Content Analysis

In recent years, multimedia documents have experienced wide distribution through the World Wide Web. In addition to the problems of receiving data over the network and storing it efficiently, users must now also cope with information overload. Search programs provide users looking for particular topics with so much information that they still must decide what is relevant. Information filtering tools, which provide the user with additional information about multimedia data besides just titles, are thus invaluable.

Most films currently available in digital form are uncommented and it is not to be expected that the large mass of existing films will ever be marked up "by hand" with a metadata track providing information about the actors or the film content. However, modern information technology is in principle capable of automatically extracting information from digital films, generating metadata that can be used to support users searching for specific content.

Pioneering works in this field include automatic cut detection in digital films [ADHC94, AHC93, ZKS93], automatic detection of newscasts [FLE95, ZGST94, ZS94], video indexing [GWJ92, RBE94, SC95, ZSW+95, ZWLS95] and the extraction of key scenes from films [Ror93, LPE97]. These works are based on various areas of research in computer science [LWT94]: in addition to compression, pattern recognition, image recognition and signal processing make vital contributions to automatic content recognition of digital films.

This chapter begins with a discussion of currently available content analysis techniques inherent to various media. In this context, new types of applications have gained in importance; several of these are presented subsequently.

9.1 Simple vs. Complex Features

Features available for content analysis are called indicators and simple (syntactic) and complex (semantic) features are distinguished.

Syntactic indicators are features that can be extracted from a digital film through direct computation without any background knowledge regarding the content of the film. As a rule, it is not possible to directly make inferences about the content of the film by means of such indicators. Syntactic indicators are thus descriptive in nature. Only after they have been transformed into semantic indicators can a film be interpreted in such a way that, for example, its genre can be determined automatically.

Fundamentally two types of indicators can be defined:

- indicators that are valid at a fixed point in time t,
- and indicators that are defined over an interval of time.

For example, video indicators such as RGB color or gray value can be extracted from an individual image, while motion vectors can only be computed from a sequence of images.

Syntactic indicators thus represent both a transformation and an aggregation of a set of digital film material. Examples of syntactic video indicators include RGB color information, gray value information, information about color differences between images, edges in an image, similarity between images, motion vector information and segmentation of individual images into unicolored regions. Examples of syntactic audio indicators include volume, speech fundamental frequency, or the frequency distribution.

Semantic indicators allow film contents to be interpreted. Most of these features are obtained through a combined evaluation of previously considered syntactic indicators. In the video area, semantic indicators include zoom, cuts, fade-outs, dissolves, wipes, and the arrangement of segmented objects into logically related groups.

One differentiates between camera effects and editing effects. Camera effects are carried out as the film is shot, by the camera operator according to stage directions; editing effects are carried out when the film is edited. Camera effects include camera motion and zooms. A zoom is a camera operation whereby the camera operator enlarges or reduces the image in the direction of the zoom center.

In a dissolve between two scenes, the earlier scene is faded out while the following scene is simultaneously faded in. During this operation, the complete individual images of the superimposed scenes can always be identified.

In a wipe, the "fade to" (new) scene is superimposed over the currently visible scene such that there is no pixel overlap between images of the new scene and those of the previous scene. In the case of a horizontal wipe from left to right, left edge regions of images of the current scene are cut off and replaced with right edge regions of images of the new scene. The size of the region that is cut off is then successively

enlarged until the new scene has completely replaced the old scene. Halfway through the time allotted for the wipe, the left half of the visible image is the right half of an image of the new scene, and the right half of the visible image is the left half of an image of the old scene.

In the audio domain, content-related features include, among many others, speech recognition; segmentation of an audio stream into discrete parts; and recognition of silence, music, and noises.

9.2 Analysis of Individual Images

Many features are available for the analysis of individual images. Such features include color, texture, objects in the image, and similarity between images or between edge images. These indicators can be used to perform classification tasks such as text recognition, which is described below. The analysis of individual images was already covered in detail in Chapter 4 (Graphics and Images).

9.2.1 Text Recognition

Examples of the need for processing information that is only available in written form on paper include the automatic recognition of addressee information in mail sorting systems, reading forms (e.g., bank transfer forms), transferring "old" data into an electronic form (e.g., converting a library's card catalog system into an online database), or handwritten text input into a PDA (Personal Digital Assistant). This section examines the automatization of these activities in greater detail.

In Optical Character Recognition (OCR) systems, text recognition takes place after an original document has been scanned in and is available as a bitmap [HHS96, SL95] (see Figure 9-1).

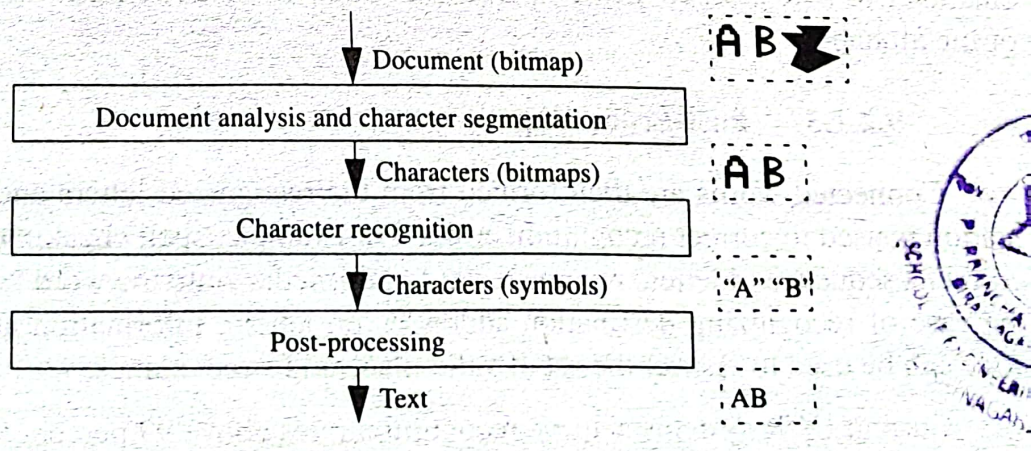

Figure 9-1 Text recognition in OCR systems.

9.2.1.1 Document Analysis and Character Segmentation

In the first step, the image is coarsely segmented into text and nontext regions. Regions that have a low probability of containing any text, for example, because they are made up of large unicolored areas, are discarded. In digital films, for example, subtitles can be recognized easily if one assumes that such text always appears in the lower fifth of the image and is oriented horizontally. Image enhancement is then performed on regions identified as containing text, for example, by removing underlining. In character segementation, the text regions are divided into a series of individual characters. In the case of printed Roman characters, connected component analysis is frequently used since once assumes that, with few exceptions, individual characters consist of connected strokes. In the case of handwriting or Japanese characters, other methods must be used. These are not covered here.

9.2.1.2 Character Recognition

In character recognition, patterns representing individual characters are classified as characters of the underlying alphabet. A set of character features is determined and forms the input to the classification process. Two classification methods can be used: template matching and structural classification.

In template matching, each individual pixel of a character is taken as a feature. The pattern is compared with a series of stored character templates (possibly multiple templates per character, depending on the number of recognized fonts). A similarity measure is computed for each stored character template. Finally, the character is classified according to the template that yielded the highest similarity.

The structural classification method works differently. Here, individual structures within a character and their arrangement are analyzed, for example, for the letter "B," vertical or horizontal strokes, curves, or holes. The resulting structural features of a character to be classified are compared with known formation rules for each character of the alphabet.

9.2.1.3 Post-Processing

Connected words are then formed from the recognized letters and context information is used to correct recognition errors. For example, spell checking can be used to correct a sequence of letters recognized as "multirnedia" into the word "multimedia." In the case of recognizing destination addresses on letters, information about the postal code can be used to restrict the set of valid place and street names.

Current OCR systems achieve recognition rates above 99 percent when analyzing printed text.

9.2.2 Similarity-Based Searches in Image Databases

In many traditional image databases, images are described through manually collected text, for example, through a brief keyword content description or the name of the photographer. To query the database, these annotations are searched for specific user-provided keywords. This procedure allows for image description and searches, but is not always satisfactory since searches are limited to image features that appeared relevant to the annotator. Moreover, many image features are complicated if not impossible to put into words, for example, textures represented in an image. Furthermore, manual annotation is very time consuming.

It thus makes sense to develop image databases that support searches based on features that can be automatically extracted from the stored images, for example, color and texture or the presence of faces. It then becomes possible to make queries such as "Find images that have coloring similar to my example image" or "Find images in which three faces are present near one another." The output consists of a set of images from which the user must make a narrower selection. Currently available techniques are not advanced enough to allow general semantic requests, although expert systems can be used if the scope is limited to a specific type of images.

If a series of features has been computed for an image, these can be combined into a feature vector that describes the image. When the database is searched, the difference between each image's feature vector and the feature vector of the query image is determined by means of a distance function. The output set includes only images from the database that have a sufficiently small distance. If multiple features enter into the distance calculation, then the distance functions of the individual features must first be suitably normalized. For this one determines, based on results of psychological perception research, an interval $[a_1, a_2]$ over which human observers can discern distance values [LEJ98]. Distance values below a_1 are insignificant to human perception. Above a_2, saturation has been reached. Figure 9-2 shows an example of a normalization function.

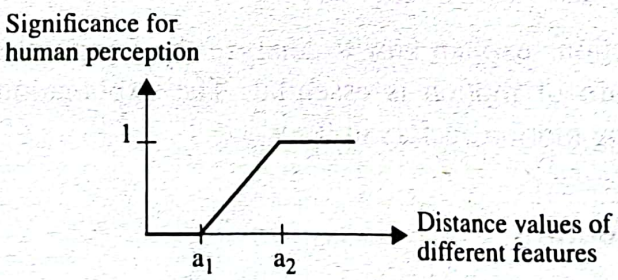

Figure 9-2 Normalization function.

For some features, such as color perception, this interval can be determined independently of a concrete query. For other features, individual determinations must be made, for example for the maximum permissible distance between human faces in the above query.

There are several ways to combine normalized feature distance functions into a total distance function. For example, one can use the so-called L_1 or L_2 metrics as a basis and introduce weights for the features. For two arbitrary feature vectors $m = (m_1, \ldots, m_n)$ and $m' = (m'_1, \ldots, m'_n)$, with weights $w_1, \ldots, w_n$ $(\Sigma w_i = 1)$ and distance $d_i = |m_i - m'_i|$ the metrics can be computed as follows:

$$d_{L_1}(m, m') = \sum_{i=1}^{n} w_i d_i$$

$$d_{L_2}(m, m') = \sqrt{\sum_{i=1}^{n} w_i d_i^2}$$

If the covariance between the measures used is known, then the Mahalanobis distance

$$d_M(m, m') = \sqrt{\left[m_1 \ \ldots \ m_n\right] C^{-1} \left[m'_1 \ \ldots \ m'_n\right]^T}$$

can be used, which is recommended for strongly correlated measures (here C is the variance-covariance matrix) [Rus94].

9.3 Analysis of Image Sequences

Analyzing image sequences requires, besides analysis of individual images (also called frames), the analysis of their temporal structure. As a rule, the smallest logical data unit (LDU) above the frame level is the shot. A shot is a sequence of frames between two cuts that was recorded continuously by one camera. Multiple related shots form a scene, for example, a dialog in which two interview partners are shown alternately.

The following sections explain how to analyze the temporal structure of a film. Besides time, the feature of motion is essential. The explanation thus begins with techniques for computing motion vectors in films.

9.3.1 Motion Vectors

Motion is an important indicator for gleaning information from digital films. With the help of this indicator, for example, newscaster scenes can be separated from correspondents' news segments in a newscast. Unlike correspondent news segments, newscaster scenes exhibit a minimal amount of motion. In the following, several approaches to computing motion vectors are presented. These can be divided into the following categories:

- Block-oriented techniques determine objects' motion. Objects can either be fixed image blocks (e.g., 8×8 pixel blocks) or "real" objects extracted through segmentation.
- Pixel-oriented techniques take the movement of every individual pixel into account.

9.3.1.1 Block-Oriented Motion Vectors

In H.261 or H.263 and in MPEG-1 or MPEG-2 (see Chapter 7 regarding compression), block-oriented motion vectors have the great advantage that, in many practical applications, they do not need to be specially computed for content analysis purposes. In these MPEG encodings, which use P and B frames, the vectors are already available. Nevertheless, these vectors do not appear well suited as a basis for exact content analysis for the following reasons:

1. It cannot be assumed that a film to be analyzed was compressed using MPEG or H.261. Other compression formats such as wavelets generally do not compute any motion vectors.
2. In MPEG and H.261, actual motion within blocks cannot be recognized using the block-oriented motion vectors.

The last point alone is a serious disadvantage. Consider an image with a square moving towards the right (see Figure 9-3). Motion vectors computed by MPEG-1 or MPEG-2 and by H.261 or H.263 are only nonzero for edge regions where the difference between both images is calculated as nonzero. This is because the color value does not change between the two images for other areas. Since only block-oriented vectors (and no object-oriented vectors) are used, it is not possible to derive actual camera or object movement. It is thus only reasonable to use this motion analysis technique as a first approximation.

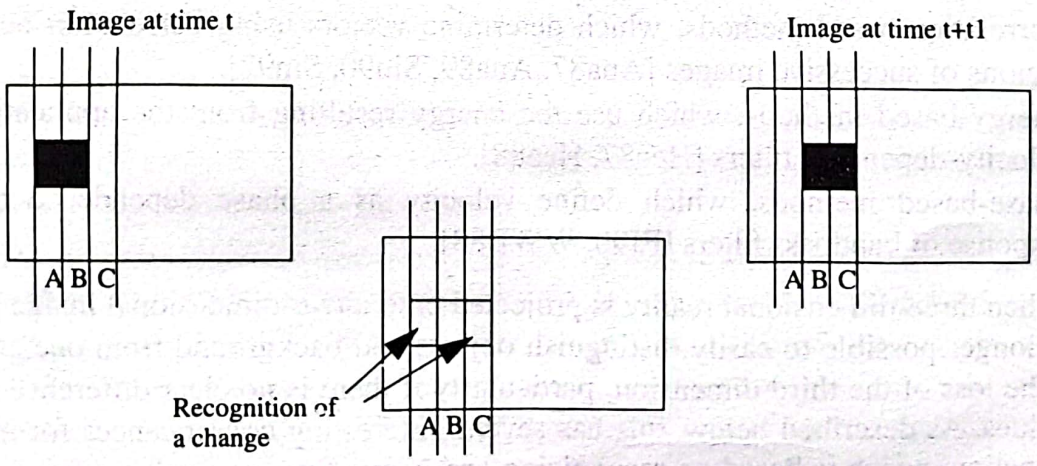

Figure 9-3 Block-based motion vectors.

9.3.1.2 Pixel-Oriented Motion Vectors

Techniques for computing motion vectors on a pixel basis are called optical flow methods.

Object movements in the real world appear in an image as color changes. In some circumstances, color changes in two successive images can indicate movements (see [Jäh97]).

As already described in Chapter 4 (Graphics and Images), tristimulus color values can be transformed into gray values. The advantage of gray values is that they can be differentiated in one-dimensional space, whereas color values span a three-dimensional space. Thus operations are generally much simpler to perform using gray values. In the following it is thus assumed that images for which motion vectors are to be computed are available as gray value images. The same technique can be used with color values, although it should be considered that in practice the computational time is significantly higher.

Optical flow refers to the movement of gray value patterns about the image area. First, the displacement vector is determined at each point for the respective gray value. Then a continuous vector field is computed, which should adequately reproduce the optical flow. Implementing both steps requires certain limiting assumptions and the results cannot be completely error-free [Jäh97].

Nevertheless, important spatio-temporal information can be obtained. It does not matter whether changes are continuous, caused by a change in vantage point, or discontinuous, due to individual objects. In any case, it should be clear that it does not make sense to consider individual images in isolation. A sequence of at least two successive images must be examined [Jäh97].

The following are classes of methods for computing optical flow [BFB94]:

- Differential methods, which use derivatives of images' gray values to determine motion vectors [Luc84, LK81, Nag83, Nag89, UGVT88].
- Correlation-based methods, which determine vectors using correlation between regions of successive images [Ana87, Ana89, Sin90, Sin92].
- Energy-based methods, which use the energy resulting from the application of velocity-dependent filters [Hee87, Hee88].
- Phase-based methods, which define velocity as a phase dependence of the response of bandpass filters [FJ90, WWB88].

When three-dimensional reality is projected onto a two-dimensional image plane, it is no longer possible to easily distinguish objects and background from one another due to the loss of the third dimension, particularly if there is no clear difference in the gray values. As described below, this has several interesting consequences for motion determination, which is based on recognizing and computing gray value differences in successive images in a sequence.

We start by assuming that gray value differences have been detected in an image pair. These are not necessarily the result of object movements. For example, the differences could have been caused by changing lighting conditions, by camera noise, as well as by a change in the camera position. In the latter case, the entire image would move, which would likely lead to difficulties at edges of the image, which are no longer identical. Here gray value changes occur without object movement. There is a physical correspondence between the images, although it is no longer understandable visually [Jäh97].

On the other hand, even if one considers objects with a uniform surface structure and coloring that rotate about an axis perpendicular to the image plane, under some circumstances gray value differences can occur that do not suffice to infer object movement. In this case, there is movement without (sufficient) gray value change [HS81].

Images always capture only a section of the real world, and the images themselves are viewed as through an aperture. If the aperture is shifted, or if an observed object moves, it is possible that there will be difficulties in correctly analyzing the image content. Gray value changes along an edge that extends across the entire image section can only be discerned in directions perpendicular to the edge. Movements which are parallel to the edge direction are present in the real world, but do not manifest themselves in gray value changes. Examples of this are shown in Figure 9-4. In each case, lines are recognized that were visible at times t_0 and t_1 in the respective images.

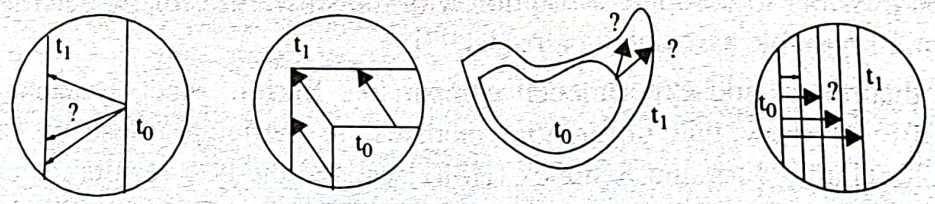

Figure 9-4 Optical flow problems.

In some cases, a nondeterminable motion component in the edge direction must be added to the computed displacement vector. This ambiguity is known as the aperture problem. Figure 9-4 illustrates this situation.

The aperture problem can obviously be solved if the image section is moved far enough that both images being examined contain either the end of the gray value edge or a corner of any angle [Jäh97]. It is then possible to unambiguously match the edges between the two images.

Additional problems occur when deformable bodies are considered. Here, an original point cannot be unambiguously mapped to its displaced point.

Even rigid bodies can lead to ambiguous computation results if they have periodic patterns or structures. The displacement vector can then only be determined to an accuracy of the order of a multiple of the width (length) of the repeated pattern or structure.

A third problem example is indistinguishable objects that move independently of one another. If these occur in large numbers, noncorresponding pairs cannot be uniquely identified. Visual correspondence (indistinguishability) provides the illusion of physical correspondence [Jäh97] (Figure 9-5).

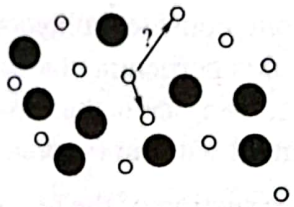

Figure 9-5 Physical correspondence.

With additional assumptions it is possible to find implementation approaches that resolve the indistinguishability problem. For example, one can assume that images are recorded at intervals short enough that movement by more than the structure or pattern width is not possible.

By assuming that indistinguishable particles move at approximately the same speed, it is possible to compute a solution with the smallest error by determining the variance of all possible associated pairs [Jäh97].

In a theoretical and experimental comparison, Barron, Fleet, Beauchemin, and Burkitt [BFB94] show that differential methods, in particular those of Lucas and Kanade [LK81] and Horn and Schunck [HS81] yielded the best results for processing digital film sequences at the time of the study (1994). Even today these methods yield outstanding results. The authors further show that of the correlation methods, the method ascribed to Singh [Sin90] yields the most reliable results.

9.3.2 Cut Detection

An important technique for dividing digital films into semantic units is the detection of editing effects, such as cuts, fade-ins and fade-outs, or dissolves and wipes [AJ94]. A cut denotes the sudden change of the image content from one image to the next. A scene consists of content-related shots, film segments demarcated by editing effects (see also [BS83, DB81, HJW95, Kuc87]).

Commercial applications of automatic digital cut detection are particularly prevalent in the area of video indexing systems [HJW94a, TATS94, ZKS93].

Well known methods for automatically recognizing editing effects are:

- methods that recognize editing effects based on histogram changes [NT91, OT93, RSK92, Ton91, ZKS93],
- methods that recognize editing effects with the aid of edge extraction [MMZ95],
- methods that recognize editing effects by means of chromatic scaling [HJW95], and
- methods that infer cuts from changes in the distribution of DCT coefficients [AHC93].

In the following, the term cut detection is used synonymously for the detection of cuts and other editing effects.

9.3.2.1 Pixel-Based Cut Detection

Pixel-based methods compute a difference between two images based on the images' pixel differences (the pixels can be either separate color values or gray values). The computation can be performed either through a pair-wise comparison of pixels or image blocks or through histogram comparison.

In the pair-wise comparison method, the pair-wise difference between two successive images is calculated. Differences occur if the color value change or gray value change of a pixel exceeds a threshold value T. In a pair-wise comparison, the number of pixels that change from one image to the next are counted. A cut is detected if a specific change percentage Z is surpassed.

The problem with this approach is that it is very sensitive to camera motion or object movement, causing these sorts of image change to be falsely recognized as cuts.

9.3.2.2 Likelihood Ratio

Here, instead of comparing individual pixels, image regions in consecutive images are compared using second order statistical methods [KJ91]. Cuts are automatically detected if an image has too many regions whose likelihood ratio exceeds a threshold value T. If m_i and m_{i+1} denote the mean intensity of a given region in two images and S_i and S_{i+1} denote the corresponding variances, then the likelihood ratio is defined as follows [KJ91]:

$$L = \frac{\left(\frac{S_i + S_{i+1}}{2} + \frac{(m_i - m_{i+1})^2}{2}\right)}{S_i \times S_{i+1}}$$

Compared to the method of pair-wise comparison, the likelihood ratio method has the advantage that small object movement or camera motion does not distort the results of the computation. Consequently, the total variance of the function is lower, which makes it easier to choose the threshold value. A potential problem is that two regions that have different distribution functions will be considered identical if they have the same mean value and the same variance. This case, however, is very rare.

9.3.2.3 Histogram Comparisons

Instead of comparing individual pixels or regions, the statistical distributions of two images' pixels can be compared. This is achieved by generating histograms that capture the distribution of gray values of an image. The idea here is that images in which objects and the background change only slightly will have similar gray value histograms. This technique can also be applied to RGB color images by using three-dimensional color histograms. However, this requires considerable computational expense, and the results are not improved substantially. Histogram-based algorithms are not sensitive to camera motion or object movement.

A potential problem is that two completely different images can have identical histograms. Figure 9-6 shows four different images that have the same histogram. The images' common histogram is shown in Figure 9-7. However, in practice this occurs so rarely between two adjacent frames in a film that this case need not be viewed as problematic.

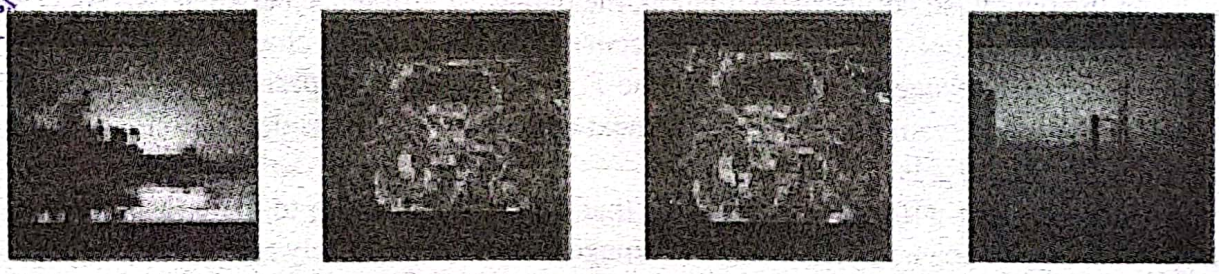

Figure 9-6 Images with identical histograms.

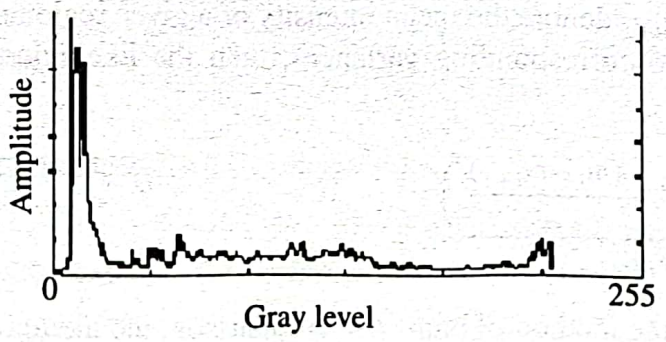

Figure 9-7 Histogram of images in Figure 9-6.

A cut is recognized if the histogram difference exceeds a specific threshold value T.

Figure 9-8 and Figure 9-9 show three consecutive images and their histograms. While the first two images have almost identical histograms, the histogram of the third image is markedly different. A cut is thus detected here.

Figure 9-8 Original images with cut after the second image.

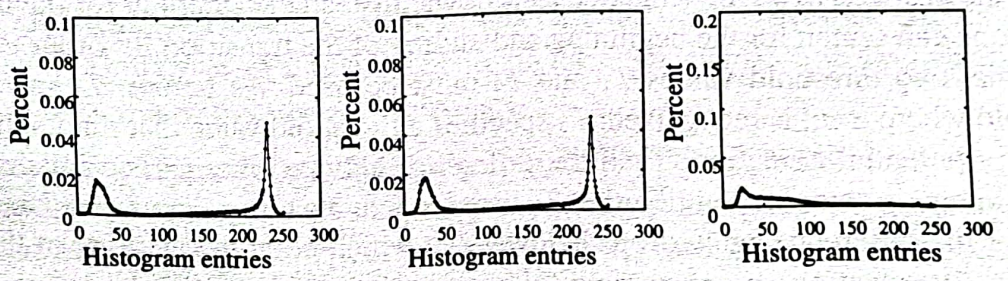

Figure 9-9 Histograms of images from Figure 9-8.

A problem with all approaches is that sudden lighting changes or deformable objects, which result in strong histogram changes, lead to false detections. Flashing lights or explosions, for example, are frequent error sources.

Nagasaka and Tanaka propose a technique to avoid these errors [NT91]. They observe that at most half the image changes, if lighting changes are excluded. The image can thus be divided into 16 rectangular regions, which are examined separately. The cut detection is then computed using the eight lowest comparison values.

9.3.2.4 Detection of Fade-Ins/Fade-Outs and Dissolves

Figure 9-10 shows the histogram differences of frames in a film scene that contains a cut and a fade-out. Note that the difference measure increases markedly for the cut, which is easily recognized by choosing an appropriate threshold value. On the other hand, the fade-ins and fade-outs show relatively minor increases in the difference measure. They cannot be recognized by simply reducing the threshold value, since many transitions would be falsely identified as cuts. In this context, transitions that are detected although none is present in the film material are called false positives.

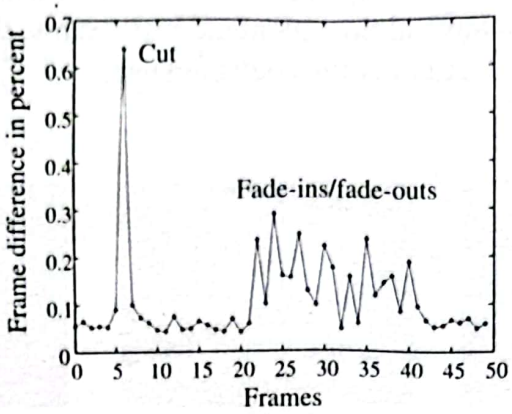

Figure 9-10 Transition detection using pixel image differences.

Observing that the first and last images of a transition are usually very different allows the problem to be reformulated as follows. Instead of searching for the transition effect, one can search for the beginning and ending of the transition. In order to accomplish this, two threshold values T_1 and T_2 must be used. As before, T_1 specifies the threshold where a cut is recognized. T_2 specifies a threshold value that marks the beginning of a potential transition. Then the difference between the marked image and the current image is computed until either T_1 is exceeded, meaning a gradual transition was found, or the difference again drops below T_2, at which point the marking of the image is ended. By selecting an appropriate tolerance value, it is possible to determine how often the difference value may fall below T_2 without the first image being unmarked [ZKS93].

A problem with this method is that the authors do not provide a way of differentiating a fade-in or fade-out from a dissolve or a wipe.

9.3.2.5 Cut Detection Based on Edge Extraction

Unlike histogram-based methods for cut detection, methods based on edge extraction make use of local information. Here the number of edge pixels that disappear or appear from one image to the next is measured (for edge extraction see Chapter 4). The individual steps of this method are as follows:

1. Calculation of the binary edge images E_i and E_{i+1} from two consecutive images I_i and I_{i+1}. In the edge images, a one represents an edge point and a zero represents any other image point.
2. Elimination of camera motion through motion compensation.
3. Calculation of the number of edge points that disappear or appear from one image to the next.
4. Decision as to whether or not a transition as occurred, whereby the measure used is the maximum of the number of edge pixels that disappeared or appeared.

Correlation-based methods are used for motion compensation between the images I_i and I_{i+1}; these include the Hausdorff distance [HKR93] and the census transform [ZW94]. Using these methods it is possible to calculate the camera motion. The Hausdorff distance is a displacement vector from which it is possible to derive the direction and length of the camera motion. If the image is simply moved by this vector, one finds that at most points, the images are not congruent, even excluding points that have newly appeared in the image due to object movement or camera motion. This process is also called warping [Wol90].

It is easy to detect transitions by looking for local maxima of the function that measures edge pixels that have newly appeared or disappeared from an image. If this exceeds a threshold value T, a transition is recognized. One must then decide whether the transition is a cut, a fade-in or fade-out, a dissolve, or a wipe. This depends on the threshold value used and on the time horizon over which the transition takes place.

A cut is easy to recognize, since in this case the time horizon is one image. Said another way, a cut causes a single deflection in the edge change function, whereas other transitions take place over greater intervals.

Fade-ins or fade-outs and dissolves can be differentiated from one another by considering the ratio of edge pixels that have come into the image to edge pixels that have disappeared. In a fade-in, there will be a strictly monotonic increase in newly appearing edge pixels; the opposite holds for a fade-out. A dissolve is comprised of a fade-in and a fade-out and can thus be recognized as their combination.

Wipes can be recognized by examining the local edge distribution and edge change. In a wipe, each image contains part of the preceding image and, at another location, part of the following image. Thus an isolated region of the image changes, while the other regions remain constant. During a horizontal wipe, there is a vertical strip that passes through the image either from right to left or left to right. In this strip, the edge change ratio is higher than in the edge regions. The problem is thus one of differentiating the strip region from other regions. Zabih [MMZ95] proposes a simple, yet robust method. The percentage of pixels that change is computed for the upper half of the image and for the left half of the image. During a horizontal wipe from left to right, most of the change will occur at the left at the beginning and at the right later. By computing change percentages for the four image quadrants at discrete times, it is possible to accurately determine the type of wipe. Moreover, in a wipe one does not find a characteristic ratio of newly appearing or disappearing edge pixels, so it is unlikely to be mistaken for a fade-in, a fade-out, or a dissolve. This is due in particular to the fact that the differences between both pixel change ratios are small since the change occurs only in a small area of the image.

9.3.2.6 Cut Detection through Chromatic Scaling

Hampapur *et al.* [Ham94, HJW94b, HJW95] describe an approach that identifies cut with the aid of the method described above, but uses a fundamentally different technique, chromatic scaling, to identify fade-ins, fade-outs, and dissolves.

The authors' cut detection method is based on cut boundaries and editing effects of partial scenes. A partial scene denotes a sequence of images that does not contain a cut. A scene can then be considered to be a set of partial scenes that belong together based on their content, for example, a dialog scene where the speakers are shot by two fixed cameras.

Hampapur *et al.* observe that editing images are never inserted between two images that form a cut boundary, though presumably they may be in the case of fade-ins or fade-outs and dissolves. They attempt to recognize these using chromatic scaling.

The basis of every video editing operation, with the exception of cuts, is the chromatic scaling of two superimposed video tracks. In a fade-in or fade-out, a black video track is used and the video track is scaled. In a dissolve, the two video tracks being mixed are both scaled [And88].

Commercial films primarily use two types of fades [Kuc87]: fade-in from black and fade-out to black. These can be modelled as chromatic scaling with positive and negative fade rates. A dissolve is the simultaneous scaling of two partial scenes and is thus the combination of a fade-in and a fade-out.

If one compares automatic transition detection methods, one finds that pixel-oriented methods recognize global features of the underlying images, whereas edge-oriented techniques tend to recognize local features. Local methods are more strongly affected by object movement than global methods, though they are more robust to brightness variations. The objective is thus to combine different approaches so as to achieve optimal transition detection. This can be accomplished as follows: since algorithms for detecting fade-ins, fade-outs, dissolves, and wipes are only known for edge-based approaches and for those based on chromatic scaling, these approaches can be used for detecting these effects. Some tests have shown that edge-based detection yields better results than chromatic scaling methods [Fis97b]. Both approaches yield similar results when recognizing fade-ins, fade-outs, and dissolves if a static image is scaled. Results with chromatic scaling are generally poorer for scenes in motion.

9.3.3 Analysis of Shots

In order to rapidly grasp the content of a shot, viewers often use the technique of "browsing." For video data, a representation is needed that greatly reduces the temporal resolution of the images. The goal is thus to find a few representative images for each shot. These so-called key frames are shown to stand in for the shot and give the viewer a rough idea of the shot's contents. When searching for specific shots in a database, this

technique substantially reduces the amount of data that must be transmitted, since only the selected shots need to be made available in full.

Ideally, expressive images would be selected by finding points where interesting objects appear or interesting events occur. However, this requires semantic understanding, which until now is hardly ever possible except for special cases such as the detection of faces [LEJ98]. Even the selection of key frames generally uses low-level features and proceeds as follows:

S denotes the shot being analyzed, consisting of images f_1, ..., f_n, and R denotes the set of images selected as representative of S. R is initialized to $R = \{f_1\}$. Further, at any time f^* denotes for the last image added to R, and m denotes its index. Thus, initially $f^* = f_1$ and $m = 1$. The distance function $d_{feature}$ measures the distance between two images; ε is the maximum tolerable difference. If exceeded, a new key frame is taken. In any case, brief periods of interference or fluctuation of up to k images are disregarded [YL95].

```
do
   l := min{l : l>m, l+k<=n, min{d_feature(f*, f_1), ..., d_feature(f*, f_l+k)}>ε}
   f* := f_l
   m := l
   R := R ∪ f*
while m<n-k
```

Examples of features that can be used to compare images include color or texture, or motion from image to image, since content change is frequently associated with camera motion or object movement [ZLSW95]. If multiple features are used, key frames should be chosen on a per feature basis. There are techniques for extracting key frames from sequences encoded in MPEG-1, MPEG-2, or motion JPEG that use features that can be computed directly from DCT coded images without requiring time-consuming decompression [DAM97].

9.3.4 Similarity-Based Search at the Shot Level

In the following it is assumed that a video is to be searched for shots that are similar to a given query shot. This problem arises, for example, if one is searching for the broadcast of a known commercial [Fis97b, RLE97]. Further we assume that feature vectors at the image level has already been computed as described in Section 9.2.

The first question that presents itself is how to represent shots to be compared. In practice, there are three approaches:

- Representation as an (unordered) set of key frames, which are described by feature vectors.
- Representation as an (ordered) sequence of key frames, which are likewise described by feature vectors.

- Representation as a disaggregated character string of feature vectors. For a shot with n images, the character string has n characters; each character is a feature vector.

Representation as a disaggregated set of images is not common.

9.3.4.1 Representation as a Set of Key Frames

This method assumes two shots S^1 and S^2 are available for which sets of representative key frames R^1 and R^2, respectively, have been chosen. The shot distance function is defined as follows, where a distance function $d_{feature}$ is used as a measure of the difference between two images [LEJ98]:

$$d_{shot}(S^1, S^2) = \sum_{k^1 \in R^1} \min\{d_{feature}(k^1, k^2) \mid k^2 \in R^2\}$$

The temporal ordering of the images is not considered here. A possible extension would be to compute mean values and deviation measures of some features as well as dominant colors across all images of the shot and to incorporate these into the comparison [ZLSW95].

9.3.4.2 Representation as a Sequence of Key Frames

If similarity also depends on whether or not the temporal ordering of the key frames in the shots being compared agrees, then the comparison algorithm presented in [DAM97] can be used. The sequence of key frames for shots S^1 and S^2 are denoted as $R^1 = \{k^1_1, ..., k^1_n\}$ and $R^2 = \{k^2_1, ..., k^2_m\}$, respectively, with $n < m$.

In the first step, k^1_1 is compared with k^2_1 and the distance is determined. Then k^1_2 is compared with the key frame(s) from R^2 that come next after k^1_2's temporal position (i.e., its offset from the beginning of the corresponding shot), whereby at most a maximum difference is permitted. For example, if k^1_2 is the third image in S^1, then it will be compared with the second, third, and fourth images in S^2, provided that these are included in R^2. The distances (up to three) are added up and the number of addends is stored. If none of the images is contained in R^2, then the comparison is skipped for k^1_2. This process is repeated for all key frames in R^1, until finally the distance sum has been added up (Step 1 in Figure 9-11). The average is then computed by dividing by the number of addends.

By processing R^1 in this manner, R^1 is displaced relative to the beginning of S^2 such that the offset of k^1_1 corresponds to the offset of k^2_2 (Step 2 in Figure 9-11). R^1 is displaced by the next offset difference in S^2 a total (m-n-1) times (the j-loop in the algorithm below). In each displacement iteration, the computed average distance is compared to the minimum of the previously computed average distances. If the new distance is smaller, it is stored as the new minimum and the current displacement is considered to be the optimal offset so far.

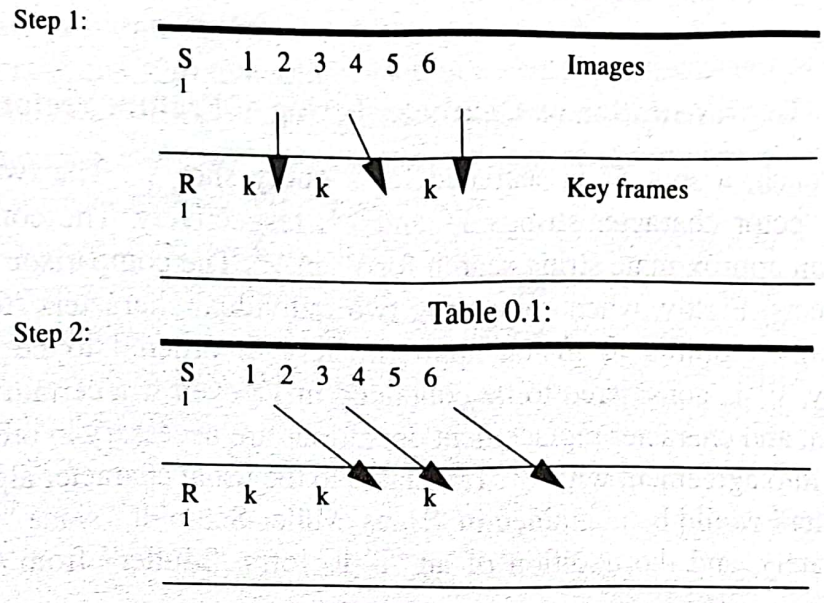

Figure 9-11 Identification of key frames.

The algorithm can be specified more precisely as follows: the function `neighbor(R2, k1i)` computes the set of offsets of the images in R^2 whose respective offsets differ by at most one position from the offset of k^1_i; the number of elements in this set is thus between zero and three:

```
min := ∞
repeat for j := 1 to m-n
    sum_j := counter_j := 0
    repeat for i := 1 to n
      A := neighbor(R2, k1_i)
      sum_j += Σ_{a∈A} d_feature(k1_i, k2_a)
      counter_j += |A|
    t := sum_j / counter_j
    if (t<min)
      min := t
      opt := j
    increase the offset of all images in R1 by
    (offset of k2_{j+1} - offset of k2_j)
```

After the algorithm completes, `min` is checked to see if it less than a similarity threshold value. If so, then shots S^1 and S^2 are considered to be similar, and `opt` represents the offset of S^1 from the beginning of S^2 where the two shots best agree.

In order for the `neighbor` function to achieve an acceptable hit rate, the key frames in R^1 and R^2 must be densely packed (ε small). However, satisfying this requirement yields a smaller reduction in the amount of data, which was the goal of the key

frames concept. This leads to the representation of a shot as a character string of feature vectors of all images.

9.3.4.3 Representation as Character String of Feature Vectors

In this approach, a shot S^2 is searched for a query shot S^1. The two shots are assigned feature vector character strings V^2 and V^1, respectively. The comparison is performed using an approximate string search for V^1 in V^2. The comparison is approximate in two respects. Firstly, when comparing two individual characters (feature vectors), they are only required to match approximately in ordered to be considered "equal." Secondly, V^1 is considered to be contained in V^2 even if a certain number of insertion, deletion, and character replacement operations are necessary in order to bring a substring of V^2 into agreement with V^1. Translated to the usual character alphabet, this means that "Muller" would be contained in "Hans_Müler-Schmidt," since "ü" and "u" match approximately, and the addition of an "l" to form "Muller" from "Müler" is allowed.

The algorithm used to perform the approximate string search must work efficiently for large alphabets (value range of the feature vectors). Appropriate algorithms are listed in [Ste94].

9.3.5 Similarity-Based Search at the Scene and Video Level

Analogous to the representation of a shot as a set or sequence of key frames or as a character string, one could represent a scene by using selected shots as well as by using all shots. Two scenes are then compared using methods analogous to those presented in Section 9.3.3 that work at a higher level of aggregation [LEJ98].

If one considers scenes in a video film, one can frequently recognize characteristic shot patterns according to which individual shots are grouped. In a dialog, there is an alternating series of shots from two classes (e.g., two interview partners shot in alternation). It is possible to directly search for such shot patterns, thereby allowing queries such as "find a dialog." In the following, shots that are similar to one another are indicated with the same letter of the alphabet, so that a dialog, for instance, can be visualized as "ABABABABA." A dialog is recognized by first searching for multiple occurrences of the same letter separated by short distances, whereby the distance between identical characters should be predominantly two. Once such a sequence is found, a test is performed to see if in the same manner, characters of a second class are arranged between the characters of the sequence. If such an embedded sequence is found, then the sequence represents a dialog. Note that the term "dialog" is used here in a broad sense—two parallel plot threads presented in alternation would also count as a dialog. If one is searching specifically for a conversation between two people, then additional search criteria must be provided.

In video production, it is not unusual to fall back on existing material that was used before either in a similar or in a completely different context. For example, one finds edited versions of original films or reports that incorporate archive material. Through a new cut, film material is used in a context that was not originally intended.

Two measures for comparing entire videos are proposed in [LEJ98]: a measure of the correspondence of two videos and a measure of the modification of the ordering of scenes within a single video (resequencing).

To compute these measures, video films are considered to be sequences of smaller units, for example, scenes or shots. Let $U^1 = \{u^1_1, ..., u^1_n\}$ denote the units that comprise video 1 and $U^2 = \{u^2_1, ..., u^2_m\}$ denote the units that comprise video 2. First the units in U^1 are compared with the units in U^2. Then an undirected graph is generated where the nodes represent the units and the edges connect units that are sufficiently similar to each other, as shown in Figure 9-12.

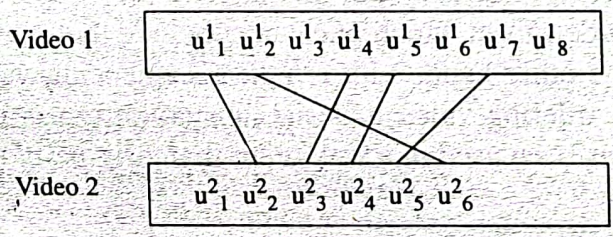

Figure 9-12 Similarity of videos.

If $|U^1 \cap U^2|$ denotes the number of units in U^1 for which there exists at least one sufficiently similar unit in U^2, then the correspondence measure is computed as follows:

$$\text{correspondence (video 1, video 2)} = \frac{|U^1 \cap U^2|}{|U_1|},$$

that is, the portion of the units in video 1 that are similar to units in video 2.

To test to what extent the common units in the videos occur in the same order, the resequencing measure is computed as follows: