

# 7. Video Indexing and Understanding

Michael S. Lew, Nicu Sebe, and Paul C. Gardner

## 7.1 Introduction

More and more video is generated every day. While today much of this data is produced and stored in analog form, the tendency is to use the digital form. The digital form allows processing of the video data in order to generate appropriate data abstractions that enable content-based retrieval of video. In the future, video databases will be able to be searched with combined text and visual queries. Additionally, video clips will be retrieved from longer sequences in large databases on the basis of the semantic video content. Ideally, the video will also be automatically annotated as a result of the machine interpretation of the semantic content of the video.

Rowe et al. [32] characterized the types of video queries a user may ask and identified the following three types of metadata indexes that should be associated with the video data in order to satisfy a query:

**Bibliographic data.** This category includes information about the video (e.g., title, abstract, subject, and genre) and the individuals involved in the video (e.g., producer, director, and cast).

**Structural data.** Video and movies can be described by a hierarchy of movie, segment, scene, and shot where each entry in the hierarchy is composed of one or more entries at a lower level (e.g., a segment is composed of a sequence of scenes and a scene is composed of a sequence of shots [9]).

**Content data.** Users want to retrieve videos based on their content (audio and visual content). In addition, because of the nature of video the visual content is a combination of static content (frames) and dynamic content. Thus, the content indexes may be: (1) sets of keyframes that represent key images (e.g., frames that show each actor in the video or a sequence of images that depict the major segments and scenes in the video); (2) keyword indexes built from the soundtrack, and (3) object indexes that indicate entry and exit frames for each appearance of a significant object or individual.

Some of the current research is addressing the problem of manual video annotation [9, 19] and the types of annotation symbols needed. These annotations are added by an individual, often with the assistance of user interfaces. The problem is that when databases of video are involved, video processing to automatically extract content information may be crucial.

The interactive query process consists of three basic steps: formulating the query, processing the query, and viewing the results returned for a query. This requires an expressive method of conveying what is desired, the ability to match what is expressed with what is present, and ways to evaluate the outcome of the search. Conventional text-based query methods which rely on keyword lookup and string pattern matching are not adequate for all types of data, particularly not for auditory and visual data. Therefore, it is not reasonable to assume that all types of multimedia data can be sufficiently described with words alone, nor as metadata when it is first entered in the database, nor as queries when it is retrieved.

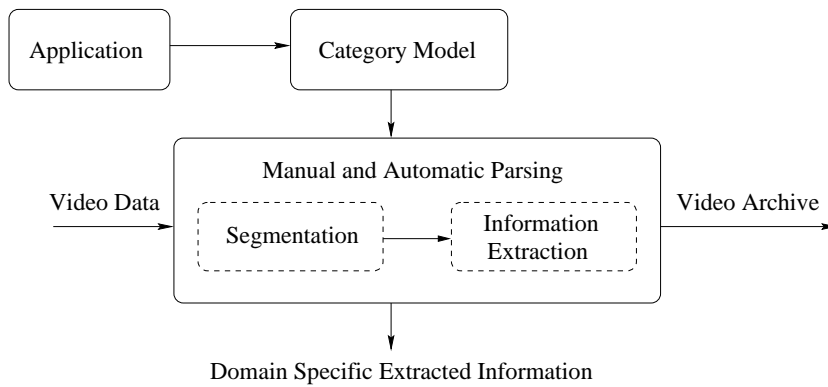
### 7.1.1 Video Query Formulation

In general, a video query is more complicated than a traditional query of text databases. In addition to text (manual annotations and closed captions) a video clip has visual and audio information. A primary question is how a query can be formulated across these multiple information modalities. Text can be used to formulate queries for visual data (image, video, graphs), but such queries are not very efficient and cannot encompass the hierarchical, semantic, spatial, and motion information. Describing visual data by text can be difficult because keywords are chosen by each user based on his subjective impression of the image and video. Thus, it is difficult to know which keywords were originally used when the target video shot was indexed. In addition, keywords must be entered manually, they are time consuming, error prone, and thus they may have prohibitive costs for large databases. For these reasons, visual methods should be considered in query formulation. A query system which allows retrieval and evaluation of multimedia data should be highly interactive to facilitate easy construction and refinement of queries.

The user can formulate diverse types of queries, which may require processing of multiple modalities. For example, a user may have seen a particular piece of video and wants to retrieve it for viewing or reusing. Another user may be only looking for a specific video without having seen it. Finally, a user may have only some vague idea of what he is looking for. Additionally, the system should accommodate queries of a user who just wants to browse the videos without having a specific goal. Ideally, the query formulation process should accommodate all these types of queries [6]. In order to achieve these goals, the different information modalities should be fully used and exploited in the query formulation and search process.

### 7.1.2 Video Categorization

In this stage the user or the search engine decides which category of video is to be searched (see Fig. 7.1). Categorization is the capability to use metadata, for example to direct the search to a specific topic. A form of classifying



**Fig. 7.1** Video data processing.

video on the basis of applications and purpose is discussed in Refs [14, 19]. The nature of the information captured and the purpose for which the video is used greatly affects the structure of the video. Four main purpose-based classes are extracted:

**Entertainment.** The information presented in this class is highly stylized depending on the particular sub-category: fiction (motion pictures, TV programs), nonfiction (sports, documentaries), and interactive video (games).

**Information.** The purpose of videos here is to convey information to the viewer (news).

**Communication.** The purpose of videos in this category is communication of information (video conferences).

**Data analysis.** Scientific video recording, like medical and psychology experiments may be used for data analysis purposes.

Such a classification of the video could create problems for certain videos, for example the ones presenting news about a sporting event. These videos will have to be placed into multiple categories (e.g., entertainment, information) or the user will have to select more than one category to search on.

In general, each video has to be classified using a combination of available manual textual annotation and visual and audio content. Considering that most of the current videos have some text associated with them, it is quite reasonable that the initial categorization is formulated in textual form.

### 7.1.3 Searching

Suppose that we are searching for a particular video shot. The result of the search is a list of candidate units that satisfy the constraints of the query. The

ultimate goal of this stage is to minimize the number of candidates without missing the video(s) of interest.

The categorization stage limits to some extent the scope of data to be searched but there is still a need for efficient and effective means of further filtering the data of interest. Data-object models are used for organizing data in databases in order to allow fast retrieval. In video, the data objects developed have to cope with the various information modalities such as text, audio, and visual content.

In order to make the search possible, information contained in the video needs to be extracted (Fig. 7.1). This can be done by parsing the data automatically, manually, or a combination of the two (hybrid). Automatic extraction depends heavily on techniques used in computer vision. Content of unstructured data such as imagery or sound is easily identified by human observation. However, few attributes lead to reliable machine identification. Therefore, we have to rely more on hybrid extraction techniques.

Text-based search could be a first step in the searching stage. It serves as a good search filter and is an extremely important search tool that should not be neglected in video-retrieval problems. The text descriptors associated with the video should describe the content sufficiently to help the users to locate segments of interest or, at least, not exclude potential segments from the candidate list. These descriptors, such as the date of creation, the title, and the source, are commonly available at the time of creation. On the other hand, attributes such as the number of shots, audiovisual events such as dialogs and music, images, etc., can be derived from automatic analysis of the video.

Audiovisual attributes beyond the text are important and augment the text-query process. The difficult problem is to define audiovisual properties which can be extracted from the video and will reduce the size of the candidate list.

Query by image content techniques have been proposed in recent years. Here the queries are formulated using sample images or videos, rough sketches, or component features of an image (outline of objects, color, texture, shape, layout). In the QBIC [12] system, thumbnail images are stored in the database along with text information. Object identification is performed based on the images and the objects. Features describing their color, texture, shape, and layout are computed and stored. Queries can consist of individual or multiple features which are compared to the stored database features. The user can also use objects in the queries by drawing outlines around the object in an image. Another approach is to use iconic search [23, 32]. These systems take advantage of a user's familiarity with the world. An icon represents an entity/object in the world, and users can easily recognize the object from the icon image. The query is formulated by selecting the icons that are arranged in relational or hierarchical classes. A user refines a query by traversing through these classes. Iconic queries reduce the flexibility in a query

formulation as queries can only utilize the icons provided, i.e., iconic databases tend to be rigid in their structure. There is also active ongoing research in audio indexing, to extract special audio features.

Another aspect of video search is that a user may not be interested in entire clips, but rather in portions of the clips. Thus, the search should be able to return as a result data objects (video portions) consisting of relevant footage. Furthermore, this implies that the data objects should be built in a hierarchical manner. A meaningful hierarchy is described in the film books: a shot is the fundamental unit, a scene is a collection of shots unified by the same dramatic incident, a segment is a collection of scenes, and a movie or clip is a collection of segments. Consequently, the attributes of a clip should be designed to include the most general features (attributes) which can be inherited by the segments, scenes, and shots it contains.

To enable video search, the video clips have to be properly segmented into semantic units. Automatic analysis of video content can achieve meaningful segmentation, provide valuable characterization of video, and offer features that are depictable with words.

#### 7.1.4 Browsing

The result of the search stage is a collection of video candidates, of which the total duration can be long. Therefore, all candidates from the list should be browsable. This means that high-level representations of the content of the candidate videos should be displayed. A user, by looking at these representations, can quickly understand the video content and browse through many videos in a short time. On the other hand, the user should have random access to any point of any video and should be able to get an overview of each candidate video by viewing only a small area of a computer-display screen. As a consequence, high-level representations of video, in the form of visual summaries are necessary.

The area of visual summaries is very important in the context of the video query process. The problem is to derive or compute a mapping from the video data, considered as a three-dimensional object (the three dimensions are the two geometrical dimensions of the individual frames and the time dimension), to the two-dimensional plane for screen representation. The video structure represented by the video summary should be highly correlated with the semantic content, and allow simple semantic interpretation for sufficient understanding. This involves automatically finding the semantically most important dramatic elements within a video clip.

The development of visual summaries depends heavily on the video category. It is obvious that story structures are prominent in news or films, while programs of sporting events do not have such story structure. Therefore, different video categories will require different forms of visual summaries. For example, the VISION [13] system allows users to dynamically filter the contents of the category-based browsing structures using multiple criteria

(source, date, and keywords). This provides a hybrid of browsing and searching.

### 7.1.5 Viewing

After one or more candidate videos are selected as more likely, the user has to be able to see parts of the candidate. The usual functions of today's video players should be available in this stage: play, pause, fast-forward, reverse, etc. More sophisticated capabilities like semantic fast-forward (the ability to move forward in the video on the basis of semantic video content) should also be available.

The huge amount of data present in the video requires compression in order to achieve efficient storage and transmission. One of the most common standard video compression formats is MPEG which uses predictive and differential coding techniques. This implies that random access to any frame of the video cannot be implemented in a straightforward way. Therefore, efficient algorithms should be implemented for manipulating MPEG videos without the need of a complete decompression.

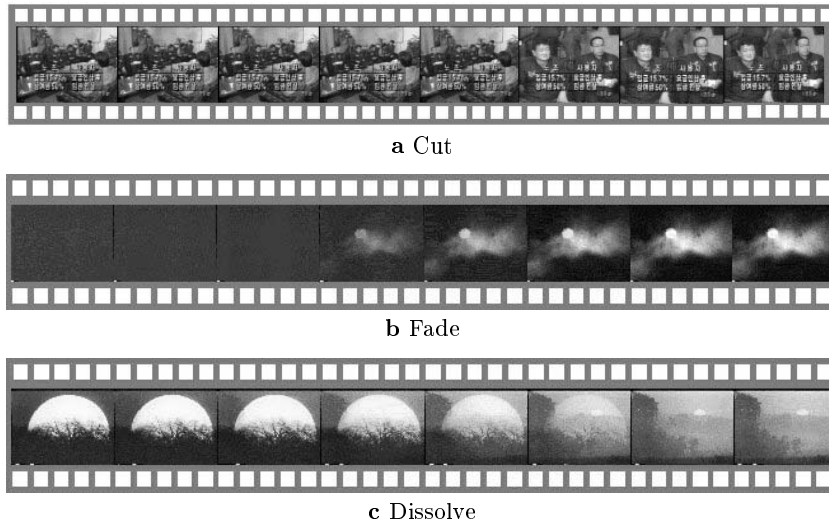
## 7.2 Video Analysis and Processing

Most of the videos (movies, TV programs, documentaries, etc.) are structured such that they convey the underlying narrations and messages. This means that the shots are combined in a specific way and according to a special orders in order to form the montage in telling the story. Because of this, certain temporal features can be recognized and associated information can be extracted by automatically analyzing the visual contents and temporal associations of the video.

### 7.2.1 Video Shots

The fundamental unit of a film and video is a shot. The content of a shot depicts continuous action captured by a single camera during some interval. The importance of the shot as the fundamental video unit has been realized by many researchers, and the computational detection of video-shot boundaries has received much attention. Usually, two types of scene changes are considered: scene cuts (or camera breaks) and gradual transitions. An *abrupt transition* or *cut* (Fig. 7.2(a)) is an instantaneous change from one shot to another where the transition boundary is between two consecutive frames. *Gradual* scene changes are frequently used for editing techniques to connect two scenes together and can be classified into three common types: fade-in/out, dissolve, and wipe. A *fade* is a continuous, monotonic change in luminance (Fig. 7.2(b)). A *dissolve* can be viewed as a fade-out and fade-in

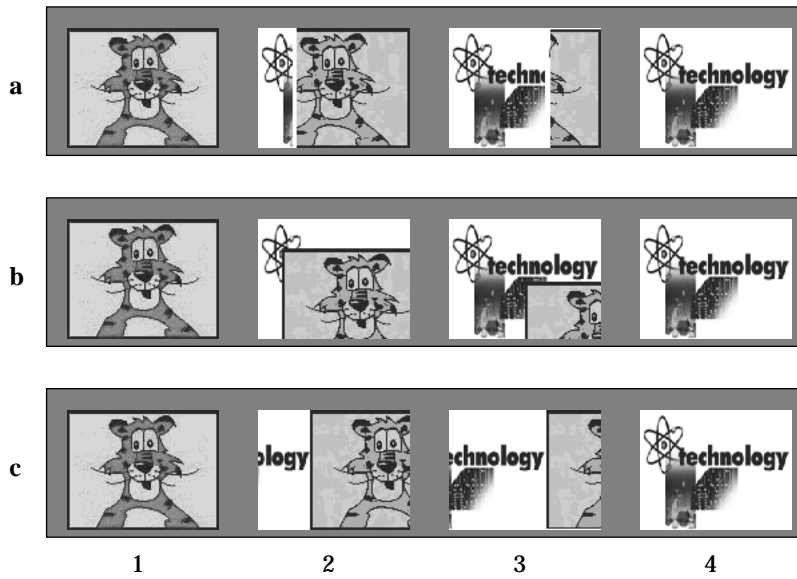
with some overlap (Fig. 7.2(c)). A dissolve is more general than a fade because the chrominance can also change during the transition and each pixel can change independently. A *wipe* is a transition from one scene to another wherein the new scene is revealed by a moving boundary in the form of a line or pattern. Figure 7.3 shows some examples of wipes.



**Fig. 7.2** Examples of shot transitions.

Fundamentally, most approaches use the concept of partitioning the continuous video data into sequences for indexing. Videos are segmented primarily on the basis of camera breaks or shots, therefore, each sequence is a segment of data having a frame or multiple consecutive frames. Automatic isolation of camera transitions requires support of tools that provide accurate and fast detection. Abrupt camera transitions can be detected quite easily as the difference between two consecutive frame is so large that they cannot belong to the same shot. A problem arises when the transition is gradual, meaning that the shot does not change abruptly but just over a period of few frames. In this case, the difference between two shots is not large enough to declare it a camera break.

Recently, many video segmentation algorithms have been proposed. Ahanger and Little [3], Idris and Panchanathan [17], and Mandal et al. [25] have presented detailed surveys on some of the existing video segmentation methods. A large number of techniques have been reported in the literature for temporal segmentation. A number of metrics have been suggested for video scene segmentation for both the raw and compressed data. The metrics used to detect the difference between two frames can be divided broadly into



**Fig. 7.3** Wipe examples. **a** piecewise replacement, neither images moves; **b** old image moves out, revealing the new one; **c** both images move, similar to panning.

four classes: (1) pixel or block comparison; (2) histogram comparison (of gray levels or color codes); (3) using the DCT coefficients in MPEG encoded video sequences; and (4) the sub-band feature comparison method.

Consider a video stream as a sequence of video frames  $\{f_1, f_2, \dots, f_N\}$ , where  $N$  is the total number of frames and each  $f_i$ ,  $1 \leq i \leq N$ , is a 24-bit RGB color image with  $X \times Y$  pixels. Different image features such as intensity, color histogram, etc., extracted from each video frame are usually used to compute a quantitative measure of the content difference between two frames. We denote the inter-frame difference (or frame difference) between consecutive frames  $f_i$  and  $f_{i+1}$  by  $D_i(f_i, f_{i+1})$  for  $1 \leq i \leq N - 1$ . In the following subsections, we describe some of the existing methods for computing the content difference for consecutive video frames.

**Pixel Difference Method.** Nagasaka and Tanaka [28] use the sum of absolute pixel-wise intensity differences between two frames as a frame difference. Let  $f_i(x, y)$  denote the intensity of pixel  $(x, y)$  in a frame  $f_i$ . The difference between frame  $f_i$  and frame  $f_{i+1}$  is defined as:

$$D_i(f_i, f_{i+1}) = \frac{1}{X \cdot Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} |f_i(x, y) - f_{i+1}(x, y)|. \quad (7.1)$$

Otsuji et al. [29] and Zhang et al. [49] count the number of the changed pixels, and a camera shot break is declared if the percentage of the total

number of changed pixels exceeds a threshold. Mathematically, the differences in pixels and threshold are

$$DP_i(x, y) = \begin{cases} 1 & \text{if } |f_i(x, y) - f_{i+1}(x, y)| > t, \\ 0 & \text{otherwise,} \end{cases} \quad (7.2)$$

$$\frac{100}{X \cdot Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} DP_i(x, y) > T. \quad (7.3)$$

If the difference between the corresponding pixels in the two consecutive frame is above a certain minimum intensity value  $t$ , then  $DP_i(x, y)$  is set to one. In Eq. 7.3, the difference percentage between the pixels in the two frames is calculated by summing the number of different pixels and dividing by the total number of pixels in a frame. If this percentage is above a certain threshold  $T$ , a camera break is declared.

Camera movement, e.g., pan or zoom, can change a large number of pixels and hence in such conditions a false shot break will be detected. Fast moving objects also have the same effect. Zhang et al. [49] propose the use of a smoothing filter (e.g.,  $3 \times 3$  window) to reduce the sensitivity to camera movement. To make cut detection more robust to illumination changes, Aigrain and Joly [4] normalize the video frame through histogram equalization.

**Likelihood Ratio.** Detecting changes at pixel level is not a very robust approach. A *likelihood ratio* approach is suggested based on the assumption of uniform second order statistics over a region [49]. The frames are first divided into blocks and then the blocks are compared on the basis of the statistical characteristics of their intensity blocks. Let  $\mu_i$  and  $\mu_{i+1}$  be the mean intensity values for a given region in two consecutive frames, and  $\sigma_i$  and  $\sigma_{i+1}$  be the corresponding variances. The frame difference is defined as the percentage of the regions whose likelihood ratios (Eq. 7.4) exceed a pre-defined threshold  $t$ :

$$\lambda = \frac{[\frac{\sigma_i + \sigma_{i+1}}{2} + (\frac{\mu_i + \mu_{i+1}}{2})^2]^2}{\sigma_i \cdot \sigma_{i+1}}, \quad (7.4)$$

$$DP_i(x, y) = \begin{cases} 1 & \text{if } \lambda > t, \\ 0 & \text{otherwise.} \end{cases} \quad (7.5)$$

This approach is better than the previous one as it increases the tolerance against noise associated with camera and object movement. However, it is possible that even though the two corresponding blocks are different, they can have the same density function. In such cases no change is detected.

Shahraray [36] proposed dividing a frame into  $K$  non-overlapping blocks and finding the “best” matching blocks between frames for comparison, similar to the block matching technique of MPEG. Typically,  $K = 12$  is used and block matching is performed on the image intensity values. A non-linear digital order-statistic filter *L-filter* is used in computing the block differences. Let  $L = \{l_1, l_2, \dots, l_K\}$  be an ordered list of the  $K$  block match values. Then the match coefficient between the two images is:

$$D(f_i, f_{i+1}) = \sum_{i=1}^K c_i l_i \quad (7.6)$$

where  $c_i$  is a predetermined coefficient whose value depends on the order of the match values  $l_i$  in the list  $L$ .

**Histogram Comparison.** The sensitivity to camera and object motion can be further reduced by comparing the gray-level histograms of the two frames [28, 49]. This is due to the fact that two frames, which have minimal changes between backgrounds and some amount of object motion have almost the same histograms.

Let  $H_i$  be the gray-level histogram of a frame  $f_i$ , then the histogram difference between two consecutive frames is defined as:

$$D_i(f_i, f_{i+1}) = \sum_{k=0}^{G-1} |H_i(k) - H_{i+1}(k)| \quad (7.7)$$

where  $G$  is the number of gray levels. If the sum is greater than a given threshold  $t$  a transition is declared.

Histogram comparison using color code is suggested in Refs [28, 49]. The authors use a 6-bit color code of a 24-bit RGB pixel obtained by combining the two most significant bits extracted from each of the 8-bit red, green, and blue color components. Let  $HC_i$  be the histogram of the color codes in a frame  $f_i$  then, the frame difference is defined as:

$$D_i(f_i, f_{i+1}) = \sum_{k=0}^{63} |HC_i(k) - HC_{i+1}(k)|. \quad (7.8)$$

Nagasaka and Tanaka [28] propose using the  $\chi^2$  test as a measure of similarity between color histograms. It computes the square of the difference between the two histograms in order to strongly reflect the difference. Let  $HC_i$  be the color histogram of a frame  $f_i$  having  $G$  bins then, the frame difference is defined as:

$$D_i(f_i, f_{i+1}) = \sum_{k=0}^{G-1} \frac{|HC_i(k) - HC_{i+1}(k)|^2}{H_{i+1}(k)}. \quad (7.9)$$

The  $\chi^2$ -test enhances the difference between the camera breaks but also small changes due to camera or object motion. Therefore, this method may not be more effective than the gray and color histogram comparison techniques.

They also propose a robust approach to accommodate momentary noise such as a the light from a flashlight, which only brightens part of the frame. The video frame is divided into  $4 \times 4$  rectangular regions, each denoted by a region number  $r$ . Histogram comparison is made on each pair of corresponding regions between two frames. The frame difference is defined as the sum of the eight smallest sub-frame histogram differences calculated using Eq. 7.9 for each  $0 \leq r \leq 15$ .

**Twin Comparison.** The previous techniques are based on a single threshold and lack the power of detecting gradual transitions. This is because the threshold is higher than the difference found between the frames in which the transition takes place due to special effects. Lowering the threshold does not solve the problem because the difference value due to the special effects can be smaller than the ones which take place within the shot. For example, object motion and/or camera motion might contribute more changes than the gradual transition. Making the value of the threshold even smaller will lead to false detections due to camera and object motions. When the beginning and end frames for the gradual transitions are detected, the frames in the transition can be declared as a separate segment.

The twin comparison method [49] takes into account the cumulative differences between the frames for gradual transitions. This method (Fig. 7.4) requires two cutoff thresholds, one higher threshold ( $T_h$ ) for detecting abrupt transitions and a lower one ( $T_l$ ) for gradual transitions. In the first stage, the higher threshold is employed to detect abrupt shot boundaries. In the next stage, the lower threshold is used on the rest of the frames. Any frame that has a difference more than this threshold is declared as a potential start ( $F_s$ ) of the transition. Once  $F_s$  is identified, it is compared with subsequent frames measuring the accumulated difference instead of frame-to-frame difference. Usually, this difference value increases and when this value increases to the level of the higher threshold, a camera break is declared at that frame ( $F_e$ ). If the value falls between the consecutive frames, then the potential frame is dropped and the search starts all over. Although the twin comparison approach is effective in detecting gradual transitions, the type of transition cannot be identified.

The gradual transitions might include special effects due to camera panning and zooming. Optical flow is used to detect camera motions and motion vectors are computed to detect the changes due to pan and zoom.

**Model-Based Segmentation.** It is not only important to identify the gradual transition position, but also the type of transition. In a video sequence, a gradual transition from one scene to another is the result of the editing

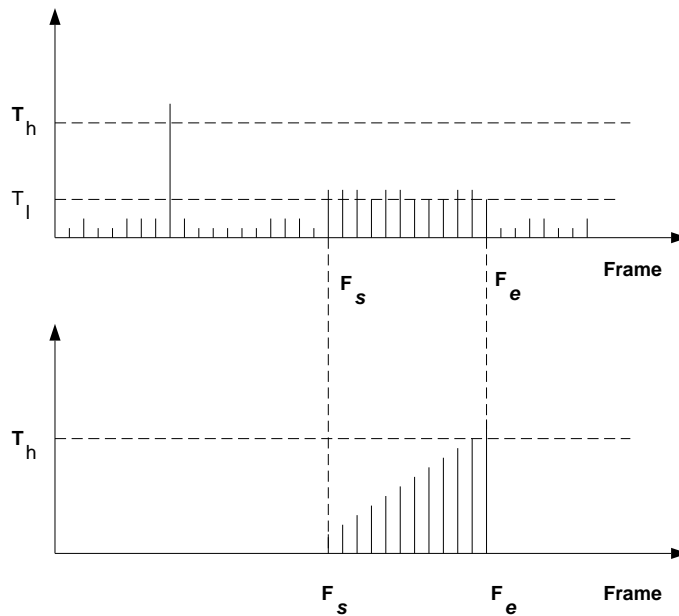


Fig. 7.4 Twin comparison.

process. Therefore, in model-based techniques the problem of video segmentation is viewed as locating the edit boundaries within the video sequence. Here, different edit types, such as cuts, translations, fades, wipes, and dissolves are modeled by mathematical functions.

Let the symbol  $S$  denote a single continuous shot which is a set of consecutive 2D images. The individual images of the set are denoted  $I(x, y, t)$ , where  $x$  and  $y$  are the pixel position and  $t$  is the discrete time index. A shot containing  $N$  images is represented as follows:

$$S = \{I(x, y, 1), I(x, y, 2), \dots, I(x, y, N)\}. \quad (7.10)$$

Abrupt cuts are formed by concatenating two different sequences as  $S = S_1 \circ S_2$ . They are the easiest to detect since the two sequences  $S_1$  and  $S_2$  are usually dissimilar. Due to the abrupt nature of this transition, it is expected that it will produce significant changes in the scene lighting and geometric structure.

Two types of fades are considered. The first is a fade-out, where the luminance of the scene is decreased over multiple frames, and the second is a fade-in, where the luminance of the scene is increased from some base level, usually black, to full scene luminance. A simple way to model a fade-out is to take a single frame in the scene,  $I(x, y, t_1)$ , and gradually decrease the luminance. One way to do this is by scaling each frame in an  $N$  frame edit sequence as:

$$S(x, y, l) = I(x, y, t_1) \times \left(1 - \frac{l}{N}\right), \quad l = 1, \dots, N \quad (7.11)$$

The shape of the intensity histogram remain fixed (ideally) for each frame in the sequence, but the width of the histogram is scaled by the multiplicative factor  $(1 - \frac{l}{N})$ . The intensity mean, median, and standard deviation values in each frame are also scaled by this factor (relative to their value in frame  $t_1$ ). Another way to implement a fade-out is to shift the luminance level as:

$$S(x, y, l) = I(x, y, t_1) - m_1 \times \left(\frac{l}{N}\right), \quad l = 1, \dots, N \quad (7.12)$$

where  $m_1$  is the maximum intensity value in the frame  $I(x, y, t_1)$ . In this model, the mean and median intensity values are shifted downward in each consecutive frame, but the standard deviation remains constant. In practice, a non-linear limiting operation is applied to the sequence because intensity values are non-negative, and the resulting negative intensity values are set equal to 0. The limiting operation decreases the width of the histogram and likewise decreases the standard deviation. A general expression for the change in standard deviation cannot be determined because it depends on the shape of the histogram. If the limiting operation is applied to any of the inputs, the standard deviation will decrease, otherwise it will remain constant.

Two analogous models for fade-ins are:

$$S(x, y, l) = I(x, y, t_1) \times \left(\frac{l}{N}\right), \quad l = 1, \dots, N \quad (7.13)$$

and

$$S(x, y, l) = I(x, y, t_1) + m_N \times \left(\frac{l}{N}\right), \quad l = 1, \dots, N \quad (7.14)$$

where  $m_N$  is the maximum intensity value in the frame  $I(x, y, t_N)$ .

The scaling model of Eqs 7.11 and 7.13 was employed for detecting chromatic edits in video sequences [15] (a chromatic edit is achieved by manipulating the color or intensity space of the shots being edited). Experimental results indicate that some, but not all, fades follow this model. The shifting model of Eqs 7.12 and 7.14 is proposed as one alternative. Both fade models are too simple because they model a sequence which is a single static image whose brightness is varied. In reality, this operation is applied to non-static sequences where inter-frame changes due to object motion occur.

During a fade it is assumed that the geometric structure of the scene remains fairly constant between frames, but that the lighting distribution changes. For example, during a fade-out that obeys Eq. 7.13 the mean, median, and standard deviation all decrease at the same constant rate, but the content of the scene remains constant. If the fade obeys Eq. 7.14 the mean

and the median decrease at the same rate, but the standard deviation may not. The converse is true for fade-ins.

Dissolves are a combination of two scenes, and can simply be a combination of a fade-out of one shot ( $I_1$ ) and a simultaneous fade-in of another ( $I_2$ ). The model for this is [15]:

$$S(x, y, l) = I_1(x, y, t_1) \times \left(1 - \frac{l}{N}\right) + I_2(x, y, t_2) \times \left(\frac{l}{N}\right). \quad (7.15)$$

Dissolves are difficult to detect since they can occur gradually, and often do not obey a simple mathematical model such as Eq. 7.15. This is because the fade rates do not have to be equal as modeled. For example, there may be object/camera motion during the transition and complex special effects can be applied. There are also other common sources of problems in detecting fades and dissolves: relatively small areas of change, e.g., title sequences where the frame is mostly black with only a few lines of text that actually fade in or out; and persistent unchanging regions, e.g., overlay graphics and station identifiers/logos.

During a dissolve the scene structure and lighting characteristics may change simultaneously. In addition, the expected changes in the sequence are difficult to predict unless the final image in the sequence is known (i.e., the edit is already detected). However, if the dissolve obeys Eq. 7.15 the frames comprising the edit will be a linear combination of the initial and final scenes. In practice, one does not need a priori knowledge of the start and end frames of a complete dissolve in order to model the expected changes. A solution is to measure the pixel differences between three consecutive frames  $A$ ,  $B$ , and  $C$ . The task is to determine if  $B$  is a dissolving (or fading) frame. The  $A/B$  differences are computed and compared to the  $B/C$  differences. Only if there is enough consistency is an attempt made to verify the effect by synthesizing the average of  $A$  and  $C$  and comparing it to  $B$ .  $A$  and  $C$  might also be dissolving (fading) frames but that does not affect the  $AC/B$  comparison, and identifying where the overall beginning or end of the effect is something a higher level of logic can determine afterward. Note that one cannot just always compute the  $AC/B$  comparison to discover dissolves (fades) because a static sequence also satisfies that condition. There must be a pixel-wise consistent trend to the change in luminance or chrominance over at least three frames. More frames may be necessary if false positives due to the transient pixel jitter and brief flashes are to be minimized.

A similar approach is used by Meng et al. [26] for detecting dissolves. Consider that  $I_1(t)$  and  $I_2(t)$  are two ergodic sequences with intensity variance  $\sigma_1^2$  and  $\sigma_2^2$ . The model used for dissolve is given by the Eq. 7.15. In the ideal case when  $I_1(t)$  and  $I_2(t)$  are ergodic with  $\sigma_1^2$  and  $\sigma_2^2$ , the variance curve in the dissolve region shows a parabolic shape. In real sequences with motions, the variances of  $I_1(t)$  and  $I_2(t)$  are not constant but the dissolve region still demonstrates the parabolic shape.

Corridoni and Del Bimbo [8] used the  $L^*u^*v^*$  color space representation, which separates the brightness feature from the color. Considering that in a fade sequence there is a linear variation in the luminance values ( $l$ ) and constancy in the chrominance values ( $u$  and  $v$ ), they introduced the following measure for fade detection:

$$D_{fade} = \frac{l_i}{u_i v_i (\varepsilon + |C_i^L - C_i|)} \quad (7.16)$$

where  $l_i$  is the number of pixels whose change in  $l$  is linear;  $u_i$  and  $v_i$  are the number of pixels whose change in  $u$  and  $v$  is lower than a threshold imposed by Gaussian noise;  $C_i$  is the spatial center of video frame  $f_i$  and

$$C_i^L = \frac{1}{l_i} (x_i^L, y_i^L) \quad (7.17)$$

where  $x_i^L$  and  $y_i^L$  are the number of pixels whose change in luminance along  $x$  and  $y$  is linear. The right term in Eq. 7.16 is a spatial constraint to impose uniform variation of brightness on the video frame. The measure is averaged over a wide temporal window (e.g., 16) and whenever the temporal average over the window overcomes a threshold, a fade is detected.

The dissolve is described by the composition of a fade-in and a fade-out for the brightness value. Although the  $u$  and  $v$  values do not remain constant, the amount of changing pixels is high and is statistically distributed uniformly over the image. The dissolve detection is based on the following measures:

$$D_{diss}^1 = \frac{l_i}{\varepsilon + |C_i^L - C_i|}, \quad (7.18)$$

$$D_{diss}^2 = \frac{u_i + v_i}{\varepsilon + |C_i^{UV} - C_i|} \quad (7.19)$$

where  $C_i^{UV}$  is defined similar to  $C_i^L$  (see Eq. 7.17).

A dissolve is characterized by two consecutive maxima in the  $D_{diss}^1$  measure and by a maximum in the  $D_{diss}^2$  measure, in correspondence with the valley between the two maxima.

Aigrain and Joly [4] developed a model for the probability density function of intershot pixel differences in successive frames for various transition effects. In their model, pixel differences in a cut transition follows the linear law with density function defined as:

$$Q(s) = \frac{2(a - |s|)}{a(a - 1)} \quad (7.20)$$

where  $a$  is the number of gray levels.

Pixel differences in a wipe transition also follow a linear law with density function defined as:

$$Q(s) = \frac{2(a - |s|)}{da(a - 1)} \quad (7.21)$$

where  $d$  is the duration of the wipe transition in a number of sampled images.

In the case of a cross-dissolve transition, the pixel differences follow the law:

$$Q(s) = \begin{cases} \frac{2d(a-d|s|)}{a(a-1)} & \text{for } d|s| \leq a, \\ 0 & \text{otherwise} \end{cases} \quad (7.22)$$

where  $d$  is the duration of the cross-dissolve.

Pixel differences in a fade to black, fade to white, fade from black or fade from white transition follow a linear law:

$$Q(s) = \begin{cases} \frac{d}{a} & \text{for } d|s| \leq a, \\ 0 & \text{otherwise} \end{cases} \quad (7.23)$$

for  $s > 0$  (fade from black or fade to white) or  $s < 0$  (fade from white or fade to black).

Shot transitions are detected by fitting these models to pixel difference data obtained from the sequence.

**Detection of Camera Motion.** To detect the camera motion, optical flow techniques are utilized. Optical flow gives the distribution of velocity with respect to an observer over the points in an image. Optical flow is determined by computing the motion vector of each pixel in the image. The fields generated by zoom, pan and tilt are shown in Fig. 7.5. Detecting these fields helps in separating the changes introduced due to the camera movements from the special-effects such as wipe, dissolve, fade-in, fade-out.

As seen in Fig. 7.5 most of the motion vectors between consecutive frames due to pan and tilt point in a single direction thus exhibiting a strong modal value corresponding to the camera movement. Disparity in direction of some of the motion vectors will result from object motion and other kinds of noise. Thus, a single modal vector is exhibited with respect to the camera motion. As given by Eq. 7.24, a simple comparison technique can be used to detect pan and tilt. Pan or tilt are detected by calculating the differences between the modal vector and the individual motion vectors [49]. Let  $\theta_l$  be the direction of the motion vectors and  $\theta_m$  the direction of the modal vectors, then:

$$\sum_{l=1}^N |\theta_l - \theta_m| \leq \Theta_p. \quad (7.24)$$

If the sum of the differences of all vectors is less than or equal to a threshold variation  $\Theta_p$  then a camera movement is detected. This variation should be zero if no other noise is present.

Motion vectors for zoom have a center of focus, i.e., a focus of expansion (FOE) for zoom-in, and a focus of contraction (FOC) for zoom-out. Due to the absence of noise, it is easy to detect the zoom because the sum of the motion vectors around the FOC/FOE will be zero. However, it is difficult to find

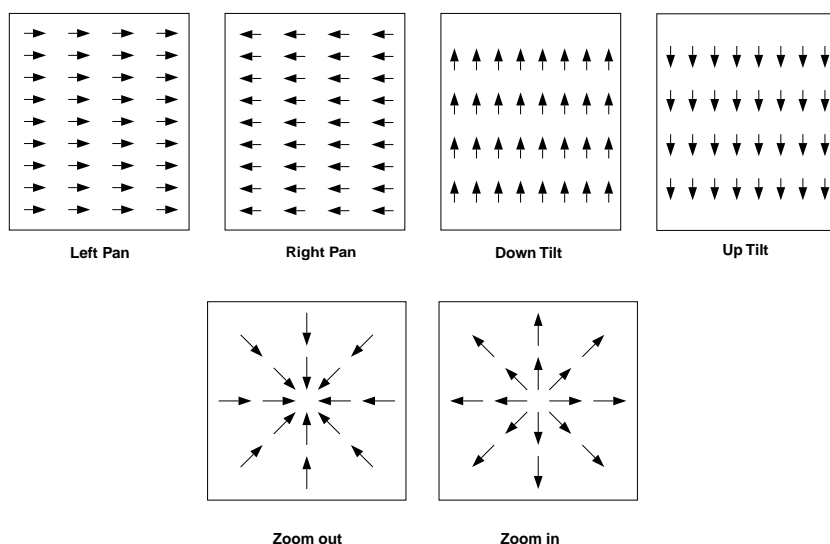


Fig. 7.5 Optical flow field produced by pan, tilt, and zoom.

the center of focus of zoom since it could be present across two consecutive frames. Zhang et al. [49] assume that the FOE/FOC lies within a frame, therefore, it is not necessary to locate it for vector comparison. A simple comparison technique can be used, the vertical vectors from the top ( $V_k^{top}$ ) and the bottom row ( $V_k^{bottom}$ ) can be compared for magnitude and horizontal vectors from the left-most ( $U_k^{top}$ ) and right-most ( $U_k^{bottom}$ ) vectors at the same row can be compared. In the case of zoom, the vectors will have opposite signs, and the magnitude of the difference of these components should exceed the magnitude of the highest individual component. This is due to the fact that in every column the magnitude of the difference between these vertical components will exceed the magnitude of both components:

$$|V_k^{top} - V_k^{bottom}| \geq (|V_k^{top}|, |V_k^{bottom}|), \quad (7.25)$$

$$|U_k^{top} - U_k^{bottom}| \geq (|U_k^{top}|, |U_k^{bottom}|). \quad (7.26)$$

When both Eqs 7.25 and 7.26 are satisfied, a zoom is declared. Thus, camera movements can be separated from the gradual transitions.

Optical flow works well for detecting motion, but it can be difficult to distinguish between object and camera motion without understanding the video content. Moreover, pure pans and tilts are exceptions in real-world videos, mixed-axis motion is more common. In general, it is more likely that pure  $X$  or  $Y$  motion is attributable to the camera, whereas mixed-axis motion is more likely attributable to objects. However, there are too many exceptions to this generalization to make it very useful.

Motion vector information can also be obtained from MPEG compressed video sequences. However, the block matching performed as a part of MPEG encoding selects vectors based on compression efficiency and thus often selects inappropriate vectors for image processing purposes.

**DCT-Based Method.** In this category of methods, the discrete cosine transform (DCT) coefficients are used in comparing two video frames. The DCT is commonly used for reducing spatial redundancy in an image in different video compression schemes such as MPEG and JPEG. Compression of the video is carried out by dividing the image into a set of  $8 \times 8$  pixel blocks. Using the DCT the pixels in the blocks are transformed into 64 coefficients which are quantized and Huffman entropy encoded. The DCT coefficients are analyzed to find frames where camera breaks take place. Since the coefficients in the frequency domain are mathematically related to the spatial domain, they can be used in detecting the changes in the video sequence.

A DCT correlation method is described by Arman et al. [5]. Given  $8 \times 8$  blocks of a single DCT-based encoded video frame  $f_i$ , a subset of blocks is chosen a priori. The blocks are chosen from  $n$  connected regions in each frame. For each block, 64 randomly distributed AC coefficients are chosen. By taking coefficients from each frame a vector  $\mathbf{V}_i = \{c_1, c_2, c_3, \dots\}$  is formed. The frame difference between two successive frames  $f_i$  and  $f_{i+1}$  is defined as the inner product of the two corresponding vectors  $\mathbf{V}_i$  and  $\mathbf{V}_{i+1}$ :

$$D_i(f_i, f_{i+1}) = 1 - \frac{\mathbf{V}_i \mathbf{V}_{i+1}}{|\mathbf{V}_i| |\mathbf{V}_{i+1}|}. \quad (7.27)$$

A transition is detected when  $D_i > T$ , where  $T$  is the threshold.

Zhang et al. [51] proposed a method which uses the pairwise difference of DCT coefficients. Let  $c_{i,b}(m, n)$  be the DCT coefficient at  $(m, n)$  of a block  $b$  in video frame  $f_i$ . A binary function for deciding whether there is a change between corresponding DCT blocks of video frames  $f_i$  and  $f_{i+1}$  is defined as follows:

$$DB_{i,b}(f_i, f_{i+1}) = \begin{cases} 1 & \text{if } \frac{1}{64} \sum_{m=0}^7 \sum_{n=0}^7 \frac{|c_{i,b}(m,n) - c_{i+1,b}(m,n)|}{\max[c_{i,b}(m,n), c_{i+1,b}(m,n)]} > t, \\ 0 & \text{otherwise} \end{cases} \quad (7.28)$$

where  $t$  is a user-defined threshold. Further, the frame difference between  $f_i$  and  $f_{i+1}$  is defined as:

$$D_i(f_i, f_{i+1}) = \frac{1}{B} \sum_{b=0}^{B-1} DB_{i,b}(f_i, f_{i+1}) \quad (7.29)$$

where  $B$  is the total number of  $8 \times 8$  blocks. So, if in Eq. 7.28  $DB_{i,b}$  is larger than the threshold, the block  $b$  is considered to be changed. If the number

of changed blocks Eq. 7.29 exceeds a threshold, a shot boundary between  $f_i$  and  $f_{i+1}$  frames is declared. Compared with the previous technique [5], the processing time of this technique is smaller; however, it is more sensitive to gradual changes.

Yeo and Liu [43] use the pixel differences of the luminance component of DC frames. The DC image is a spatially reduced version (1/64) of the original video frame. Each pixel of a DC image is equal to the average intensity of each  $8 \times 8$  block of the frame. Under these conditions the frame difference between  $f_i$  and  $f_{i+1}$  is defined as:

$$D_i(f_i, f_{i+1}) = \sum_{b=0}^{B-1} |dc_i(b) - dc_{i+1}(b)| \quad (7.30)$$

where  $B$  is the total number of  $8 \times 8$  blocks and  $dc_i(b)$  is the DC coefficient of block  $b$  in frame  $f_i$ . Although this technique is fast, cuts may be misdetected between two frames which have similar pixel values but different density functions.

Patel and Sethi [30] have experimented with various statistics and have found that the  $\chi^2$  statistic gives the best performance. They use intensity histograms obtained from the entire frame. The histograms are found using DC coefficients of MPEG video for only  $I$  frames. Taskiran and Delp [40] use a *generalized trace* (GT). This high dimensional feature vector is composed of a set of features extracted from each DC frame. The GT is then used in a binary regression tree to determine the probability that each frame is a shot boundary. These probabilities can then be used to detect frames corresponding to shot boundaries.

**Vector Quantization.** Idris and Panchanathan [16] use vector quantization to compress a video sequence using a codebook of size 256 and 64-dimensional vectors. The histogram of the labels obtained from the codebook for each frame is used as a frame similarity measure and a  $\chi^2$  statistic is used to detect cuts. Consider the histogram of the labels of a frame  $f_i$ ,  $\{H_i(k); k = 1, 2, \dots, N\}$  where  $H_i(k)$  is the number of labels  $k$  in the compressed frame and  $N$  is the number of codewords in the codebook. The difference between two frames  $f_i$  and  $f_j$  is:

$$D_i(f_i, f_j) = \sum_{k=1}^N \frac{(H_i(k) - H_j(k))^2}{|H_i(k) - H_j(k)|} \quad (7.31)$$

A large value of  $D_i(f_i, f_j)$  indicates that  $f_i$  and  $f_j$  belong to different scenes. An abrupt change is declared if the difference between two successive frames exceeds a threshold and a gradual transition is detected if the difference between the current frame and the first frame of the current shot is greater than a threshold.

**Color-Ratio-Based Method.** A color-ratio-based method for uncompressed and compressed videos is introduced by Adjero et al. [2]. Using the fact that neighborhood-based color ratios are invariant under various changes in viewing conditions, including both spectral and intensity variations of the illuminant, their aim is to obtain illumination invariant and motion invariant features for window-based video partitioning.

Let  $h(x, y)$  be the red, green, or blue intensity of pixel  $(x, y)$ , then the four-neighbor color ratio for the pixel  $(x, y)$  of a video frame  $f_i$  is defined as:

$$\Phi_i(x, y) = \frac{h(x, y-1) + h(x, y+1) + h(x-1, y) + h(x+1, y)}{4h(x, y)} \quad (7.32)$$

The window-based difference between frames  $f_i$  and  $f_{i+1}$  is computed:

$$\mathcal{J}_{i,w}(f_i, f_{i+1}) = \begin{cases} 1 - \xi_{i,w}(f_i, f_{i+1}) & \text{if } \xi(\cdot) \leq 1, \\ 1 - \frac{1}{\xi_{i,w}(f_i, f_{i+1})} & \text{otherwise} \end{cases} \quad (7.33)$$

where  $w$  denotes a window, and

$$\xi_{i,w}(f_i, f_{i+1}) = \frac{\Psi_i(w)}{\Psi_{i+1}(w)}, \text{ and } \Psi_i(w) = \prod_{(x,y) \in w} \Phi_i(x, y). \quad (7.34)$$

The frame difference between  $f_i$  and  $f_{i+1}$  is further defined as:

$$DW_{i,w}(f_i, f_{i+1}) = \begin{cases} 1 & \text{if } \mathcal{J}_{i,w}(f_i, f_{i+1}) > t, \\ 0 & \text{otherwise,} \end{cases} \quad (7.35)$$

$$D_i(f_i, f_{i+1}) = \frac{1}{W} \sum_{w=0}^{W-1} DW_{i,w}(f_i, f_{i+1}) \quad (7.36)$$

where  $t$  is a user-defined threshold for deciding whether there is a change between corresponding windows of two video frames and  $W$  is the total number of windows.

In the case of compressed video, similar to the case of uncompressed video, an  $n$ -neighbor color ratio for a DCT coefficient  $H_{i,b}(u, v)$  of a DCT image is defined as:

$$\Phi_i(u, v) = \frac{\sum_{r \in \{n\text{-neighbor of } b\}} H_{i,r}(u, v)}{nH_{i,b}(u, v)} \quad (7.37)$$

where  $n$  is the number of neighboring blocks involved,  $H_{i,b}(u, v)$  is the transform coefficient at the location  $(u, v)$  in a block  $b$ , and  $H_{i,r}(u, v)$  is the coefficient at the corresponding  $(u, v)$  location in neighboring block  $r$ . The block-based difference between  $f_i$  and  $f_{i+1}$  is computed as:

$$\mathcal{J}_{i,b}(f_i, f_{i+1}) = \begin{cases} 1 - \xi_{i,b}(f_i, f_{i+1}) & \text{if } \xi(\cdot) \leq 1, \\ 1 - \frac{1}{\xi_{i,b}(f_i, f_{i+1})} & \text{otherwise} \end{cases} \quad (7.38)$$

where  $b$  denotes a block, and

$$\xi_{i,b}(f_i, f_{i+1}) = \frac{\Psi_i(b)}{\Psi_{i+1}(b)}, \quad \Psi_i(b) = \prod_{(u,v) \in b} \Phi_i(u, v). \quad (7.39)$$

Since the DC and AC coefficients usually carry different types of information about the scene, a weighting factor  $\phi, 0 \leq \phi \leq 1$ , can be defined to vary the block-based difference adaptively, depending on the scene.

1. DC represents an average of the sample value in the spatial domain.
2. AC components provide information on the level of “scene activity” or detail:

$$\mathcal{J}_{i,b}(f_i, f_{i+1}) = \phi \mathcal{J}_{i,b}^{DC}(f_i, f_{i+1}) + (1 - \phi) \mathcal{J}_{i,b}^{AC}(f_i, f_{i+1}). \quad (7.40)$$

The frame difference between  $f_i$  and  $f_{i+1}$  is finally defined as:

$$DB_{i,b}(f_i, f_{i+1}) = \begin{cases} 1 & \text{if } \mathcal{J}_{i,b}(f_i, f_{i+1}) > t, \\ 0 & \text{otherwise,} \end{cases} \quad (7.41)$$

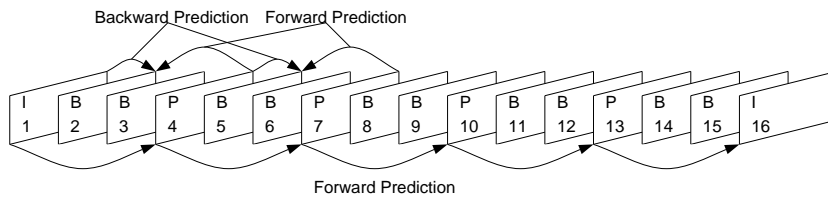
$$D_i(f_i, f_{i+1}) = \frac{1}{B} \sum_{b=0}^{B-1} DB_{i,b}(f_i, f_{i+1}) \quad (7.42)$$

where  $t$  is a user-defined threshold for deciding whether there is a change between corresponding windows of two video frames and  $B$  is the total number of blocks.

**Motion-Based Segmentation.** Motion analysis is an important step in video processing. A video stream is composed of video elements constrained by the spatiotemporal piecewise continuity of visual cues. The normally coherent visual motion becomes suddenly discontinuous in the event of shot boundaries. Hence, motion discontinuities may be used to mark the change of a scene, the occurrence of occlusion, or the inception of a new activity.

In the MPEG standard, to reduce temporal redundancy, video frames are coded with reference to the contents of the other previous or following frames by motion compensation prediction. Figure 7.6 illustrates an MPEG sequence of frames with a GOP (group of pictures) size of 15.

There are three types of frames, namely  $I$ -frame,  $P$ -frame, and  $B$ -frame. Every  $I$ -frame is encoded without referencing to other frames. In this case no motion compensation is performed. A  $P$ -frame is predictively coded with motion compensation from past  $I$ - or  $P$ -frames. Both these frames are used as a basis for bidirectional motion compensated  $B$ -frames. The prediction is



**Fig. 7.6** A typical MPEG frame sequence in display order.

based on matching a  $16 \times 16$  pixel macroblock of a frame as closely as possible to another macroblock of a preceding or succeeding frame. For those predicted frames (i.e.,  $P$  or  $B$ ) a number of motion vectors indicating the types and the distances will be generated. The number of motion vectors associated with a frame depends on the residual error after motion compensation. If two frames are quite different, motion compensation will be abandoned and most macroblocks will be intra-coded, resulting in a smaller number of motion vectors. Thus, a scene cut can be detected based on the counts of motion vectors.

Zhang et al. [51] have proposed a technique for cut detection using motion vectors in MPEG. In  $P$ -frames,  $M$  is the number of motion vectors and in  $B$ -frames,  $M$  is the smaller of the counts of the forward and backward non-zero motion. Then  $M < T$  will be an effective indicator of a camera boundary before or after the  $B$ - and  $P$ -frame, where  $T$  is a threshold value close to zero. However, this method yields false detection where there is no motion. This is improved by applying the normalized inner product metric (Eq. 7.27) to the two  $I$ -frames on the sides of the  $B$ -frame where a break has been detected.

Kuo et al. [20] define two types of reference ratios ( $RRs$ ) for  $P$ - and  $B$ -frames. Forward reference ratio ( $FRR$ ) is defined as the ratio between the number of the forward-predicted macroblocks ( $R_f$ ) and the total number of macroblocks ( $N$ ). On the other hand, backward reference ratio ( $BRR$ ) is defined as  $R_b/N$ , where  $R_b$  is the number of backward predicted macroblocks.  $RRs$  of frames within the same shot will be high (compared with a pre-defined threshold) and when a shot boundary occurs, since the contents of frames are assumed to be different from the preceding frames, the  $RRs$  will be low. The conditions for a scene change to occur are described as follows:

**Scene cut at  $I$ -frame.** The  $BRRs$  of previous  $B$ -frames are low.

**Scene cut at  $P$ -frame.** The  $BRRs$  of previous  $B$ -frames and the  $FRR$  of this  $P$ -frame are low.

**Scene cut at  $B$ -frame.** The  $BRRs$  of previous  $B$ -frames (if any), the  $FRR$  of this  $B$ -frame, and the  $FRRs$  of following  $B$ - and  $P$ -frames (if any) are all low.

They propose a mask matching approach to detect the above different ratio patterns. For example, let the frame pattern be  $IBBPBBPBBI \dots$ , then  $\{I : (B_b, B_b, @I)\}$  is a mask which specifies the condition that the  $BRRs$

of  $B$ -frames before the current  $I$ -frame should be low, if a scene change occurs at the current  $I$ -frame. A frame marked with a symbol @ denotes the frame under consideration. Similarly,  $\{P : (B_b, B_b, @P_f)\}$  is a mask designed for detecting scene cuts at the current  $P$ -frame.  $P_f$  denotes a criterion that the  $RR$  of the  $P$ -frame should be low for a scene change to occur. For a  $B$ -frame, either  $\{B : (@B_f, B_f, P_f)\}$  or  $\{B : (B_b, @B_f, P_f)\}$  is used depending on whether or not the current  $B$ -frame is immediately preceded by an  $I$ - or  $P$ -frame. A function is defined to convert the results of mask matching to shot change probabilities from which the global threshold method is used to detect scene cuts:

$$P_i = 1 - \frac{RR_{f_1}^2 + RR_{f_2}^2 + \cdots + RR_{f_n}^2}{RR_{f_1} + RR_{f_2} + \cdots + RR_{f_n}} \quad (7.43)$$

where  $f_1, f_2, \dots, f_n$  are mask frames of a mask  $i$  and  $RR_f$  is the  $RR$  of mask frame  $f$ .

The global threshold  $T_b$  is defined as  $(F + F')/2$ , where  $F$  is the average of the 97.5% smallest scene change probabilities and  $F'$  is the average of the 2.5% largest scene change probabilities. This is based on the assumption that on average there is a scene change for every 40 frames.

To reduce the effects of illumination and fast motions, and to avoid the situations of two or more shot changes in a short period of time, an adjustment to shot change probabilities can be defined as follows:

$$F(P_i) = \begin{cases} P_i - \min(P_k) & \text{if } P_i = \max(P_k), \quad i - j \leq k \leq i + j, \\ 0 & \text{otherwise} \end{cases} \quad (7.44)$$

where  $j$  is a parameter of the window size.

Liu and Zick [24] have presented a technique based on the error signal and the number of motion vectors. A scene cut between a  $P$ -frame  $P_m$  and a past reference  $P$ -frame  $P_n$  increases the error energy. Hence, the error energy provides a measure of similarity between  $P_m$  and the motion compensated frame  $P_n$ :

$$S(P_m, P_n) = \frac{\sum_{i=1}^{R_f} E_i}{R_f^2} \quad (7.45)$$

where  $R_f$  is the number of forward predicted macroblocks and  $E_i$  is the error energy of macroblock  $i$ :

$$E_i = \sum_m \sum_{j=1}^{64} c_{i,m,j}^2 \quad (7.46)$$

where  $c_{i,m,j}$  is the  $j$ th DCT coefficient in block  $m$  of macroblock  $i$ .

For the detection of scene changes based on  $B$ -frames, the difference between the number of forward predicted macroblocks  $R_f$  and backward predicted  $R_b$  is used. A scene change between a  $B$ -frame and its past reference  $B$ -frame will decrease  $R_f$  and increase  $R_b$ . A scene change is declared if the difference between  $R_f$  and  $R_b$  changes from positive to negative.

Meng et al. [26] define three ratios  $R_p$ ,  $R_b$ , and  $R_f$  based on the MPEG coding characteristics. Let  $X$  be the number of forward predicted macroblocks,  $Y$  the number of backward predicted macroblocks, and  $Z$  the number of intra-coded blocks. Then  $R_p = Z/X$ ,  $R_b = Y/X$ , and  $R_f = 1/R_b$ . Within the same scene, motion vector ratios tend to be similar. However, different scenes may have very different motion vector ratios. An adaptive local window threshold setting technique for detecting peaks in  $R_p$ ,  $R_b$ , and  $R_f$  is proposed. Ratio values in a window of two GOPs are used in calculating a ratio histogram with 256 bins. If the peak-to-average ratio is greater than a pre-defined threshold  $T_d$ , then the peak frame is declared as a suspected scene change. Note that the peak values are not included in calculating the average and the typical value for  $T_d$  is 3. If a scene change happens on a  $B$ -frame, the ratio of the immediately subsequent  $B$ -frame will also be high. Both ratios will be considered as peaks and only the first  $B$ -frame will be marked as a suspected scene change. The summary of the method is as follows:

**Detect  $R_p$  peaks in  $P$ -frames.** Mark them as suspected shot boundaries.

**Detect  $R_b$  peaks in  $B$ -frames.** Mark them as suspected shot boundaries.

**Detect  $R_f$  peaks in  $B$ -frames.** Detect all  $|\Delta\sigma^2|$  peaks in  $I$ - and  $P$ -frames.

Mark  $I$ -frames as a suspected shot boundary if they have  $|\Delta\sigma^2|$  peaks and the  $B$ -frames immediately before them have  $R_f$  peaks.

**All the marked frames are then examined.** If the difference between the current marked frame and the last shot boundary exceeds a threshold, then the current marked frame is a true shot boundary.

We should note some shortcomings of the techniques involving MPEG. The most severe one is the dependence of the algorithms on the characteristics of a particular encoder. Different MPEG encoders are more or less aggressive in their exploitation of motion compensation and can vary widely in the size of their GOP and selection of  $I$ -,  $P$ -, and  $B$ -frames. As a consequence, compressed domain shot boundary detection that works well with video produced by one MPEG encoder could require different parameters settings or might not even find the information it requires in video from another encoder.

**Segmentation Based on Object Motion.** Video data can also be segmented based on the analysis of encapsulated object motion [22]. The dynamic-scene analysis is based on recognizing the objects and then calculating their motion characteristics. Objects in video data are recognized either by their velocity or by their shape, size, and texture characteristics.

The motion which we perceive is the effect of both an object and camera motion. Camera motion information can be exploited in the scene segmentation process based on object motion. For example, if the camera is in motion,

then the image has nonzero relative velocity with respect to the camera. The relative velocity can be defined in terms of the velocity of the point in the image and the distance from the camera. For segmenting moving-camera scenes, the component of the motion due to camera motion should be removed from the total motion of an object. Accumulative-difference pictures [18] can be used to find areas that are changing (usually due to object motion) and hence imply scene segmentation. Velocity components of the points in an image can be used for segmenting a scene into different moving objects. This is due to the fact that points of an object optimistically have the same velocity components under certain camera motions and image surfaces. Equation (7.47) can be used for calculating the image intensity velocity, where  $I(x, y, t)$  is the image intensity,  $\partial I/\partial x$  and  $\partial I/\partial y$  denote the image intensity gradient, and  $dx/dt$  and  $dy/dt$  denote the image velocity:

$$dI/dt = \partial I/\partial x \times dx/dt + \partial I/\partial y \times dy/dt + \partial I/\partial t. \quad (7.47)$$

If the image intensity velocity changes abruptly, a shot boundary can be declared. This method is much more computationally intensive because the object must be traced through a sequence of frames.

**Segmentation Based on Subband-Coded Video Data.** Lee and Dickinson [21] have presented a scene detection algorithm where the temporal segmentation is applied on the lowest subband in subband-compressed video. Four different metrics for segmentation of video data have been developed as follows:

1. Difference of histograms: measures the absolute sum of the histograms of two frames. This metric is insensitive to object motion; however, it is sensitive to camera operations such as panning and zooming.
2. Histogram of difference frame: measures the histogram of the pixel to pixel difference frame. The degree of change is large if there are more pixels distributed away from the origin. It is more sensitive to object motion than the previous measure.
3. Block histogram difference: block histogram difference is obtained by computing histograms of each block and summing the absolute difference of these block histograms between the two frames. This metric is sensitive to local object motion.
4. Block variance difference: instead of using the histogram the variance of the block is used. Since it is a block-based metric, it is sensitive to local object motion.

**Segmentation Based on Features.** Work on feature-based segmentation is being done by Zabih et al. [48]. The segmentation process involves analyzing intensity edges between two consecutive frames. During cut and dissolve operations, new intensity edges appear far from the old ones due to change in content. In addition, old edges disappear far from the location of new edges. The authors define an edge pixel that appears far from an existing edge pixel

as an *entering* edge pixel, and an edge pixel that disappears far from an existing edge pixel as an *exiting* edge pixel. By counting the entering and exiting edge pixels, cuts, fades, and dissolves are detected and classified. By analyzing the spatial distribution of entering and exiting edge pixels, wipes are detected and classified.

The algorithm takes as input two consecutive images  $I$  and  $I'$  and using an edge detection step, two binary images  $E$  and  $E'$  are obtained. Let  $\rho_{in}$  denote the fraction of edge pixels in  $E'$  which are more than a fixed distance  $r$  from the closest edge pixel in  $E$ .  $\rho_{in}$  measures the proportion of the entering edge pixels. It should assume a high value during a fade-in, or a cut, or at the end of a dissolve. Similarly, let  $\rho_{out}$  be the fraction of edge pixels in  $E$  which are farther than  $r$  away from the closest edge pixel in  $E'$ .  $\rho_{out}$  measures the proportion of exiting edge pixels. It should assume a high value during a fade-out, or a cut, or at the beginning of a dissolve.

The basic measure of similarity introduced is:  $\rho = \max(\rho_{in}, \rho_{out})$ . This represents the fraction of changed edges. Scene breaks can be detected by looking for peaks in  $\rho$ . Cuts are easy to distinguish, because a cut is the only scene break that occurs entirely between two consecutive frames. As a consequence, a cut will lead to a single isolated frame with a high value of  $\rho$ , while the other scene breaks will lead to an interval where  $\rho$ 's value is elevated.

Fades and dissolves can be distinguished from each other by looking at the relative values of  $\rho_{in}$  and  $\rho_{out}$  in a local region. During a fade-in,  $\rho_{in}$  will be much higher than  $\rho_{out}$ , since there will be many entering edge pixels and few exiting edge pixels. Similarly, at a fade-out,  $\rho_{out}$  will be higher than  $\rho_{in}$ . A dissolve, on the other hand, consists of an overlapping fade-in and fade-out. During the dissolve, there is a initial peak in  $\rho_{in}$  followed by a peak in  $\rho_{out}$ .

During a wipe, each frame will have a portion of the old scene and a portion of the new scene. Between adjacent frames, a single strip of the image will change from the old scene to the new scene. For an horizontal wipe there is a vertical strip that passes either left-right or right-left, depending on the direction of the wipe. Since the between-scene transition occurs in this strip, the number of edge pixels that either enter or exit should be higher inside the strip and lower in the other areas of the image. The edge pixel that is either entering or exiting is called *changing* pixel. When computing  $\rho$  the location of the changing edge pixels can be recorded and their spatial distribution analyzed. There are different types of wipes that are detected. For vertical and horizontal wipes, the percentage of changing pixels in the top half and the left half of the image is calculated. For a left-to-right horizontal wipe, the majority changing pixels will occur in the left of the image during the first half of the wipe, then in the right half of the images during the rest of the wipe. The other cases can be handled similarly.

Motion compensation is also provided in the algorithms to give better results in the presence of camera and object motion. Shen et al. [37] applied

this technique to MPEG sequences using a multi-level Hausdorff distance histogram.

**Video Shot Representation.** Various researchers observed that the representation of a video shot as a single still image is a significant step towards retrieving pieces of video from large collections of video clips. If there is no independent motion (only camera motion is presented) in a shot, the transformation between subsequent frames can be determined, and the frames can be “pasted” together into a larger image, possibly of higher resolution. This technique is called mosaicking [33]. On the other hand, if there is independent motion, the moving objects must be separated from the background, and the image of the background can still be reconstructed along with images and trajectories of moving objects. Techniques proposed for obtaining such still images are salient stills [38, 41], mosaicking [33, 39] and video space icons [42]. One of the reasons for this extracting process is that once a complete shot is transformed into a single still image, then all image database browsing and retrieval techniques can be applied. Along with providing the information about this static image (color, texture, shape) this process also provides information about the camera motion and independent object motions, which permits motion queries [33].

A simpler way to obtain a single image to represent a shot is to select a keyframe or, if there is much motion and action in the shot, to select multiple keyframes [44, 51]. In this way the problem of video query is reduced to the problem of image query. However, in this case usually one will obtain a large collection of shots from a normal video clip. Thus, searching large video databases may be transformed to searching large image databases.

In summary, video search requires more information to be used than just static images derived from video. In the case of video, the search has to go beyond the static features of images and incorporate the dynamics within a shot and between the shots.

### 7.2.2 Video Stories

During a film, a story is told using the presentation of the images in time; the sequence of such a presentation forms the montage or the edit of the film. Technically, [1] the montage is “the most essential characteristic of the motion picture. Montage refers to the editing of the film, the cutting and piecing together of exposed film in a manner that best conveys the intent of the work.” Properly edited video has a continuity of meaning that overwhelms the inherent discontinuity of presentation. The idea is that intrinsically a film has two levels of discontinuity. One is the frame-to-frame discontinuity which is mostly imperceptible; the other one is the shot-to-shot transition which can be easily detected and perceived. In essence, a shot in a video is very much like a sentence in a piece of text; it has some semantic meaning, but taken out of context, it may be ambiguous. Thus, there is a need to concatenate groups

of shots in order to form a three-dimensional event which is continuous in time. This concatenation forms a story unit which often represents a time and place in the narration. The continuity in time is not as important as the continuity in meaning. Therefore, one important challenging problem of video analysis is to find the discontinuities in meaning or, equivalently, establishing from shot to shot whether there is a continuity in the subject of video.

Some work towards such achievements was carried out in the HyperCafe Project [34]. The authors described a formal approach for the design of video-centric hypermedia applications. Their framework defines a way to structure video-based scenes into narrative sequences that can be dynamically presented as spatial and temporal opportunities. Some other efforts were directed towards creation of storyboards [7, 31, 47]. Here, high-level video structure is extracted and presented to the user. In such conditions, the user can conveniently browse through the video on a story-unit basis. Typically, there is an order of magnitude reduction from the number of shots to the number of scenes [47]. This represents a significant reduction of information to be conveyed or presented to the user, thus making it easy for the user to identify segments of interest.

Furthermore, the ability to automatically label a story unit as a “dialogue” or “action” means that one can query on the basis of such semantic characteristics. Video summaries can be labeled with such semantic descriptions, allowing video content viewing and fast random access.

### 7.2.3 Video Parsing

Video parsing has been successfully applied to recover individual news items from news programs [50]. A priori models are constructed through the use of state transitions, where each state corresponds to a phase of a news broadcast such as news anchor speaking, a reporter speaking and a weather forecast. In the recognition stage, visual and temporal elements are used together with domain knowledge of spatial layouts of objects.

After the different items of the news are recovered the user can view any part of a news program. This offers a high-level random access into a different segment of the news according to what the user is interested in. If he is interested only in the weather forecast he can directly jump into the segment corresponding to weather forecast without having to look into all of the video. In work by Mohan [27] a shot-boundary detection algorithm in combination with the detection of audio silences is used to segment the news into individual news items. Additionally, the closed caption text is synchronized with the visual news item and used for text-based retrieval of the news items.

A different approach is used in the system by Shahraray and Gibbon [35]. The authors use the closed-caption information for news archive retrieval. The pictorial transcript system they developed transcribes news programs into HTML-based summaries. Each HTML page contains several news items,

each being represented by keyframes with detailed text derived from closed-captions.

#### 7.2.4 Video Summaries

Visual queries give the end user the opportunity to have a quick idea of what a particular candidate is about. The summaries allow the user to refine his precedent query or to have an idea whether the query has been posed correctly. Using a summary the user can have rapid non-linear access in viewing the video and can get an idea of the visual content.

In Yeung et al. [45] a compact representation of video content called *scene-transition graph* can be built on clusters of similar video shots. This representation is a form of visual summary that maps a sequence of video shots onto a two-dimensional display on the screen. The graph representation has nodes, which capture the temporal video content information, and edges, which depict the temporal flow of the story.

Another way to provide the user with a video summary is to emulate a video player that displays some extracted keyframes. In Ding et al. [11], they propose a *slide show interface* that flips through computer-selected video keyframes in a temporally ordered sequence, displaying the “slides” at rates set by the user. A similar approach is described in the work by DeMethod et al. [10]. Their interface allows the user to change the summarization level on the fly. They also provide a way to automatically select the summarization level that provides a concise and representative set of keyframes.

Another form of visual summary is the so called *pictorial summary* [46]. The pictorial summary consists of a sequence of representative images arranged in temporal order. Each representative image, called a video poster, consists of one or more subimages, each of which represents a unique “dramatic element” of the underlying story, such that the layout of the subimages emphasizes the dramatic “importance” of individual story units. Note that in this case a kind of semantic VCR is emulated. The pictorial summary is thus a general scheme for visual presentation, and each video poster visually summarizes the dramatic events taking place in a meaningful story unit of the story.

#### 7.2.5 Automatic Attributes Generation for Video

The features that are extracted by the described algorithms can provide high-level annotation of video. The annotations form the attributes that are attached to the different levels of the hierarchy of video representation described in Section 7.1.3. For a video sequence the story units partition the video into a small number of meaningful units, each representing a dramatic event. The number of dialogs and their duration can be an example of attributes for story units. At the lowest level of the hierarchy will be the attributes of each

shot. The temporal ones can be derived from motion analysis, object tracking, color changes, etc., while the static ones include the properties of the shot itself (duration) and those of its images (keyframes or mosaics) such as color histograms, texture patterns, etc.

Beyond visual features, the information from the audio track, bibliographic data and closed caption can also provide textual annotation. For example audio analysis can be used to classify the audio information of a video (silence, speech, music).

### 7.3 Discussion

A framework for video retrieval and query formulation that is based on a sequence of categorization, searching, browsing, and viewing has been proposed. This framework considers that a video query system should incorporate certain capabilities such as the search on dynamic visual properties of video and the ability for rapid nonlinear viewing of video. To achieve these capabilities, algorithms have to be developed for automatic extraction of dynamic visual video properties and for processing of video to extract visual features for compact presentation of the video content. Such representations facilitate nonlinear access into video and give quick views of the visual content of video.

Several techniques for detecting shot boundaries were discussed. Some of these techniques are suitable for detecting abrupt transitions where others were aimed at detecting gradual transitions. Simple methods based on pixel differences can be successfully used for detecting cuts only for particular videos where the shot boundaries are clearly delimited. In practice, these methods are not likely to work properly. Camera movement, e.g. pan or zoom, fast moving objects, illumination changes, etc., can change a large number of pixels and consequently can result in detection of a false shot break. Moreover, these techniques are based on a single threshold and therefore, lack the power of detecting gradual transitions. The twin comparison technique was used for detecting gradual transitions but this method cannot be used to detect the type of the transition. In order to achieve this, different transition types such as cuts, fades, wipes, and dissolves were modeled by mathematical functions. However, these models do not take into account perturbing factors such as object and camera motion. Optical flow techniques are utilized for detecting the camera motion. Optical flow works well but may not distinguish between object and camera motion without understanding the content of the scene. Velocity components of the points in an image can be used for segmenting a scene into different moving objects. Using the fact that neighborhood-based color ratios are invariant under various changes in view condition, including both spectral and intensity variations of the illuminant, a window-based video partitioning method based on color ratio was also proposed. Several other techniques were used for detecting shot boundaries in the compressed domain. A problem here is the dependence of the algorithms on the characteristics of

a particular encoder. As a consequence, compressed domain shot boundary detection that works well with video produced by one MPEG encoder could require different parameter settings or might not even find the information it requires in video from a different encoder.

An important aspect is the evaluation of the search results of a video query. Accurate content processing and annotation, whether it is automatic or manual, may not result in the retrieval of the desired segments. Different types of queries can be expected from different types of end users: one who may have seen a particular piece of video and wants to find it or may not have seen it but knows that the video exists; and one who may not have seen the piece but has some vague idea what he or she is looking for. In either case, one could define what it would mean for a retrieved video to be appropriate to the query. For the first case, a candidate can be appropriate only if it is the particular video clip or if the candidate has the clip embedded in it in some form. For the second case, the concept of appropriateness is less clear, and the user must decide if the candidate is what he or she is interested in.

The evaluation of the performance of a video retrieval system is made difficult by the interactiveness of the query process. The performance has to be measured not only in terms of recall and precision, but also in terms of the average amount of time required to select the segments of interest. The discipline of text retrieval has already given some valuable thought to these issues, which will be useful for video-query performance evaluation. Nonetheless, the measures for performance of video query are still open research issues.

The audiovisual features, together with textual descriptions, are integral components of video queries. Analysis of visual content is one step beyond the traditional database query approach. Analysis of audio features is another step forward, and the integration of the available multiple-media features is the ultimate step toward the successful deployment of video queries in multimedia database systems.

## References

1. "Britannica Online" <http://www.eb.com:180>.
2. Adjeroh, D, Lee, M, and Orji, C, "Techniques for Fast Partitioning of Compressed and Uncompressed Video," *Multimed Tools Applic*, 4(3), pp. 225-243, 1997.
3. Ahanger, A and Little, T, "A Survey of Technologies for Parsing and Indexing Digital Video," *J Visual Commun Image Represent*, 7(1), pp. 28-43, 1996.
4. Aigrain, P and Joly, P, "The Automatic Real-Time Analysis of Film Editing and Transition Effects and Its Applications," *Computers Graphics*, 18(1), pp. 93-103, 1994.
5. Arman, F, Hsu, A, and Chiu, M, "Image Processing on Encoded Video Sequences," *Multimed Syst*, pp. 211-219, 1994.
6. Bolle, R, Yeo, B, and Yeung, M, "Video Query: Research Directions," *Multimed Syst*, 42(2), 1998.

7. Chen, J, Taskiran, C, Delp, E, and Bouman, C, "ViBE: A New Paradigm for Video Database Browsing and Search," IEEE Workshop on Content-Based Access of Image and Video Database, 1998.
8. Corridoni, JM and Del Bimbo, A, "Structured Digital Video Indexing," Int. Conf. Pattern Recognition, 1996.
9. Davenport, G, Smith, TA, and Pincever, N, "Cinematic Primitives for Multimedia," IEEE Computer Graphics Applic, 6(5), pp. 67-74, 1991.
10. DeMethon, D, Kobla, V, and Doermann, D, "Video Summarization by Curve Simplification," ACM Multimed, 1998.
11. Ding, W, Marchionini, G, and Tse, T, "Previewing Video Data: Browsing Key Frames at High Rates," Int. Symp. on Digital Libraries, 1997.
12. Flicker, M, Sawhney, H, Niblack, W, Ashley, J, Huang, Q, Dom, B, Gorkani, M, Hafner, J, Lee, D, Petkovic, D, Steele, D, and Yanker, P, "Query by Image and Video Content: The QBIC System," IEEE Computer, 28(9), pp. 23-32, 1995.
13. Gauch, S, Gauch, J, and Pua, K, "VISION: A Digital Video Library," ACM Digital Libraries, 1996.
14. Hampapur, A, Digital Video Indexing in Video Databases, PhD Thesis, University of Michigan, 1994.
15. Hampapur, A, Jain, R, and Weymouth, T, "Digital Video Segmentation," ACM Multimedia, pp. 357-364, 1994.
16. Idris, F and Panchanathan, S, "Indexing of Compressed Video Sequences," Proc. SPIE 2670: Storage and Retrieval for Image and Video Databases IV, pp. 247-253, 1996.
17. Idris, F and Panchanathan, S, "Review of Image and Video Indexing Techniques," J Visual Commun Image Represent, 8(2), pp. 146-166, 1997.
18. Jain, RC, "Segmentation of Frame Sequence Obtained by a Moving Observer," IEEE Trans Patt Anal Mach Intell, 6(5), pp. 624-629, 1984.
19. Jain, R and Hampapur, A, "Metadata in Video Databases," ACM SIGMOD, 23(4), 1994.
20. Kuo, T, Lin, Y, Chen, A, Chen, S, and Ni, C, "Efficient Shot Change Detection on Compressed Video Data," Proc. Int. Workshop on Multimedia Database Management Systems, 1996.
21. Lee, J and Dickinson, BW, "Multiresolution Video for Subband Coded Video Databases," Proc. SPIE: Storage and Retrieval for Image and Video Databases, 1994.
22. Lee, SY and Kao, HM, "Video Indexing: An Approach Based on Moving Object and Track," Proc. SPIE 1908: Storage and Retrieval for Image and Video Databases, pp. 25-36, 1993.
23. Little, T, Ahanger, G, Folz, R, Gibbon, J, Reeves, F, Schelleng, D, and Venkatesh, D, "A Digital On-Demand Video Service Supporting Content-based Queries," ACM Multimedia, pp. 427-433, 1993.
24. Liu, HC and Zick, GL "Scene Decomposition of MPEG Compressed Video," Proc. SPIE 2419: Digital Video Compression, pp. 26-37, 1995.
25. Mandal, M, Idris, F, and Panchanathan, S, "A Critical Evaluation of Image and Video Indexing Techniques in the Compressed Domain," Image Vision Comput, 17, pp. 513-529, 1999.
26. Meng, J, Juan, Y and Chang, SF, "Scene Change Detection in a MPEG Compressed Video Sequence," Proc. SPIE 2419: Digital Video Compression, pp. 267-272, 1995.
27. Mohan, R, "Text Based Indexing of TV News Stories," Proc. SPIE 2916: Multimedia Storage and Archiving Systems, 1996.

28. Nagasaka, A and Tanaka, Y, "Automatic Video Indexing and Full-Video Search for Object Appearances," *Visual Database Systems II*, pp. 113–127, 1992.
29. Otsuji, K, Tonomura, Y, and Ohba, Y, "Video Browsing using Brightness Data," *Proc SPIE 1606: Visual Communications and Image Processing*, pp. 980–989, 1991.
30. Patel, N and Sethi, I, "Video Shot Detection and Characterization for Video Databases," *Patt Recogn*, 30(4), pp. 583–592, 1997.
31. Poncelon, D, Srinivasan, S, Amir, A, and Petkovic, D, "Key to Effective Video Retrieval: Effective Cataloging and Browsing," *ACM Multimed*, 1998.
32. Rowe, LA, Boreczky, JS, and Eads, CA, "Indices for User Access to Large Video Databases," *Proc. SPIE 2185: Storage and Retrieval for Image and Video Databases II*, 1994.
33. Sawhney, H and Ayer, S, "Compact Representations of Videos Through Dominant and Multiple Motion Estimation," *IEEE Trans Patt Anal Mach Intell*, 18(8), pp. 814–830, 1996.
34. Sawhney, N, Balcom, D, and Smith, I, "Authoring and Navigating Video in Space and Time," *IEEE Multimed J*, 1997.
35. Shahraray, B and Gibbon, D, "Automatic Generation of Pictorial Transcripts of Video Programs," *Proc. SPIE 2417: Multimedia Computing and Networking*, 1995.
36. Shahraray, B, "Scene Change Detection and Content-Based Sampling of Video Sequence," *Proc. SPIE 2419: Digital Video Compression*, pp. 2–13, 1995.
37. Shen, B, Li, D, and Sethi, IK, "Cut Detection via Compressed Domain Edge Extraction," *IEEE Workshop on Nonlinear Signal and Image Processing*, 1997.
38. Szeliski, R, "Image Mosaicking for Tele-Reality Applications," *ACM Multimedia*, 1993.
39. Taniguchi, Y, Akutsu, A, and Tonomura, Y, "PanoramaExcerpts: Extracting and Packing Panoramas for Video Browsing," *ACM Multimedia*, 1997.
40. Taskiran, C and Delp, E, "Video Scene Change Detection Using the Generalized Trace," *IEEE Int. Conf. on Acoustic, Speech and Signal Processing*, pp. 2961–2964, 1998.
41. Teodosio, L and Bender, W, "Salient Video Stills: Content and Context Preserved," *TR. DEC Cambridge Research Laboratory*, 1994.
42. Tonomura, Y, Akutsu, A, Otsuji, K, and Sadakata, T, "VideospaceIcon: Tools for Anatomizing Video Content," *ACM INTERCHI*, 1993.
43. Yeo, BL and Liu, B, "Rapid Scene Analysis on Compressed Video," *IEEE Trans Circuits Syst Video Technol*, 5(6), pp. 533–544, 1995.
44. Yeung, MM and Liu, B, "Efficient Matching and Clustering of Video Shots," *International Conf. Image Processing*, pp. 338–341, 1995.
45. Yeung, MM, Yeo, B, Wolf, W, and Liu, B, "Video Browsing Using Clustering and Scene Transitions on Compressed Sequences," *Proc. SPIE 2417: Multimedia Computing and Networking*, pp. 399–413, 1995.
46. Yeung, MM and Yeo, B, "Video Visualization for Compact Presentation of Pictorial Content," *IEEE Trans Circuits Syst Video Technol*, 7(5), pp. 771–785, 1997.
47. Yeung, MM and Yeo, B, "Video Content Characterization and Compaction for Digital Libraries," *Proc. SPIE 3022: Storage and Retrieval for Image and Video Databases V*, pp. 45–58, 1997.
48. Zabih, R, Miller, J, and Mai, K, "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks," *ACM Multimed*, 1995.
49. Zhang, HJ, Kankanhalli, A, and Smoliar, SW, "Automatic Partitioning of Full-Motion Video," *Multimed Syst*, 3(1), pp. 10–28, 1993,

50. Zhang, HJ, Tan, Y, Smoliar, SW, and Gong, Y, "Automatic Parsing and Indexing of News Video," *Multimed Syst*, 6(2), pp. 256-266, 1995.
51. Zhang, HJ, Low, CY, and Smoliar, SW, "Video Parsing and Browsing using Compressed Data," *Multimed Tools Applic*, 1, pp. 89-111, 1995.