of images, the 2D-String, we have defined the notion of a 3D-String for repre-

sentation of the spatial and temporal relationships between icons in a video query. Based on 3D-Strings, the problem of video query processing is transformed into a problem of three-dimensional pattern matching. Since the string matching algorithms proposed in the past cannot solve this problem, the 3D-List data structure and its related algorithms have been proposed. There are two major techniques involved in the proposed algorithms.

Content-based video data retrieval is an active research area. Although fruitful research results have been reported, there are many challenging problems which need to be solved.

First, based on the relationships between macroblocks, the technique for content object detection can be extended. Using this technique, content objects as well as their spatial and temporal information can be detected. Video indexes for supporting content-based queries can, therefore, be automatically and efficiently constructed.

Second, integration of various approaches to content-based video data retrieval is necessary to provide users with a uniform interface to access video data. Data structures which can provide efficient access to various video features need to be developed.

Third, for video retrieval and query processing, the integration of video databases and generic object databases must be considered. In the integrated model, not only can video data be integrated the data of generic objects, but generic objects can also be integrated with the content objects in a video. The process of object identification, integration, and execution plan derivation between two databases needs to be investigated with the goal of providing more information to users.

References

- Adjeroh, D. A. and M. C. Lee (1997) Adaptive transform domain video scene analysis. *Proc. of IEEE Multimedia Computing and Systems*, pp. 203-210. Ottawa, Ontario, Canada.
- Aho, A. V. and M. J. Corasick (1979) Efficient string matching: an aid to bibliographic search. *Comm. ACM*, **18**(2), 333-340.
- Allen, J. F. (1983) Maintaining knowledge about temporal intervals. *Comm. ACM*, **26**(11), 832-842.
- Arman, F., A. Hsu, and M. Y. Chiu (1993) Image processing on compressed data for large video databases. *Proc. of First ACM Int'l Conf. on Multimedia*, pp. 251-258. Anaheim, CA, U.S.A.
- Baeza-Yates, R. and G. H. Gonnet (1992) A new approach to text searching. *Comm. ACM*, **35**(10), 74-82.
- Baker, T. P. (1978) A technique for extending rapid exact-match string matching to arrays of more than one dimension. *SIAM J. Comput.*, **7**(11), 533-541.
- Bakhmutova, I. V., V. D. Gusev, and T. N. Titkova (1997) The search for adaptations in song melodies. *Computer Music Journal*, **21**(1), 58-67.
- Balaban, M. (1996) The music structures approach to knowledge representation for music processing. *Computer Music Journal*, **20**(2), 96-111.
- Bird, R. S. (1977) Two dimensional pattern matching. *Information Processing Letters*, **6**(5), 168-170.
- Boyer, R. S. and J. S. Moore (1977) A fast string searching algorithm. *Comm. ACM*, **20**(10), 762-772.
- Brown, M. G. (1995) Automatic content-based retrieval of broadcast news. *Proc. of ACM Third Intl. Conf. on Multimedia*, pp. 35-43.
- Chang, C. W. and S. Y. Lee (1995) Statistical and topological feature extraction and matching in video sequences. *Proc. of National Computer Symposium*, pp. 693-700.
- Chang, S. F. (1997) Compressed-domain techniques for image/video indexing and manipulation. *Proc. of IEEE Int'l. Conf. on Circuits and Systems*.
- Chang, S. K., Q. Y. Shi, and C. W. Yan (1987) Iconic indexing by 2-D strings. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **PAMI-9**(3), 413-428.
- Chang, S. K., C. W. Yan, D. C. Dimitroff, and T. Arndt (1988) An intelligent image database system. *IEEE Trans. on Software Eng.*, **14**(5), 681-688.
- Chen, A. L. P., C. C. Liu, K. L. Lee, and C. C. Chen (1995) The design of a video database system. *Proc. Real-Time and Media Systems*.
- Chen, J. C. C. and A. L. P. Chen (1998) Query by rhythm: an approach for song retrieval in music databases. *Proc. IEEE Eighth International Workshop on Research Issues in Data Engineering: Continuous-Media Databases and Applications*.
- Chiueh, T. C. (1994) Content-based image indexing. *Proc. of the 20th International Conference on VLDB*, pp. 582-593.
- Chou, T. C., A. L. P. Chen, and C. C. Liu (1996) Music databases: indexing techniques and implementation. *Proc. IEEE Intl. Workshop on Multimedia Data Base Management Systems*.
- Chua, T. S. and L. Q. Ruan (1995) A video retrieval and sequencing system. *ACM Trans. on Information Systems*, **13**(4), 373-407.
- Davenport, G., T. A. Smith, and N. Pinciver (1991) Cinematic primitives for multimedia. *IEEE Computer Graphics & Applications*, 67-74.
- Dimitrova, N. and F. Golshani (1995) Motion recovery for video content classification. *ACM Trans. on Info. Sys.*, **13**(4), 408-439.
- Fan, J. J. and K. Y. Su (1993) An efficient algorithm for matching multiple patterns. *IEEE Trans. on Knowledge and Data Engineering*, **5**(12), 339-351.
- Gall, D. Le (1991) MPEG: a video compression standard for multimedia applications. *Communications of ACM*, **34**(4), 46-58.
- Ghias, A., J. Logan, D. Chamberlin, and B. C. Smith (1995) Query by humming: musical information retrieval in an audio database. *Proc. of ACM Multimedia*, pp. 231-236. San Francisco, CA, U.S.A.
- Gibbs, S., C. Breiteneder, and D. Tsichritzis (1993) Audio/video database: an object-oriented approach. *Proc. on Int'l. Conf. on Data Eng.*, pp. 381-390.
- Gudivada, V. N. and V. V. Raghavan (1995) Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Trans. on Info. Sys.*, **13**(2), 115-144.
- Jagadish, H. V. (1991) A retrieval technique for similar shapes. *Proc. of ACM SIGMOD Conf. on Management of Data*, pp. 208-217. Denver, CO, U.S.A.
- Kuo, T. C. T. and A. L. P. Chen (1996) Indexing, query interface and query processing for Venus: a video database system. *Proc. of Cooperative Databases for Advance Applications*.
- Kuo, T. C. T., Y. B. Lin, and A. L. P. Chen (1996) Efficient shot change detection on compressed video data. *Proc. of IEEE Workshop on Multimedia Database Management Systems*, pp. 101-108.
- Lee, S. Y. and H. M. Kuo (1993) Video indexing: an approach based

- on moving object and track. *Proc. of SPIE - The Int'l Society for Optical Engineering*, pp. 25-36.
- Lin, Y. B. (1996) *An Efficient Method to Build Video Indexes from Compressed Data*. M.S. Thesis. Computer Science of National Tsing Hua Univ., Hsinchu, Taiwan, R.O.C..
- Little, T. D. C. and A. Ghafoor (1993) Interval-based conceptual models for time-dependent multimedia data. *IEEE Transaction on Knowledge and Data Engineering*, pp. 551-563.
- Liu, C. C. and A. L. P. Chen (1996a) Extending object data model for representing the spatial-temporal relationships and constraints of multimedia data. *Proc. 7th Workshop on Object-Oriented Technology and Applications*, pp. 272-277.
- Liu, C. C. and A. L. P. Chen (1996b) Modeling and query processing of distributed multimedia databases. *Proc. Real-Time and Media Systems*.
- Liu, C. C. and A. L. P. Chen (1997) Vega: a multimedia database system supporting content-based retrieval. *Journal of Information Science and Engineering*, **13**(3), 369-398.
- Liu, C. C. and A. L. P. Chen (1999) 3D-List: a data structure for efficient video query processing. *IEEE Trans. on Knowledge and Data Engineering* (accepted).
- Liu, C. C., A. J. L. Hsu, and A. L. P. Chen (1999) Efficient near neighbor searching using multiple indexes for content-based multimedia retrieval. *Multimedia Tools and Systems Journal* (accepted).
- Meng, I., Y. Juan, and S. F. Chang (1995) Scene change detection in a MPEG compressed video sequence. *SPIE Symposium on Electronic Imaging Science & Technology-Digital Video Compression: Algorithms and Technologies*.
- Myron, F., H. Sawhny, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker (1995) Query by image and video content: the QBIC system. *IEEE Computer Magazine*, **28**(9), 23-32.
- Nagasaka, A. and Y. Tanaka (1991) Automatic video indexing and full-video search for object appearances. *2nd Working Conference on Visual Database Systems*, pp. 119-133. Budapest, Hungary.
- Oomoto, E. and K. Tanaka (1993) OVID: Design and implementation of a video-object database system. *IEEE Trans. on Knowledge and Data Eng.*, **5**(4), 629-643.
- Otsuji, K. and Y. Tonomura (1993) Projection detecting filter for video cut detection. *Proc. of ACM Multimedia*, pp. 251-257. Anaheim, CA, U.S.A.
- Petrakis, E. G. M. and S. C. Orphanoudakis (1993) Methodology for the representation, indexing and retrieval of images by content. *Image and Vision Computing*, **11**(8), 504-521.
- Philips, M. and W. Wolf (1996) Video segmentation techniques for news. *Proc. of SPIE Photonic East*.
- Roussopoulos, N., C. Faloutsos, and T. Sellis (1988) An efficient pictorial database system for PSQL. *IEEE Trans. on Software Eng.*, **14**(5), 639-650.
- Smoliar, S. W. and H. J. Zhang (1994) Content-based video indexing and retrieval. *IEEE Multimedia*, **1**(2), 62-72.
- Swanberg, D., C. F. Shu, and R. Jain (1993) Knowledge guided parsing in video databases. *Proc. of SPIE Conference on Storage and Retrieval for Image and Video Database*, pp. 13-24.
- Tonomura, Y. and S. Abe (1990) Content oriented visual interface using video icons for visual database systems. *Journal of Visual Languages and Computing*, pp. 183-198.
- Tonomura, Y. and S. Abe (1994) Structured video computing. *IEEE Multimedia*, **1**(3), 34-43.
- Ueda, H., T. Miyatake, and S. Yoshizawa (1991) Impact: an interactive natural-motion-picture dedicated multimedia authoring system. *Proc. of Human Factors in Computing Systems (CHI91)*, pp. 343-354. New Orleans, LA, U.S.A.
- Wai, T. T. Y. and A. L. P. Chen (1997) Retrieving video data via motion tracks of content symbols. *Proc. ACM 6th Int'l. Conf. on Information and Knowledge Management*.
- Weiss, R., A. Duda, and D. K. Gifford (1994) Content-Based Access to Algebraic Video. *Proc. of the Int'l. Conf. on Multimedia Computing and Systems*, pp. 140-151.
- Weiss, R., A. Duda, and D. K. Gifford (1995) Composition and search with a video algebra. *IEEE Multimedia*, **2**(1), 12-25.
- Wold, E. T. Blum, D. Keislar, and J. Wheaton (1996) Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, **3**(3), 27-36.
- Yeung, M. M., B. L. Yeo, W. Wolf, and B. Liu (1995) Video browsing using clustering and scene transitions on compressed sequences. *IS&T/SPIE Symposium Proceedings*, pp. 399-413.
- Yoshitaka, A. (1994) Knowledge-assisted content-based retrieval for multimedia databases. *IEEE Multimedia*, **1**(4), 12-21.
- Yu, H. H. and W. Wolf (1997) A visual system for video and image databases. *Proc. of IEEE Multimedia Computing and Systems*, pp. 517-524. Ottawa, Ontario, Canada.
- Zhang, H. J., A. Kankanhalli, and S. W. Smoliar (1993) Automatic partitioning of video. *IEEE Multimedia System*, **1**(1), 10-28.

視訊資料的內涵式查詢

陳良弼 劉志俊 郭秋田

國立清華大學資訊工程學系

摘 要

由於全球資訊網、數位圖書館與隨選視訊的日益普遍，視訊資料庫成爲一項重要的研究課題。在本文中，我們綜合探討了各種不同的視訊資料內涵式查詢方法。若與傳統依照關鍵字查詢的方法比較，內涵式查詢可以提供使用者更豐富而強大的查詢能力。我們特別針對其中的三種方法進行深入的說明。第一，因爲視訊資料的分段技術是建立視訊索引所必備的基礎，我們首先探討如何由MPEG格式的視訊資料中自動地進行鏡景偵測以建立劇本結構的方法。第二，由於視訊資料是由一連串的影像所組成，我們可以藉由給定一張圖片來找尋所有包含與這張圖片相似的影像之視訊物件。我們將說明如何進行相似性比對的有關技術。最後我們將說明一種以視訊資料中的符號物件爲索引基礎的視訊查詢模式與查詢處理方法。我們將解釋三維字串的涵義及其比對方法。