CHAPTER 5.5

VIDEO CONTENT ANALYSIS AND RETRIEVAL

HONGJIANG ZHANG

*Hewlett-Packard Laboratories, 1501 Page Mill Road
Palo Alto, California 94304, USA
E-mail: hjzhang@hplhjl.hpl.hp.com*

This chapter discusses a number of available techniques and state-of-the-art research issues in video content analysis and retrieval. It focuses on basic algorithms for video structure parsing, content representation, content-based abstraction and applications tools for content-based video indexing, retrieval and browsing.

Keywords: Video content analysis, video partition, visual abstraction, content-based indexing, content-based retrieval and browsing, digital video, multimedia, digital library.

1. Background and Motivations

With rapid advances in communication and multimedia computing technologies, accessing a vast amount of visual data is becoming a reality on information super-highways and in digital libraries. Though the most important investments have been targeted at the information infrastructure, it has been realized that video content analysis and processing are key issues in putting together successful applications [1,2]. The matter of fact is that interacting with multimedia data, video in particular, requests much more than just connecting users with data banks and delivering data via networks to customers' homes or offices. It is simply not enough to just store and display video as in commercial video-on-demand services. The need, from the point of view of content, is that the visual data resources managed by such systems have to be structured and indexed based on the content before being accessed. However, when it is possible, the human production of such content descriptive data is so time consuming — and thus costly — that it is almost impossible to generate it for the vast amount of visual data available. Further, human-produced descriptive data is often subjective, inaccurate and incomplete. Thus, fundamentally, what we need are new technologies for video content analysis and representation to facilitate organization, storage, query and retrieval of mass collections of video data in a user-friendly way.

When text is indexed, words and phrases are used as index entries for sentences, paragraphs, pages or documents. Similarly, video indexing will require partitioning of video documents into shots and scenes and extracting key frames or key sequences as entries for scenes or stories. Therefore, automated indexing of video will require

the support of tools that can *detect* such meaningful segments, and *extract* content features of any video source. Figure 1 shows a system diagram of such a video content analysis process consisting of three major steps.

- (1) **Parsing:** This process will partition a video stream into generic clips of different levels of granularity and extract structural information of video. These clips will be the units for representation and indexing.
- (2) **Content analysis, abstraction and representation:** Ideally, individual clips will be decomposed into *semantic primitives* based on which a clip can be represented and indexed with a *semantic description*. In practice, the abstraction process will generate a visual abstract of clips and low-level visual features of clips will be used to represent their visual content.
- (3) **Retrieval and browsing:** Indices are built based on content primitives or meta- data through, for instance, a clustering process which classifies shots into different visual categories. Schemes and tools are needed to utilize these content representation and indices to query, search and brows large video databases for retrieving desired video clips.

Fig. 1. Process diagram for video content analysis and retrieval.

The temporal segmentation process is analogous to sentence segmentation and paragraphing in parsing textual documents, and many effective algorithms are now available for temporal segmentation [4–13], as to be described in detail later in Section 2. Since fully automated visual content understanding or mapping low level visual features to high level semantic content is not feasible in the near future, video content representation will be mostly based on low-level content features. Such retrieval approaches using low-level visual features as content representation have shown a great potential for retrieval of images in large image databases [14].

While we tend to think of indexing for supporting retrieval, browsing is equally significant for video source material since the volume of video data also requires techniques to present information landscape or structure to give an idea of what is out there. The task of browsing is actually very intimately related to retrieval. On the one hand, if a query is too general, browsing is the best way to examine the results. Hence, browsing also serves as an aid to formulate queries, making it easier for the user to *just ask around* in the process of figuring out the most appropriate

query to pose. A truly content-based approach to video browsing also requires some level of analysis of video content, both structural and semantic, rather than simply providing a more sophisticated view of temporal context.

The need for video content analysis tools as summarized above poses many research challenges to scientists and engineers across all multimedia computing disciplines. In this chapter, we will try to discuss the approaches to visual content analysis, representation and their applications and to survey some open research problems. Section 2 covers video content parsing algorithms and schemes, including temporal segmentation, video abstraction, shot comparison and soundtrack analysis. Section 3 presents briefly video content representation schemes in terms of visual features, objects and motions. Section 4 describes tools for content-based video retrieval and browsing and Section 5 reviews some research prototypes and application systems. Section 6 summarizes some current research issues.

2. Temporal Partition of Video Sequences

The basic temporal unit for indexing and manipulation of video is shots, consisting of one or more frames recorded contiguously and representing a continuous action in time and space. A collection of one or more adjoining shots that focus on an object or objects of interest may comprise a scene. *Temporal partitioning* is the process of detecting boundaries between every two consecutive shots, so that a sequence of frames belonging to a shot will be grouped together. Hence, temporal partitioning is the first step in parsing video, and has been one of the first issues addressed by many researchers in video content analysis.

The temporal partitioning problem is also important for applications other than video indexing and editing. It also figures in performing motion compensated video compression, where motion vectors must be computed *within* segments, rather than *across* segment boundaries [15]. However, accuracy is not as crucial for either of these applications; for example, in compression, false positives only increase the number of reference frames. On the other hand, in video segmentation for indexing, such false positives would have to be corrected by manual intervention. Therefore, high accuracy is a more important requirement in automating the partitioning process for video content analysis.

There are a number of different types of transitions or boundaries between shots. The simplest transition is a *cut*, an abrupt shot change which occurs between two consecutive frames. More sophisticated transitions include *fades*, *dissolves* and *wipes*, etc. [16]. A *fade* is a slow change in brightness of images usually resulting in or starting with a solid black frame. A *dissolve* occurs when the images of the first shot get dimmer and the images of the second shot get brighter, with frames within the transition showing one image superimposed on the other. A *wipe* occurs when pixels from the second shot replace those of the first shot in a regular pattern such as in a line from the right edge of the frames. A robust partitioning algorithm should be able to detect all these different boundaries with good accuracy.

The basis of detecting shot boundaries is the fact that consecutive frames on either side of a boundary generally display a significant change in content. Therefore, what is required is some suitable quantitative measure which can capture the quantitative difference between such a pair of frames. Then, if that difference exceeds a given threshold, it may be interpreted as indicating a shot boundary. Hence, establishing suitable metrics is the key issue in automatic partitioning. The optimal metric for video partitioning should be able to detect the following three different factors of image change:

- Shot change, abrupt or gradual;
- Motion, including those introduced by both camera operation and object motion;
- Luminosity changes and noise.

In this section, a number of algorithms for temporal partitioning of video data either in its original format or in compressed domain representations will be presented.

2.1. Shot Cut Detection Metrics using Original Format of Digital Video Data

Figure 2 illustrates a sequence of three consecutive video frames with a cut occurring between the second and third frames. The significant difference in content is readily apparent. If that difference can be expressed by a suitable metric, then a segment boundary can be declared whenever that metric exceeds a given threshold. The major difference among a variety of automatic video partitioning is the difference metrics used to quantitatively measure changes between consecutive frames and schemes to apply these metrics.

Difference metrics used in partitioning can be divided into two major types: those based on local pixel feature comparison, such as pixel values and edges, and those based on global features such as pixel histograms and statistical distributions of pixel-to-pixel change. These types of metrics may be implemented with a variety of different modifications to accommodate the idiosyncrasies of different video sources and have been successfully used in shot boundary detection.

Fig. 2. Three frames across a sharp cut.

2.1.1.1. Pixel value comparison

A simple way to detect a qualitative change between a pair of frames is to compare the spatially corresponding pixels in the two frames to determine how many have changed. This approach is known as *pair-wise pixel comparison*. In the simplest case of monochromatic images, a pixel is judged as changed if the difference between its intensity values in the two frames exceeds a given threshold t . This algorithm simply counts the number of pixels changed from one frame to the next. A shot boundary is declared if more than a given percentage of the total number of pixels (given as a threshold T) have changed.

A potential problem with this metric is its sensitivity to camera movement. For instance, in the case of camera panning, a large number of objects will move in the same direction across successive frames; this means that a large number of pixels will be judged as changed even if the pan entails a shift of only a few pixels.

To make the detection of camera breaks more robust, instead of comparing individual pixels, we can compare corresponding *regions* (blocks) in two successive frames. One such approach applies motion compensation at each block [8]. That is, each frame is divided into a small number (e.g. 12 in [8]) of non-overlap blocks. Block matching within a given search window is then performed to generate a motion vector and match value, normalized to lie in the interval $[0, 1]$ with zero representing a perfect match. A common block matching approach used in video coding is that for each block in frame j , find the best fitting region in image $(j + 1)$ in a neighborhood of the corresponding block according to the matching function

$$\text{Diff} = \frac{1}{F_{\max}} \frac{1}{x} \frac{1}{mn} \sum_{l=-n/2}^{n/2} \sum_{k=-m/2}^{m/2} |F_j(k, l) - F_{j+1}(k + dx, l + dy)| \quad (2.1)$$

where:

$F_j(k, l)$ represents the value at pixel (k, l) of a $(m \times n)$ block in the current frame j ;

$F_{j+1}(k, l)$ represents the value at pixel (k, l) of the same $(m \times n)$ block in the next frame $(j + 1)$;

(dx, dy) is a vector representing the search location and the search space is $dx = \{-p, +p\}$ and $dy = \{-p, +p\}$;

$F_{\max}$ is the maximum pixel value of frames and is used to normalized the matching function.

The set of (dx, dy) that produces the minimum value of Diff defines the motion vector from the center of block i in the current frame to the next frame.

Then, the difference between two frames can be defined as

$$D_M = \sum_{i=1}^K c_i R_i \quad (2.2)$$

where i is the block number in a frame, K is the total number of blocks, R_i is the match value for block i which equals to the minimum Diff in a given searching space, and c_i is a set of predetermined weights for each block. To eliminate the effect of noise, the 2 highest and 2 lowest match values are discarded in calculating D_m . A cut is declared if D_m exceeds a given threshold.

2.1.2. Histogram comparison

An alternative to comparing corresponding pixels or blocks in successive frames is to compare some statistic and global features of the entire image. One such feature is the histogram of intensity levels. The principle behind this algorithm is that two frames having an unchanging background and objects will show little difference in their respective histograms. The histogram comparison algorithm should be less sensitive to object motion than the pair-wise pixel comparison algorithm, since it ignores the spatial changes in a frame.

Let $H_f(i)$ denote the histogram value for frame f , where i is one of the G possible pixel levels, then, the difference between frame f and its successor ($f + 1$) may be given by the following χ^2 -test formula,

$$D_h = \frac{1}{G} \sum_{i=1}^G \frac{|H_f(i) - H_{f+1}(i)|^2}{(H_f(i) + H_{f+1}(i))^2}. \quad (2.3)$$

If the overall difference D_h is larger than a given threshold T , a segment boundary is declared. To be more robust to noise, each frame is divided into a number of regions of same size, e.g. 16 regions. That is, instead of comparing global histograms, histograms for corresponding regions in the two frames are compared and the 8 largest differences are discarded to reduce the effects of object motion and noise [4].

2.1.3. Edge pixel comparison

Edges in images provide useful information about the image content and changes in edge distributions between successive frames is a good indication of content changes. When a cut or a graduation transition occurs, new intensity edges appear far from the locations of old edges and similarly, old edges disappear far from the locations of new edges. Based on this observation, an effective video partitioning algorithm which can detect both cuts and gradual transitions has been developed as described below [9].

First, we define an edge pixel in a current frame that appears far from (a distance r) an existing edge pixel in the last frame as an *entering* edge pixel, and an edge pixel in the last frame that disappears far from an existing edge pixel in the current frame as an *exiting* edge pixel. Then, by counting the fraction of the entering edge pixels (p_{in}) and that of the exiting edge pixels (p_{out}) over the total number of pixels in a frame, respectively, we can detect transitions between two shots. That is, the difference metric between frames f and $f + 1$ can be defined as:

$$D_e(f, f + 1) = \max(p_{in}, p_{out}). \quad (2.4)$$

$D_e(f, f + 1)$ will assume a high value across shot boundaries and generate peaks in the time sequence. Once a peak is detected, it can be further classified as corresponding to a cut or gradual transitions since cuts usually correspond to sharp peaks occurring over 2 or 3 frames, while gradual transitions usually correspond to low but wide peaks over a larger number of consecutive frames. Experiments have shown that this algorithm is very effective in detecting both sharp cuts and gradual transitions. However, since it requires more computation, this algorithm is slower than others.

2.2. Gradual Transition Detection using Original Format of Digital Video Data

Cuts are the simplest shot boundary and easy to detect using the difference metrics described above. Figure 3 illustrates a sequence of inter-frame difference resulting from the histogram comparison. It is easy to select a suitable cutoff threshold value (such as 50) for detecting the two cuts represented by the two high pulses. However, sophisticated shot boundaries such as dissolve, wipe, fade-in, and fade-out are much more difficult to detect since they involve more gradual changes between consecutive frames than does a sharp cut. Furthermore, changes resulting from camera operations may be of the same order as that from gradual transitions, which further complicate the detecting.

Figure 4 shows five frames from a typical dissolve: the last frame of the current shot just before the dissolve begins, three frames within the dissolve, and the frame in the following shot immediately after the dissolve. The actual dissolve occurs across about 30 frames, resulting in small changes between every two consecutive frames in the dissolve. The sequence of inter-frame differences of this dissolve defined by the histogram calculation is displayed in the inset of the graph shown in Fig. 3, whose values are higher than those of their neighbors but significantly lower than the cutoff threshold. This sequence illustrates that gradual transitions will downgrade the power of a simple difference metric and a single threshold for camera break detection algorithms.

Fig. 3. A sequence of histogram based inter-frame differences.

Fig. 4. An example of dissolve sequences.

The simplest approach to this problem would be to lower the threshold. Unfortunately, this cannot be effectively employed, because noise and other sources of changes often introduce the same order of difference between frames as of gradual transitions, resulting in “false positives.” In this subsection, we discuss four video partition algorithms which are capable of detecting gradual transitions with acceptable accuracy while achieving very high accuracy in detecting sharp cuts. They are: Zhang’s *twin comparison algorithm*, Agrain’s algorithm based on *distribution of pixel change*, Zabih’s *edge comparison* algorithm and Hampapur’s *editing model* based algorithm.

2.2.1. *Twin-comparison approach*

This algorithm was the first published one which achieves high accuracy in detecting both cuts and gradual transitions. As shown in Fig. 4 it is obvious that the first and the last frame across the dissolve are different, even if all consecutive frames are very similar. In other words, the difference metric with the threshold as shown in Fig. 3 would still be effective as it was applied to the comparison between the first and the last frames directly. Thus, the problem becomes one of detecting these first and last frames. If they can be determined, then the period of gradual transition can be isolated as a segment unto itself. If we look at the inset of Fig. 3, it can be noticed that the difference values between most of the frames during the dissolve (as well as wipes and fades) are higher, although only slightly, than those in the preceding and following segments. What is required is a threshold value which can detect this *sequence* and distinguish it from an ordinary camera shot. Based on this observation, the *twin comparison* algorithm was developed which introduces two comparisons with two thresholds.

The algorithm uses two thresholds: T_b for sharp cut detection in the same manner as was described in the last subsection; and the second and lower threshold T_s is introduced for gradual transition detection. As illustrated in Fig. 5, whenever the

Fig. 5. Twin-comparison approach.

difference value exceeds T_b , a camera cut is declared, e.g. F_B in Fig. 5. However, the twin-comparison also detects differences which are smaller than T_b but larger than T_s . Any frame which exhibits such a difference value is marked as the potential start (F_s) of a gradual transition. Such a frame is labeled in Fig. 5. This frame is then compared against subsequent frames, which is called an *accumulated comparison*. The end frame (F_E) of the transition is detected when the difference between consecutive frames falls below threshold T_s while the accumulated difference exceeds T_b . Note that the accumulated comparison needs only to be computed when the difference between consecutive frames exceeds T_s . If the consecutive difference value drops below T_s before the accumulated comparison value exceeds T_b , then the potential start point is dropped and the search continues for other gradual transitions.

A potential problem with this algorithm is that camera panning and zooming and large object motion may introduce similar gradual changes as gradual transitions, which will result in “false positives”. This problem can be solved by global motion analysis as presented later in Section 2.4. That is, every potential transition sequence detected will be passed to a motion analysis process to further verify if it is actually a global motion sequence [5]. Experiments show that the twin comparison algorithm is very effective and achieves a very high level of accuracy.

2.2.2. Pixel change classification

Another algorithm for detecting both cuts and gradual transitions is derived based on the statistic of pixel value changes between two consecutive frames [6]. It is assumed that the inter-frame pixel values change due to a combination of sources: First, a small amplitude additive zero-mean Gaussian noise, modeling camera, tape and digitization noise sources; second, changes of pixels resulting from object or camera operation and lighting within a given shot; and third, changes caused by cuts and gradual transitions. According to analytical models for each of these

changes, cuts can be found by looking at the number of pixels whose change of value between two consecutive frames falls in the range $\{128, 255\}$. Dissolves and fade to/from white/black can be identified by the number of pixels which change in value between two consecutive frames in range $\{7, 40\}$ for 8-bit coded gray-level images. However, changes of pixel value resulting from wipes are also in the range of $\{128, 255\}$; thus, wipes may not be detected reliably. This may be solved by further looking at the spatial distributions of pixels whose range of changes are in $\{128, 255\}$ since during wipes, each frame will have a portion of the current shot and a portion of the new shot, thus, the changes usually occur in the boundary areas of the two portions. Based on these statistics, sharp cuts and gradual transitions can be detected by examining the difference ranges of corresponding pixels in two consecutive frames.

This algorithm is not designed to detect cuts or transitions based on only two consecutive frames, but incorporating a temporal filtering over a sequence of consecutive frames. Also, histogram equalization needs to be applied to frames for wipe and cut detection, which slows down the detection process.

2.2.3. Edge pixel comparison

The edge pixel comparison algorithm as defined by Eq. (2.4) can also be applied in detecting gradual transitions. This is because gradual transitions usually introduce relatively lower but wide peaks of $D_e(f, f')$ values over a number of consecutive frames, different from sharp and narrow peaks for cuts. Fades and dissolves can be distinguished from each other by looking at relative values of local regions. During a fade in, p_{in} will be much higher than p_{out} since there should be more entering edge pixels and fewer exiting edge pixels while a new shot progressively appears into the frames from back. In contrast, during fade out, p_{out} will be much higher than p_{in} since the current shot progressively disappears into the back frames. A dissolve, on the other hand, consists of an overlapping fade in and fade out: during the first half of dissolve, p_{in} will be greater while during the second half, p_{out} will be greater. Wipes can be distinguished from dissolves and fades by looking at the spatial distribution of entering and exiting pixels, since frames of a wipe sequence usually have a portion of the current shot and a portion of the new shot. Therefore, if we take the location into the analysis when calculating the fraction of changed edge pixels, wipes can be detected and distinguished from other types of transitions.

2.2.4. Editing model fitting

Hampapur *et al.* have studied algorithms for detecting different types of gradual transitions by fitting sequences of inter-frame changes to editing models, one for a given type of gradual transition [10]. However, the potential problem with such model based algorithms is that as more and more different types of editing effects (which still fall in mainly three basic classes: dissolve, wipe and fade) become available, it is hard to model each one of them. Furthermore, transition sequences may

not follow any particular editing model, due to possibly noises and/or combination of editing effects. Such problems may exist in other detection algorithms as well, though this particular algorithm may be more prone to this problem.

2.3. Video Partitioning Algorithm using Compressed Domain Representation

As JPEG, MPEG and H.26X [15] have become industrial standards, more and more video data have been and will continue to be stored and distributed in one of these compressed formats. It would, therefore, be advantageous for the tools we envisage to operate directly on compressed representations, saving on the computational cost of decompression. More importantly, the compressed domain representation of video defined by these standards provide features which may be more effective in detecting content changes.

DCT (Discrete Cosine Transform) coefficients are the basic compressed domain features encoded in JPEG, MPEG and H.26X. Another important feature encoded in the latter two standards is motion vectors. These are the two main features to be utilized in algorithms for video partitioning, with the most effective one being the one that combines both features. In this sub-section, we discuss in detail three basic types of algorithms for video partitioning using compressed video data.

2.3.1. DCT coefficient-based comparison

In general, compression of a video frame begins by dividing a frame into a set of 8×8 pixel blocks [15], as shown in Fig. 6. The pixels in the blocks are then transformed by the forward DCT (FDCT) into 64 DCT coefficients. That is,

$$F(u, v) = \frac{1}{4} C(u) C(v) \left[\sum_{x=0}^7 \sum_{y=0}^7 p(x, y) \cdot \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right] \quad (2.5)$$

where $C(k) = 1/\sqrt{2}$ if $k = 0$ and 1 otherwise. $F(u, v)$ are DCT coefficients and $p(x, y)$ is value of pixel (x, y) in a block. $F(0, 0)$ is the DC term or DC coefficient of a

Fig. 6. 8×8 block based DCT and zig-zag encoding.

block, which is the average value of the 64 pixels, and the remaining 63 coefficients are termed the AC coefficients. These DCT coefficients are then quantized and encoded in a zig-zag order by placing the low frequency coefficients before the high frequency coefficients as shown in Fig. 6. The coefficients are finally Huffman entropy encoded. The process can then be reversed for decompression.

Since the DCT coefficients are mathematically related to the spatial domain and represent the content of each frame, they can be used to detect the difference between two video frames. Based on this idea, the first DCT comparison metric for partitioning JPEG videos was developed by Arman *et al.* [11]. In this algorithm, a subset of the blocks in each frame (e.g. take out the boundary blocks of a frame and use every other one of the rest) and a subset of the DCT coefficients for each block were used as a vector representation (V_f) for each frame. That is,

$$V_f = (c_0, c_1, \dots, c_i, \dots, c_m) \quad (2.6)$$

where c_i is the i th coefficients of the selected subset. The members of V_f , are randomly distributed among all AC coefficients. One way to choose the subset is to use every other coefficient. The members of V_f remain the same throughout the video sequence to be segmented.

The difference metric between frames is then defined by content correlation in terms of a normalized inner product:

$$D_{DCTC} = 1 - |V_f \bullet V_{f+\varphi}| / |V_f| |V_{f+\varphi}| \quad (2.7)$$

where φ is the number of frames between the two frames being compared.

It has been observed that for detecting shots boundaries, DC components of DCT's of video frames provide sufficient information [18]. Based on the definition of DCT, this is equivalent to a low resolution version of frames, averaged over 8×8 non-overlap blocks [15]. Applying this idea makes calculation of (2.5) much faster while maintaining similar detection accuracy.

Using DC sequences extracted from JPEG or MPEG data also makes it easy to apply histogram comparison. That is, each block is treated as a pixel with its DCT value as the pixel value, then, histograms of DCT-DC coefficients of frames calculated and compared using metrics (2.3). This algorithm has been approved to be very effective, achieving both high detection accuracy and speed in detecting sharp cuts [19].

The DCT based metrics can be directly applied to JPEG video, where every frame is intra-coded. However, in MPEG, temporal redundancies are reduced by applying block-based motion compensation techniques, while spatial redundancies are reduced by block-based intra-coding as in JPEG. Therefore, as shown in Fig. 6, in MPEG, there are three types of frames used: intra(I)-framed, predicted (P) frames and bi-directional predicated and interpolated (B) frames. Only the DCT coefficients of I frames are transformed directly from original images, while for P and B frames DCT coefficients are in general residual errors from motion compensated prediction. This means that DCT based metrics can only be applied in

comparing **I** frames in MPEG video. Since only a small portion of frames in MPEG are **I** frames, this significantly reduces the amount of processing which goes into computing differences. On the other hand, the loss of temporal resolution between **I** frames will introduce a large fraction of false positives in video partitioning which have to be handled with subsequent processing.

2.3.2. Motion vector based segmentation

Apart from pixel value, motion resulting from either moving objects, camera operations or both represents another important visual content in video data. In general, the motion vectors should show continuity between frames within a camera shot and show discontinuity between frames across two shots. Thus, a continuity metric for a field of motion vectors should serve as an alternative criterion for detecting segment boundaries.

The motion vectors between video frames are in general obtained by block matching between consecutive frames, which is an expensive process. However, if the video data are compressed using either MPEG standards, motion vectors can be obtained from the bit streams of the compressed images. In MPEG data streams, as shown in Fig. 7, there is one set of motion vectors associated with each **P** frame representing the prediction from the last or next **I** frame; and there may be two sets of motion vectors associated with each **B** frame, forward and backward. That is, each **B** frame is predicted and interpolated from its preceding and succeeding **I/P** frames by motion compensation. If there is a significant change (discontinuity) in content between two frames, either two **B** frames, a **B** frame and an **I/P** frame or a **P** frame and an **I** frame, there will be many blocks in the frame in which the residual error from motion compensation is too high to tolerate. For those blocks, MPEG will not apply the motion compensation prediction but instead intra-code them using DCT. Subsequently, there will be either no or only a few motion vectors associated with those blocks in the **B** or **P** frames. Therefore, if there is a

Fig. 7. Frame types in MPEG video.

Fig. 8. Camera break detection based on motion vectors — A sequence of numbers of motion vectors associated with **B** frames from a documentary video compressed in MPEG.

shot boundary falling in between two frames, there will be a smaller number of inter-coded, but a larger number of intra-coded blocks, due to the discontinuity of the content between the frames. Based on this observation, we can detect a shot boundary by counting the number of inter-coded blocks in **P** and **B** frames [12]: That is, if

$$N_{\text{inter}}/N_{\text{intra}} < T_b \quad (2.8)$$

is lower than a given threshold, then a shot boundary is declared between the two frames.

Figure 8 illustrates values of $N_{\text{inter}}/N_{\text{intra}}$ for a video sequence. In this case, the camera cuts are accurately represented as valleys below the threshold level. However, this algorithm may fail to detect gradual transitions because the number of inter-coded blocks in frames during such a transition sequence is often much higher than the threshold.

2.3.3. Hybrid approach

Combining the DCT based and motion based metrics into a hybrid algorithm will improve the detection accuracy as well as processing speed in partitioning MPEG compressed video. That is, DCT-DC histograms of every two consecutive **I** frames are first compared to generate a difference sequence [12]. Since there is a large temporal distance between two consecutive **I** frames, it is assumed that if we set the threshold relatively low, all shot boundaries, including gradual transitions, will be detected by looking at the points where there is a high difference value. Of course, this first pass will also generate some false positives. Then, **B** and **P** frames between two consecutive **I** frames, which have been detected as potentially containing shot boundaries, will be examined in a second pass using motion-based metrics (2.8). That is, the second pass is only applied to the neighborhood of the potential boundary frames. In this way, both high processing speed and detection accuracy

are achieved at the same time. The only potential problem with this hybrid algorithm is that it may detect false motion sequences as transition sequences, just like the twin-comparison algorithm, which requires a motion analysis based filtering process as discussed later.

In summary, experimental evaluations have shown that compression domain feature based algorithms perform with at least the same order of accuracy as those using video data in the original format, though the detection of sharp cuts are more reliable than that of gradual transitions. On the other hand, compression feature based algorithms achieve much high processing speed, which make software only real time video partitioning possible.

2.4. Camera Operation and Object Motion Analysis

Camera operation and object motion are another important source of information and attributes for shot content analysis and classification. It is well known that camera operations or framing of shots are elaborately done by directors/camera operators to present certain scenes or objects and to guide the viewer's attention. Object motions usually represent human activity and major events in video shots. Detection of camera works and object motion need to be performed in at least the following five processes in video content analysis and representation:

- Motion-based temporal partitioning;
- Filtering false positives result from motion in gradual transitions detection;
- Recovering global motion to construct salient stills for representing video contents;
- Selection of key-frames;
- Video content representation and motion based shot retrieval.

The first two applications have already been discussed in detail in the description of motion based video segmentation algorithms in the last section. The other three applications will be described in the next two sections.

The scientific problem of camera work analysis resides in the discrimination between camera work-induced apparent motion and object motion-induced apparent motion, followed by analysis to identify particular camera works and describe object motion. These are classical and unsolved problems in computer vision. However, for our needs in video content analysis and representation, several algorithms can solve this problem with satisfactory accuracy and speed.

Camera works include panning and tilting (horizontal or vertical rotation of camera), zooming (focal length change), in which camera position does not change; and tracking and booming (horizontal and vertical transverse movement of camera), and dollying (horizontal lateral movement of camera) in which position of camera *does* change, as well as combination of these operations. The specific feature which serves to classify camera works is the *motion field*, as each particular camera operation will result in a specific pattern of motion field [6]. Based on this, Zhang *et al.*

have developed a simple, yet effective approach to camera operation analysis which distinguishes the gradual transition sequences and classifies camera pan and zoom operations [6].

A more sophisticated quantitative approach to detecting camera operation uses the transformation relation between a point in the space and its coordinates in image space [20]. The transformation is then used to derive the pixel coordinate changes when there is a pan or zoom. A simple zoom is caused by a change in the camera's focal length and there is no camera body movement. Camera pan, on the other hand, is caused by rotating the camera about an axis parallel to the image plane. The combination effects of panning and zooming can be expressed as

$$\mathbf{U}' = f_z \mathbf{U} + \mathbf{p} \quad (2.9)$$

where f_z is called the *zoom factor*, and $\mathbf{p}$ the *pan vector*.

We can derive f_z and $\mathbf{p}$ using a motion field calculated from two frames and an iterative algorithm [20]. That is,

$$f_z = \frac{\langle \mathbf{U}'_i, \mathbf{U}_i \rangle - \frac{1}{N} \langle \sum_i \mathbf{U}'_i, \sum_i \mathbf{U}_i \rangle}{\sum_i \langle \mathbf{U}_i, \mathbf{U}'_i \rangle - \frac{1}{N} \langle \sum_i \mathbf{U}_i, \sum_i \mathbf{U}'_i \rangle} \quad (2.10)$$

$$\mathbf{p} = \frac{1}{N} \left(\sum_i \mathbf{U}'_i - f_z \sum_i \mathbf{U}_i \right). \quad (2.11)$$

This approach not only detects camera pans, zooms, and combinations but also describes them quantitatively. The price paid for this information is a significant increase in computation time.

We can combine the above two algorithms together, meaning that the first algorithm is used to detect a potential camera operation while the second algorithm is applied only to the potential frames. A limitation of the two algorithms or their combination is that when a sequence of frames is covered by a single moving object, a panning will be detected falsely.

More sophisticated motion detection algorithms for video content parsing include those based on discrete tomography for camera work identification [21] and visual icon construction [22]. The distribution of the angles of edges in 3-D tomography images resulting from video can be matched to camera work models, and camera motion classification and temporal segmentation can be obtained directly. Discrimination between pan and lateral traveling and between zoom and boom-ing can be achieved only through a complete projective model including parallax analysis [21].

3. Visual Abstraction of Video

Considering the large amount of data video, it is critical to offer means for quick relevance assessment of video documents. How can we spend only few minutes to

view an hour of video and still have a fairly correct perception of its contents? In other words, how can we map an entire segment of video to a small number of representative frames or images? Obviously, what we need is a representation to present information on landscape or structure of video in a more abstracted manner. We call this the *video content abstracting* problem. In this section, we discussed three approaches to visual abstraction of video data: key-frames, video icons and skimmed highlights.

3.1. Key-Frame Extraction

Key-frames are still images extracted from original video data which best represent the content of video shots in an abstract manner. Key-frames have been frequently used to supplement the text of a video log [23], but there has been little work done in identifying them automatically. Apart from browsing, key-frames can also be used in representing video in retrieval: video index may be constructed based on visual features of key-frames, and queries may be directed at key-frames using query by image content techniques [25,26]. In some prototype systems and commercial products, the first frame of each shot has been used as the only key-frame to represent the shot content. However, while such a representation does reduce the data volume, its representation power is very limited since it often does not give sufficient clue as what actions are presented by a shot, except for shots with no change or motion.

Key-frame based representation views video abstraction as a problem of mapping an entire segment (both static and motion content) to some small number of representative images. The challenge is that the extraction of key-frames needs to be automatic and content based so that they maintain the important content of the video while removing all redundant information. In theory semantic primitives of video, such as interesting objects, actions and events should be used. However, such general semantic analysis is not currently feasible, especially when information from soundtracks and/or close caption is not available. In practice, we have to rely on low-level image features and other readily available information instead. An approach to key-frame extraction based on low-level video features has been proposed by Zhang *et al.* [12,26,27]. This approach determines a set of key-frames for each shot according to the following steps.

Segementation: Key-frames will be extracted at shot level based on features and content information of a shot. Given a shot, the first frame will always be selected as the first key-frame; but, whether more than one key-frame needs to be chosen will depend on the following two criteria.

Color feature based frame comparison: After the first key-frame is selected, following frames in the shot will be compared against the last key-frame sequentially as they are processed, based on their similarities defined by a color histogram. If a significant content change occurs between the current frame and the last

key-frame, the current frame will be selected as a new key-frame. Such a process will be iterated until the last frame of the shot is reached. In this way, any significant action in a shot will be captured by a key-frame, while static shots will result in only one key-frame.

Motion based selection: Dominant or global motion resulting from camera operations and large moving objects are the most important source of content changes and thus an important input for key-frame selection. Color histogram representation often does not capture such motion quantitatively, due to its insensitivity to motion. For key-frame extraction, it is necessary to detect sequences involving two types of global motions: panning and zooming. In reality, there are more than just these two types of motion in video sequences, such as tilting, tracking, and dolly-ing. But, due to their similar visual effects, at least in key-frame selection, camera panning, tilting and tracking, as well as horizontal and vertical motion of large objects are treated as one type of motion: *panning-like*. Similarly, camera zooming and dollying and perpendicular (to the imaging plan) motion of large objects are treated as another type: *zooming-like*.

To select key-frames representing these two types of motion, there are two criteria: For a zooming-like sequence, at least two frames will be selected — the first and last frames since one will represent a global, while the other will represent a more focused view. For a panning-like sequence, the number of frames to be selected will depend on the scale of panning: ideally, the spatial context covered by each frame should have little overlap, or each frame should capture a different, but sequential part of object activities. The overlap ratio can be varied to determine the density of key-frame selection.

Figure 9 shows an example in which three key-frames from a tilt-up shot were extracted automatically. It is obvious that from the three key-frames one can see clearly that it is a tilt-up sequence, which is impossible to see from any single key-frame. In this respect, extracting three key-frames is a more adequate abstraction than only a single key-frame. Although choosing any one of the 3 key-frames will capture the visual content of the shot: *business district buildings*, it will not be able to show the camera movement which is important for users (especially producers and editors) who want to choose some shots from stock footages.

Fig. 9. Examples of key-frame extracted automatically from a shot.

3.2. Video Icon

Another effective way of representing shot content of video for browsing is using static icons, which has attracted much research work, with two major approaches:

- Construction of a visual icon based on a key-frame, supplemented with pseudo-depth for the representation of the duration of the shot, and perhaps arrows and signs for the representation of object and camera motion;
- Synthesis of an image representing the global visual contents of the shot.

The first approach has been favored when the emphasis is on building a global structured view of a video document, fitted for quick visual browsing, such as in the IMPACT system [5]. Some researchers have used icon spacing or image size instead of pseudo-depth for representing the duration of the shot, but this does result in an efficient screen space use.

Teodosio and Bender [6] have proposed methods for the automatic construction of an overview image representing the visible contents of an entire shot. Using camera work analysis and the geometrical transformations associated with each camera motion, successive images are mapped into a common frame, and the synthetic image is progressively built. This image is not generally rectangular. Recently, Irani *et al.* have perfected this type of method on two points [7]:

- They use a more complete projective model, including parallax;
- They have shown that it is possible to compute what they called dynamic mosaic images with emphasis being given to the moving parts of the image (action) instead of background oriented images.

The resulting images have been termed *salient stills* [26], *mosaic images* [27], *micon* (motion icon) or *VideoSpaceIcons* [22].

3.3. Video Skimming

Video skimming is the scheme for answering the request of abstracting an hour of video, for instance, into 5-minute highlights with a fair perception of the video contents. This is a relatively new research area and requires a high level of content analysis. A successful approach is to utilize information from text analysis of the video soundtrack. Researchers [e.g. [30]] working on documents with textual transcriptions have suggested producing video abstracts by first abstracting the text using classical text skimming techniques and then looking for the corresponding parts in the video.

A successful application of this type of approach has been the *Informedia* project, in which text and visual content information are fused to identify video sequences that highlight the important contents of video [29]. More specifically, low-level and

mid-level visual features, including shot boundary, human face, camera and object motion and subtitles of video shots are integrated with keywords, spotted from text obtained from close caption and speech recognition, using the following procedures:

- Keyword selection using the well-known TF-IDF technique to skim audio;
- Sequence characterization by low-level and mid-level visual features;
- Selecting a number of keywords according to the required skimming factor;
- Prioritizing image sequences located in close proximity to each selected keyword:
 - (1) Frames with faces or text;
 - (2) Static frames following camera motion;
 - (3) Frames with camera motion and human faces or text;
 - (4) Frame at the beginning of the scene;
- Composite a skimmed highlight sequence with selected frames.

Experiments using this skimming approach have shown impressive results on limited types of documentary video which have very explicit speech or text (close caption) contents, such as education video, news or parliamentary debates [31]. However, satisfactory results may not be achievable using such a text (keyword) driven approach to other videos with a soundtrack containing more than just speech, or stock footage without soundtrack.

4. Video Content Representation and Similarity

After partitioning and abstraction, the next step in video analysis is to identify and compute representation primitives, based on which the content of shots can be indexed, compared, and classified. Ideally these should be semantic primitives that a user can employ to define *interesting* or *significant* events. These semantic primitives include constituent objects' names, appearance and motion, as well as relationships among different objects at different times and the contributions of all these attributes and relationships to the story being presented in a video sequence [32]. However, automatic extraction of such primitives is not feasible; so that we have to build content representation based on low-level features, such as color, texture and motion statistics of shots.

The first set of low-level visual primitives for video content representation should be extracted from key-frames. However, such a representation alone will be insufficient to support event-based classification and retrieval, since key-frame based features capture most the spatial information while motion is an essential and unique feature of video. Therefore, the second type of primitives should be based on temporal variation and motion information in shots. With these two types of representations, we can then index video shots and define shot similarity used in video retrieval.

4.1. Key-Frames Based Features for Shot Content Representation

Key-frames based representation of video content uses the same features as those for content based still image retrievals. These features include *color*, *texture*, and *shape*, which may be defined in different formats and extracted by different operations [3,14,31].

Color features: Color has been one of the first choice for image content representation and similarity measuring since it has excellent discrimination power in measuring image similarity (See Chap. 2.4 for a detailed discussion): It is very rare that two images of totally different objects will have similar colors [32]. The popular representation schemes for color include histogram [14,32,33], dominant colors [32,33], and statistical moments [34]. To make the representation effective and invariant to illumination conditions, different color spaces have been evaluated. It is concluded that $L^*u^*v^*$ color space tends to be the best one [33]. Also, it is noticed that a small number of color ranges capture the majority of pixels in most of the images, thus, a few dominant colors lead to a good approximated representation of color distribution.

Texture features: Texture has long been recognized as being as important a property of images as is color, if not more so, since textural information can be conveyed as readily with gray-level images as it can in color. A detailed discussion on definition and calculation of a variety of texture measures is given in Chap. 2.1. Among many alternatives, the most popular and effective texture models used in image retrieval are: Tamura features (contrast, directionality, and coarseness) [35] and the Simultaneous Auto-regressive (SAR) model [36]. To define a rich perceptive space, Picard and Liu [37] have shown that it is possible to do so by using the Wold decomposition of the texture considered as a luminance field. One gets three components (periodic, evanescent and random) corresponding to the bi-dimensional periodicity, mono-dimensional orientation, and complexity of the analyzed texture.

Shape features: Dominant objects in key-frames represent important semantic content and are best represented by their shapes, if they can be identified by either automatic or semi-automatic spatial segmentation algorithms. A proper definition of shape similarity calls for the distinctions between shape similarity in images (similarity between actual geometrical shapes appearing in the images) and shape similarity between the objects depicted by the images, i.e. similarity modulo a number of geometrical transformations corresponding to changes in viewing angle, optical parameters and scale. In some cases, one wants to include even deformations of non-rigid bodies [38]. Even for the first type of similarity, it is desirable to use shape representations which are scale independent, based on curvature, angle statistics and contour complexity. Systems such as QBIC [14] use circularity, eccentricity, major axis orientation (not angle-independent) and algebraic moment.

As for color and texture, the present schemes for shape similarity modeling face serious difficulties when images include several objects or background. A

preliminary segmentation as well as modeling of spatial relationships between shapes is then necessary (are we interested in finding images where one region represents a shape similar to a given prototype or to some spatial organization of several shapes?).

4.2. Temporal Features for Representation of Shot Content

Though a set of key-frames will represent temporal content of shots to some extent, more precise representation of shot content should incorporate motion features, apart from features of static images discussed in above. In response to such a requirement, a set of statistical measures of motion features of shots has been proposed and applied in news anchor detection and shot clustering for browsing and annotation [3,39]. However, defining more quantitative measures of shots similarity that capture the nature of motion video still remains a challenging research topic.

Temporal variation and camera operations: The means and variances of average brightness and a few dominant colors calculated over all frames in a shot may be used to present quantitative temporal variations of brightness and colors. An example is that such temporal variations have been used to classify news video clips into anchorperson shots and news shots [41]. Motion information resulting from algorithms presented in Section 3 can be used to classify video sequences into static and motion sequences.

Statistic motion features: Since motion features have to roughly match human perception and it is still not clear how humans describe motions, one may have to base the motion representation on statistical motion features, rather than object trajectories [3]. More specifically, these features include directional distributions of motion vectors and average speed in different directions and areas, which may be derived from optical flow calculated between consecutive frames [3,39].

To obtain localized motion information, we can also calculate the average speed and its variance in blocks uniformly divided in frames. That is, instead of calculating average speeds in M directions for the entire frame, we calculate a set of motion statistics in M blocks of each frame. Then, the motion based comparison of shot contents will be based on the motion statistics comparison of corresponding blocks in consecutive frames.

4.3. Shot Similarity

The visual features presented above provide content representations of shots, but the goal is to define shot similarity based on these representations, to enable shot comparison or clustering for video retrieval and browsing, as discussed in Section 5.

When key-frames are used as the representation of each video shot, we can define video shot similarity based on the similarities between the two key-frame sets. If two shots are denoted as S_i and S_j , their key-frame sets as $K_i = \{f_{i,m}, m = 1, \dots, M\}$

and $K_j = \{f_{j,n}, n = 1, \dots, N\}$, then the similarity between the two shots can be defined as

$$S_k(\mathbf{S}_i, \mathbf{S}_j) = \max[s_k(f_{i,1}, f_{j,1}), s_k(f_{i,1}, f_{j,2}), \dots, s_k(f_{i,1}, f_{j,N}), \dots, s_k(f_{i,m}, f_{j,1}), s_k(f_{i,m}, f_{j,2}), \dots, s_k(f_{i,m}, f_{j,N})] \quad (4.1)$$

where s_k is a similarity metric between two images defined by any one or a combination of the image features; and there are totally $M \times N$ similarity values, from which the maximum is selected. This definition assumes that the similarity between two shots can be determined by the pair of key-frames which are most similar, and it will guarantee that if there is a pair of similar key-frames in two shots, they are considered similar.

Another definition of key-frame based shot similarity is:

$$S_k(\mathbf{S}_i, \mathbf{S}_j) = \frac{1}{M} \sum_{m=1}^M \max[s_k(f_{i,m}, f_{j,1}), s_k(f_{i,m}, f_{j,2}), \dots, s_k(f_{i,m}, f_{j,N})] \quad (4.2)$$

This definition states that the similarity between two shots is the sum of the most similar pairs of key-frames. When only one pair of frames match, this definition is equivalent to the first one. The key-frame based similarity measure as defined above can be further combined with that of motion feature based to make the comparison more meaningful for video.

4.4. Summary

This section has discussed a set of visual features for shot content representation and similarity computation for shot comparison. It should be noted that each of the features presented above represents a particular property of images or image sequences and is effective in matching those in which the particular feature is the salient one. Therefore, it is important to identify salient features for a given image or shot and to apply the appropriate similarity metric. Developing such algorithms remains a long term research topic. Also, how to integrate the different features optimally into a feature set that has the combined representation power of each feature is another challenging problem. Finally, how to link the low level features to high level semantics (presence of object and faces, and their actions) remains and will continue to remain for a long time an open problem in computer vision.

5. Video Scene Analysis and Shot Clustering

There can be hundreds of shots in one hour of a typical video program. Thus, the production of a synoptical view of the video contents usable for browsing or for quick relevance assessment calls for the recognition of meaningful time segments of longer duration than a shot, or for grouping similar shots. In media production, the level immediately higher than shots is called *sequence* or *scene*, a series of consecutive shots constituting a unit from the narrative point of view, because they are shot

in the same location, or they share some thematic visual content. The process of detecting these video scenes is analogous to paragraphing in text document analysis and requires a higher level content analysis.

5.1. Scene Analysis

Two different kinds of approaches have been proposed for the automatic recognition of sequences of programs. Aigrain *et al.* have used rules formalizing medium perception in order to detect local (in time) clues of macro-scope change [40]. These rules refer to transition effects, shot repetition, shot setting similarity, apparition of music in the soundtrack, editing rhythm and camera work. After detection of the local clues, an analysis of their temporal organization is done in order to produce the segmentation in sequences and to choose one or two representative shots for each sequence.

Zhang *et al.* have used structure models of specific types of programs such as TV news [41]. They recognize specific shot types such as shots with an anchor person, and then use the model to analyze the succession of shot types and produce a segmentation in sequences. Such model or knowledge based approaches can also be applied to, for instance, sport video parsing [42]. However, when we extend the application domain, we are facing the same difficulties as in computer vision.

In summary, video scene analysis requires higher level content analysis and one cannot expect that it can be fully automated based on visual content analysis using current image processing and computer vision techniques. Fusion of information from video, audio and close caption or transcript text analysis may be a solution and a successful example is the *Informedia* Project [29].

5.2. Shot Clustering

Clustering shots into groups, each of which contains similar content, are essential to building an index of shots for content-based retrieval and browsing. There are mainly two types of clustering approaches: partitional clustering arranges data in separate clusters; and hierarchical clustering leads to a hierarchical classification tree [44]. For the purpose of clustering a large number of shots to allow class-based video indexing and browsing with different levels of abstractions, partitional methods are more suitable since they are capable of finding optimal clustering at each level and are more suitable to obtain a good abstraction of data items. An approach of such type was proposed for video shot grouping, which is very flexible such that different feature sets, similarity metrics and iterative clustering algorithms can be applied at different levels [39].

One implementation of this approach is to use an enhanced *K*-means clustering algorithm incorporating fuzzy classification, which allows assignment of data items at the boundary of two classes to both of them according to membership function of the data item to all the classes. This is useful especially at higher levels of hierarchical browsing, where users expect all similar data items to be under a smaller number of nodes. The fuzzy clustering algorithm is as follows.

- (1) Get N classes using the K -means algorithm.
- (2) For every data item v_i , $i = 1, \dots, M$
 - calculate its similarity with each class k ($k = 1, \dots, N$) as

$$s_{ik} = \frac{d_{ik}^{-2/(\phi-1)}}{\sum_j^N d_{ij}^{-2/(\phi-1)}}$$

- where d_{ij} is the distance between data item v_i and the reference vector of class j ; ϕ is the fuzzy exponent ($\phi > 1.0$).
- if $s_{ik} \geq \rho$ where ρ is the threshold set by users ($0 < \rho < 1$), add item v_i into class k .
- if v_i is not assigned to any class in above step, assign it to the miscellaneous class.

The clustering can be based on key-frames and motion features of video shots. A comprehensive evaluation of this approach using color and motion features can be found in [3,39]. Implementation and evaluation of this clustering approach using the Self-Organization Map (SOM) method can be found in [39,45]. The advantage of SOM is its learning ability without prior knowledge and good classification performance, which have been shown by many researchers. Another benefit of using SOM is that the similarities among the extracted classes can be seen directly from the two-dimensional map. This will allow horizontal exploring as well as vertical browsing of the video data, which is very useful when we have a large number of classes at lower levels.

6. Content-Based Video Retrieval and Browsing Tools

Once a video sequence has been segmented and a scheme for representation of video content has been established, tools for content-based retrieval and browsing can be built upon that representation. This section discusses briefly a set of such tools.

6.1. Retrieval Tools

With the representation and similarity measures described in Section 4, querying a video database to retrieve shots of interest can be performed based on the meta-data, including key-frame features and motion features, or a combination of the two. The retrieval process using these features needs to be interactive and iterative, with the system accepting feedback to narrow or reformulate searches or change its link-following behavior, and to refine any queries that are given.

In case of using key-frame based features for retrieval, the index schemes and query and retrieval tools developed for image databases, such as QBIC [14], can be applied directly to video retrieval. That is, each shot is linked to its key-frames, and search of particular shots becomes a matter of identifying those key-frames from the

database which are similar to the query, according to the features and similarities defined in Section 4. Similar to QBIC image database, to accommodate different user requests, three basic query approaches should be supported: query by template manipulation, query by object feature specification, and query by visual examples. Also, the user can specify a particular feature set to be used for a retrieval [3,14,15].

The retrieved video shots in such systems will be visually represented by their key frames. The user may then view the actual shots by clicking a “Video” button, as implemented in the SWIM system [3]. That is, a video player, or other types of viewers, will be initialized and play the selected segment to allow the user verifying the retrieval results. Retrieval may thus be followed by browsing as a means to examine the broader context of the retrieved key-frame. On the other hand a query can also be initiated from browsing. That is, a user may select an key-frame while browsing and offer it as a query: a request to find all key-frames which resemble that image. Figure 10 shows an example of sketch based retrieval of key-frames and a player window launched by clicking a key-frame to allow the users to view the shot represented by the first key-frame.

Fig. 10. Example of key-frame-based retrieval: (a) A sketch based query; (b) Top five candidates from a database of key-frames.

Shot based temporal features provide another set of primitives that captures the motion nature of video and are used in our system as another retrieval tool to improve the retrieval performance. For detailed examples of shot based retrieval, see [3]. It should be pointed out that shot example based video retrieval is far less reliable than image example based image retrieval. This fact calls for more robust and effective representation schemes for video.

Though there have been many research efforts, the development of image feature based video and image retrieval algorithms is still in its early stage and will need much more progress before they can be applied to the full search of a complete video archival. Among other things, developing efficient indexing schemes based on

similarity features for managing large data volume is a critical problem for retrieval. It has been shown that traditional database indexing techniques like R-trees, etc. fail in the context of content based image search, and currently there is no technique that allows retrieval of similar objects in multi-dimensional space. Ideas from statistical clustering, multi-dimensional indexing, and dimensionality reduction may be useful in this area.

6.2. Browsing Tools

Interactive browsing of full video contents is probably the most essential feature of new forms of interactive access to digital video. Content-based video browsing tools should support two different nonlinear approaches to accessing video source data: sequential and random access. In addition, these tools should accommodate two levels of granularity, *overview* and *detail*, along with an effective bridge between the two levels. Such browsing tools can only be built by utilizing structure and content featured obtained in the video parsing, abstraction and feature extraction processes as discussed in the previous sections.

There are mainly five types of browsing tools built based on different structural and content features of video as follows,

- (1) time-line display of frames;
- (2) content-based video player;
- (3) light table of video icons;
- (4) hierarchical browser;
- (5) graph based story board.

Time-line based browsers have been favored by users in video production and editing systems, for which time-line interfaces are classical. Some browsers rest on a single shot-based image component line [4,46]; but the multidimensional dimension character of video, calling for multi-line representation of the contents has been stressed by researchers working in the frame of the Muse toolkit [47,48]. This has been systematized in the strata models proposed by Aguierre-Smith and Davenport [49]. The limitation of time-line browsers is that, since it is difficult to zoom out while keeping a good image visibility, the time-scope of what is actually displayed at a given moment on screen is relatively limited.

Another sequential browser takes an improved VCR-like interface illustrated in Fig. 11. Apart from the conventional video play functions, this browser makes use of the meta-data of video structure and content extracted from the parsing process [26]. That is, a user can select to view key-frames at a selected rate, skip between shots such as "go to the next similar shot", and play a skimmed version of the original video. In this way, the user is no longer constrained to linear and sequential viewing such as fast forward available in most conventional VCR's.

The light-table kind of video browser is often called clip window, in which a video sequence is spread in space and represented by video icons which function rather like a light table of slides [4,42]. In other words the display space is traded for time

Fig. 11. Smart video player.

to provide a rapid overview of the content of a long video. The icons are constructed with methods presented in Section 3.3. A window may contain a sequentially listed shots or scenes from a video program, a sequence of shots from a scene, or a group of similar or related shots from a stock archival. A user can get a sense about the content of a shot from the icon, especially when salient stills are used. They can also be used to display the results of retrieval operations. The clip window browser can also be constructed hierarchically just like the window file systems used in PC operating systems nowadays. That is, each icon in a clip window can be zoomed in to open another clip window, in which each icon represents a next level and finer segment of video sequences [42].

A first attempt at building hierarchical browsers — called the Video Magnifier [50] simply used successive horizontal lines each of which offered greater time detail and narrower time scope by selecting images from the video program. To improve the content accessibility of such browsers, the structural content of video obtained in video parsing are utilized [3,26]. As shown in Fig. 12, videos are accessed as a tree. At the top of the hierarchy, an entire program is represented by five key-frames, each corresponding to a sequence consisting of an equal number of consecutive shots. Any one of these segments may then be subdivided to create the next level of the hierarchy. As we descend through the hierarchy, our attention focuses on smaller groups of shots, single shots, the representative frames of a specific shot, and finally a sequence of frames represented by a key-frame. We can also move to more detailed granularity by opening a video player to view sequentially any particular segment of video selected from this browser at any level of the hierarchy.

As shown in Fig. 12, icons displayed at the top level of the hierarchy are selected based on a clustering process using shot similarities. That is, programs or clips will be first clustered into groups of similar shots or sequences as described in Section 5.2; then, each group is represented by an icon frame determined by the centroid of the group, which is then displayed at the higher levels of the hierarchical browser. As one can see from this example, with such a similarity based hierarchical browsing,

Fig. 12. Similarity-based hierarchical browser.

Fig. 13. Similarity-based hierarchical browser.

the viewer can get a sense of the content of all shots in a group roughly even without moving down to a lower level of the hierarchy [2].

An alternative approach to hierarchical browsers is the class based transition graph, proposed by Yeung *et al.* [51]. Using the clustering of visually similar shots, a directed graph whose nodes are clusters of shots is constructed, as shown in Fig. 13. Cluster A is linked to cluster B if one of the shots in A is immediately followed by a shot in B. The resulting graph is displayed for browsing, each node being represented by a key-frame extracted from one of the shots in the node. The graph can be edited for simplification by a human operator. The drawbacks of this approach lie in the difficulty of the graph layout problem, resulting in poor screen space use, and in the fact that the linear structure of the document is no longer perceptible.

With the browsing tools, another advantage of using key-frames is that we are able to browse the video content down to the key-frame level without necessarily storing the entire video. This is particularly advantageous if our storage space is limited. Such a feature is very useful not only in video databases and information systems but also to support previewing in VOD systems. What is particularly important is that the network load for transmitting small numbers of static images is far less than that required for transmitting video. Through the hierarchical browser, one may also identify a specific sequence of the video which is all that one may wish to “demand”. Thus, the browser not only reduces network load during browsing but may also reduce the need for network services when the time comes to request the actual video.

7. Video Content Analysis — A Challenging Area

In this chapter, we have discussed a number of techniques for video content analysis for computer-assisted video parsing, abstraction, retrieval and browsing. A majority of these techniques address problems in recovering structure and low level information of video sequences, though these techniques are basic and very useful in facilitating intelligent video browsing and search. Successful efforts in extracting semantic content are still very limited and there is a long way to go before we can achieve our goal of automatic video content understanding and content-based retrieval. Many of the bottlenecks and challenges in video content analysis are the classic ones in pattern recognition and computer vision. On the other hand, it is believed that low level feature-based retrieval algorithms and tools will provide us a bridge to a more intelligent solution and will improve our productivity in image and video retrieval, though the current tool set is far from satisfactory.

When we strive for visual content analysis and representation, it should be pointed out that integration of different information sources, such as speech, sound and text is as important as visual data itself in understanding and indexing visual data. Keywords and conceptual retrieval techniques are and will always be an important part of visual information systems.

An application-oriented approach is critical to the success of visual data representation and retrieval research and will prevent us from being too theoretical. By working on strongly focused applications, the research issues reviewed in this chapter can be addressed in the context of well-defined applications and will facilitate the applications; while achieving general solutions remain a long term research topic. On the other hand, it should be pointed out that, one of the most important points is that video content analysis, retrieval and management should not be thought of as a fully automatic process. We should focus on developing video analysis tools to facilitate human analysts, editors and end users to manage video more intelligently and efficiently.

It is clear that the number of research issues and their scope are rather large and expanding rapidly with advances in computing and communication [1,2]. As a result, visual data is becoming the center of multimedia computing and image analysis research, and more and more researchers from different fields are attracted and have started to explore these issues.

References

- [1] R. Jain, A. Pentland and D. Petkovic (eds.), *Workshop report: NSF-ARPA workshop on visual information management systems*, Cambridge, Mass., USA, Jun. 1995.
- [2] P. Aigrain, H. J. Zhang and D. Petkovic, Content-based representation and retrieval of visual media: A state-of-the-art review, *Int. J. Multimedia Tools and Applications* **3**, 3 (1996).
- [3] H. J. Zhang, J. H. Wu, D. Zhong and S. W. Smoliar, Video parsing, retrieval and browsing: an integrated and content-based solution, *Pattern Recogn.*, 1996.
- [4] A. Nagasaka and Y. Tanaka, Automatic video indexing and full-search for video appearances, in E. Knuth and I. M. Wegener (eds.), *Visual Database Systems* (Elsevier Science Publishers, Vol. II, Amsterdam, 1992) 113–127.
- [5] H. Ueda, T. Miyatake and S. Yoshisawa, IMPACT: An interactive natural-motion-picture dedicated multimedia authoring system, *Proc. CHI'91*, ACM, 1991, 343–350.
- [6] H. J. Zhang, A. Kankanhalli and S. W. Smoliar, Automatic partitioning of full-motion video, *Multimedia Systems*, ACM-Springer **1**, 1 (1993) 10–28.
- [7] P. Aigrain and P. Joly, The automatic real-time analysis of film editing and transition effects and its applications, *Computers & Graphics* **18**, 1 (1994) 93–103.
- [8] B. Shahraray, Scene change detection and content-based sampling of video sequences, *SPIE Proceedings Digital Video Compression: Algorithm and Technologies*, San Jose, February, **2419** (1995) 2–13.
- [9] R. Zabih, K. Mai and J. Miller, A robust method for detecting cuts and dissolves in video sequences, *Proc. ACM Multimedia'95*, San Francisco, Nov. 1995.
- [10] A. Hampapur, R. Jain and T. E. Weymouth, Production model based digital video segmentation, *Multimedia Tools and Applications* **1**, 1 (1995) 9–46.
- [11] F. Arman, A. Hsu and M. Y. Chiu, Feature management for large video databases, *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases I*, SPIE, **1908**, Feb. 1993, 2–12.
- [12] H. J. Zhang, C. Y. Low, Y. Gong and S. W. Smoliar, Video Parsing Using Compressed Data, *Proc. SPIE'94 Image and Video Processing II*, San Jose, CA, USA, Feb. 1994, 142–149.

- [13] J. Meng, Y. Juan and S.-F. Chang, Scene change detection in an MPEG compressed video sequence, *IS&T/SPIE'95 Digital Video Compression: Algorithm and Technologies*, San Jose, **2419**, Feb. 1995, 14–25.
- [14] M. Flickner *et al.*, Query by image and video content, *IEEE Computer*, Sept. 1995, 23–32.
- [15] B. Furht, S. W. Smoliar and H. J. Zhang, *Image and Video Processing in Multimedia Systems* (Kluwer Academic Publishers, 1995).
- [16] D. Bordwell and K. Thompson, *Film Art: An Introduction* (McGraw-Hill, New York, 1993).
- [17] Dynamic Vision, in *Computer Vision: Principles*, R. Kasturi and R. Jain (eds.) (IEEE Computer Society Press, Washington) 469–480.
- [18] B.-L. Yeo and B. Liu, A unified approach to temporal segmentation of motion JPEG and MPEG compressed video, *Proc. IEEE Int. Conf. Multimedia Computing and Networking*, Washington D.C., May 1995, 81–88.
- [19] I. S. Sethi and N. Patel, A statistical approach to scene change, *Proc. SPIE Conf. Storage and Retrieval for Video Databases III*, San Jose, CA, USA, Feb. 1995.
- [20] Y. T. Tse and R. L. Baker, Global zoom/pan estimation and compensation for video compression, 1991, May, *Proc. ICASSP'91*, Vol. 4.
- [21] A. Akutsu and Y. Tonomura, Video tomography: An efficient method for camerawork extraction and motion analysis, *Proc. ACM Multimedia Conference*, San Francisco, Oct. 1993.
- [22] Y. Tonomura, A. Akutsu, K. Otsuji and T. Sadakata, VideoMAP and VideoSpaceIcon: Tools for anatomizing video content, *Proc. InterChi'93*, ACM (1994) 131–136.
- [23] B. C. O'Connor, Selecting key frames of moving image documents: A digital environment for analysis and navigation, *Microcomputers for Information Management* **8**, 2 (1991) 119–133.
- [24] H. J. Zhang, S. W. Smoliar and J. H. Wu, Content-based video browsing tools, *Proc. IS&T/SPIE'95 Multimedia Computing and Networking*, San Jose, **2417**, Feb. 1994.
- [25] H. J. Zhang, C. Y. Low, S. W. Smoliar and J. H. Wu, Video parsing, retrieval and browsing: an integrated and content-based solution, *Proc. ACM Multimedia'95*, San Francisco, Nov. 1995, 15–24.
- [26] L. Teodosio and W. Bender, Salient video stills: Content and context preserved, *Proc. ACM Multimedia'93*, Anaheim, CA, USA, Aug. 1993.
- [27] M. Irani, P. Anandan, J. Bergen, R. Kumar and S. Hsu, Mosaic based representations of video sequences and their applications, *Image Communication special issue on Image and Video Semantics: Processing, Analysis and Application*, 1995.
- [28] A. Takeshita, T. Inoue and K. Tanaka, Extracting text skim structures for multimedia browsing, in Mark Maybury (ed.), *Working Notes of IJCAI Workshop on Intelligent Multimedia Information Retrieval*, Montreal, Aug. 1995, 46–58.
- [29] A. G. Hauptmann and M. Smith, Text, speech and vision for video segmentation: The informedia project, *Working Notes of IJCAI Workshop on Intelligent Multimedia Information Retrieval*, Montreal, Aug. 1995, 17–22.
- [30] M. Davis, Media streams: An iconic visual language for video annotation, *Proc. Sym. Visual Languages*, Bergen, Norway, 1993.
- [31] A. Pentland, R. W. Picard and S. Sclaroff, Photobook: Content-based manipulation of image databases, *Proc. Storage and Retrieval for Image and Video Databases II*, **2185**, San Jose, CA, USA, Feb. 1994.
- [32] M. J. Swain and D. H. Ballard, Color indexing, *Int. J. Computer Vision* **7** (1991) 11–32.

- [33] H. J. Zhang *et al.*, Image retrieval based on color features: An evaluation study, *Proc. SPIE Phonics East, Conf. Digital Storage & Archiving*, Philadelphia, Oct. 1995.
- [34] M. Stricker and M. Orengo, Similarity of color images, *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases III*, 1995, February, *SPIE Conf. Proc.* **2420**, San Jose, CA, USA, Feb. 1995, 381–392.
- [35] H. Tamura, S. Mori and T. Yamawaki, Texture features corresponding to visual perception, *IEEE Trans. Syst. Man Cybern.* **6**, 4 (1979) 460–473.
- [36] J. Mao and A. K. Jain, Texture classification and segmentation using multiresolution simultaneous autoregressive models, *Pattern Recogn.* **25**, 2 (1992) 173–188.
- [37] R. Picard and Fang Liu, A new word ordering for image similarity, *Proc. Int. Conf. Acoustic Signals and Signal Processing*, Adelaide, Australia, **5**, Mar. 1994, 129.
- [38] S. Sclaroff and A. Pentland, Modal matching for correspondence and recognition, *IEEE Trans. Pattern Analy. Mach. Intell.* **17**, 6 (1995) 544–561.
- [39] D. Zhong, H. J. Zhang and S.-F. Chang, Clustering methods for video browsing and annotation, *Proc. Storage and Retrieval for Image and Video Databases IV*, San Jose, CA, USA, Feb. 1995.
- [40] P. Aigrain and P. Joly and V'eronique Longueville, Medium-knowledge-based macro-segmentation of video into sequences, *Working Notes of IJCAI Workshop on Intelligent Multimedia Information Retrieval*, Montreal, Aug. 1995, 5–14.
- [41] H. J. Zhang, S. Y. Tan, S. W. Smoliar and Y. Gong, Automatic parsing and indexing of news video, *Multimedia Systems*, **2**, 6 (1995) 256–265.
- [42] Y. Gong, L. T. Sin, H. C. Chuan, H. J. Zhang and M. Sakauchi, Automatic parsing of TV soccer programs, *Proc. Second IEEE Int. Conf. Multimedia Computing and Systems*, Washington D.C., May 1995, 167–174.
- [43] H. J. Zhang and S. W. Smoliar, Developing power tools for video indexing and retrieval, *Proc. SPIE'94 Storage and Retrieval for Video Databases*, San Jose, CA, USA, Feb. 1994.
- [44] A. K. Jain and R. Duber, *Algorithms for Clustering Data* (Prentice Hall, 1988).
- [45] H. J. Zhang and D. Zhong, A scheme for visual feature based image indexing, *Proc. IS&T/SPIE Conf. Image and video Processing III*, San Jose, CA, 1995, 36–46.
- [46] P. Aigrain and V. Longueville, A connection graph for user navigation in a large image bank, *Proc. RIAO'91*, Barcelona, Spain, 1991, **1**, 67–84.
- [47] M. E. Hodges and R. M. Sassnett and M. S. Ackerman, A construction set for multimedia applications, *IEEE Software* (1989) 37–43.
- [48] W. E. Mackay and G. Davenport, Virtual video editing in interactive multimedia applications, *Communications of the ACM* **32**, 9, July, 1989.
- [49] T. G. Aguiere-Smith and G. Davenport, The stratification system: A design environment for random access video, *Proc. 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, USA, Nov. 1992, 250–261.
- [50] M. Mills, J. Cohen and Y. Y. Wong, A magnifier tool for video data, *Proc. INTERCHI'92*, 1992, ACM, May, 93–98.
- [51] M. M. Yeung, B.-L. Yeo, W. Wolf and B. Liu, Video browsing using clustering and scene transitions on compressed sequences, *IS&T/SPIE'95 Multimedia Computing and Networking*, San Jose, **2417**, Feb. 1995, 399–413.