Reliable Transition Detection in Videos: A Survey and Practitioner's Guide

RAINER LIENHART
MRL, Intel Corporation, 2200 Mission College Blvd.
Santa Clara, CA 95052, USA
Rainer.Lienhart@intel.com

A large number of shot boundary detection, or equivalently, transition detection techniques have been developed in recent years. They all can be classified based on a few core concepts underlying the different detection schemes. This survey emphasizes those different core concepts underlying the different detection schemes for the three most widely used video transition effects: hard cuts, fades and dissolves. Representative of each concept one or a few very sound and thoroughly tested approaches are present in detail, while others are just listed. Whenever reliable performance numbers could be found in the literature, they are mentioned. Guidelines for practitioners in video processing are also given.

Keywords: Shot boundary detection, hard cut detection, fade detection, dissolve detection, content-based indexing and retrieval, video databases

1. Introduction

In recent years the research on automatic shot boundary detection techniques has exploded, and one may wonder why solving the problem of automatic shot boundary detection is that important. There are various reasons why reliable transition detection is needed: Firstly, shots are generally considered as the elementary units constituting a video. Detecting shot boundaries thus means recovering those elementary video units, which in turn provide the ground for nearly all existing video abstraction and high-level video segmentation algorithms[15,17,26]. Secondly, during video production each transition type is chosen carefully in order to support the content and context of the video sequences. Automatically recovering all their positions and types, therefore, may help the computer to deduce high-level semantics. For instance, in feature films dissolves are often used to convey a passage of time[4]. Also dissolves occur much more often in features films, documentaries, biographical and scenic video material than in newscasts, sports, comedy and shows. The opposite is true for wipes. Therefore, automatic detection of transitions and their type can be used for automatic recognition of the video genre[9]. Shot detection is also useful to color black-and-white movies. For each shot a different gray-to-color look-up table is chosen.

This survey tries to present the few core concepts underlying research work on automatic transition detection. It emphasizes algorithms specialized in detecting specific types of transitions. Representative of each concept only one or a few very sound and thoroughly tested approaches are present in detail, while others are just listed. This survey does not try to give an exhaustive listing of all relevant work, but to help practitioners and engineers new in the field to get a thorough overview of the current state of the art in automatic shot

boundary detection.

Often shot boundary detection algorithms are strictly classified by whether the originally proposed detection algorithm was designed to operate on uncompressed or compressed videos streams. In this survey, we do not make this artificial distinction, since practically all proposed detection algorithms can be applied to the compressed as well as to the uncompressed domain. There may be slight differences in how certain features such as color histograms, edge maps, and motion information are derived, but the core concept of the classification algorithm and the kind of feature(s) that is exploited remains basically unaffected by that choice. This is especially true since compressed processing often only stands for working on so-called DC images, i.e., images which are sub-sampled by a factor of 8 after applying an 8x8 block filter[25].

Shot transitions can be classified into four classes based on the 2D image transformations applied during transition production[13]:

(i) *Identity class*: Neither of the two shots involved are modified, and no additional edit frames are added. Only hard cuts qualify for this class.

(ii) *Spatial Class*: Some spatial transformations are applied to the two shots involved. Examples are wipe, page turn, slide, and iris effects.

(iii) *Chromatic Class*: Some color space transformations are applied to the two shots involved. Examples are fade and dissolve effects.

(iv) *Spatio-Chromatic Class*: Some spatial as well as some color space transformations are applied to the two shots involved. All morphing effects fall into this category. Note that in practice often all effects in the spatial class in principle fall into the spatio-chromatic class since some chromatic transformations are always applied at the boundary between the pixels of the first and second shot such as anti-aliasing, smoothing or shading operations.

An alternative shot transition classification scheme more directed towards transition detection classifies the different transition effects based on whether the two shots involved are spatially and/or temporally well separated (see Table 1)[16]. For instance, for hard cuts and fades the two sequences involved are temporally and spatially well-separated. Their detection comes down to identifying that the video signal is abruptly governed by a new statistical process, as in the case of hard cuts, or that the video signal has been scaled by some mathematically simple and well-defined function, as in the case of fades. For wipes the two video sequences involved in the transition are spatially well-separated at any time. This is not the case for dissolves. At any time two video sequences are temporally as well as spatially intermingled, requiring dissolve detection algorithms to deal with a two source

problem (see Table 1).

Table 1: Transition classification scheme for transition detection

type of transition	The two involved sequences are	
	spatially separated	temporally separated
hard cut	yes	yes
fade	yes	yes
wipe, door, slide	yes	no
dissolve	no	no

The paper is structured as follows. A section is devoted to each of the three most common transition types. Section 2 discusses the various principles behind hard cut detection and Section 3 those behind fade detection, before Section 4 focuses on dissolves. For each transition type, a diverse set of representative approaches is presented. Whenever possible, performance numbers and pointers to comparative surveys are mentioned. Section 5 summarizes and concludes the paper.

2. Hard Cut Detection

Hard cuts are by far the most common transition effect between shots. A hard cut is defined as the direct concatenation of two shots $S_1(x, y, t)$ and $S_2(x, y, t)$. No transitional frames are involved. Thus the resulting sequence *S(x,y,t)* is formally given by

$$S(x, y, t) = (1 - u_{-1}(t - t_{hardcut})) \cdot S_1(x, y, t) + u_{-1}(t - t_{hardcut}) \cdot S_2(x, y, t) \qquad \text{(eq. 1)}$$

where $t_{hardcut}$ denotes the time stamp of the first frame after the hard cut and $u_{-1}(t)$ the unit step function (*1* for $t \geq 0$, *0* else).

A hard cut produces a temporal visual discontinuity in the video stream. Existing hard cut detection algorithms differ in the feature(s) used to measure that discontinuity and in the classification technique used to detect that discontinuity. However they almost all define hard cuts as isolated peaks in the feature's time series.

2.1. Features to measure visual discontinuity

2.1.1. Intensity/color histograms

Almost all possible variations in calculating intensity or color histogram differences between two contiguous frames have been proposed for hard cut detection such as using bin-wise differences (with or without the exploitation of a color similarity matrix), chi-square tests, or histogram intersections combined with different color spaces such as RGB, HSV, YIQ, Lab, Luv, Munsell and opponent colors. Instead of listing all those combinations, we only like to mention that in practice the simple bin-wise difference between frame-based RGB or YUV color histograms with 4x4x4 or 8x8x8 bins has proven

to be a simple, yet effective method for detecting hard cuts[11,14,20,24]. For a detailed performance characterization of the various histogram-based hard cut detection flavours the reader should consult [5,7,11,18]. Note that in general the performance improvement that can be attained by the right choice of the discontinuity classification algorithm exceeds by far the one that can be attained by fine-tuning the color space and the histogram difference function.

2.1.2. Edges/contours

Temporal visual discontinuity usually comes along with structural discontinuity, i.e., the edges of objects in the last frame before the hard cut usually cannot be found in the first frame after the hard cut, and the edges of objects in the first frame after the hard cut in turn usually cannot be found in the last frame before the hard cut. The so-called edge change ratio proposed by Zahib et al. exactly exploits this fact[28,29].

The edge change ratio (*ECR*) is defined as follows. Let σ_n be the number of edge pixels in frame *n, and* X_n^{in} and X_{n-1}^{out} the number of entering and exiting edge pixels in frames *n* and *n*-1, respectively. Then

$$ECR_n = max(X_n^{in}/\sigma_n, X_{n-1}^{out}/\sigma_{n-1}) , \qquad (1.1)$$

defines the edge change ratio ECR_n between frames *n*-1 and *n*. It ranges from 0 to 1. In [28], the edges are calculated by the Canny edge detector[6]. In order to make the measure robust against object motion, edge pixels in one image which have edge pixels nearby in the other image (e.g. within 6 pixels' distance) are not regarded as entering or exiting edge pixels. Moreover, a global motion compensation based on the Hausdorff distance was performed before the calculation of the *ECR*. Figure 1 visualizes the calculation of the *ECR*.

A comparison of ECR-based hard cut detection algorithms against histogram and motion-based algorithms can be found in [7,18,16]. For hard cut detection usually the ECR-based algorithms do not outperform the above simple color histogram-based methods, but are computationally much more expensive. The strength of the *ECR* feature is that it can also be used for fade, dissolve and wipe detection[28,29].

2.1.3. Motion

Hard cuts are also accompanied by motion discontinuity. The simplest measure of motion is the pixel-wise frame difference. At hard cut locations the pixel-wise difference is usually large[7,30,18]. Unfortunately, this simple motion measure is very susceptible to object and global camera motion. Even if global camera motion is compensated, object motion still poses a significant challenge to this feature.

More sophisticated motion features calculate the optical flow and use the number and distribution of motion vectors and the strength of the residual derived by block matching as features[1,18,22]. However, as Gargi et al. mentioned in [11], block-matching based methods do

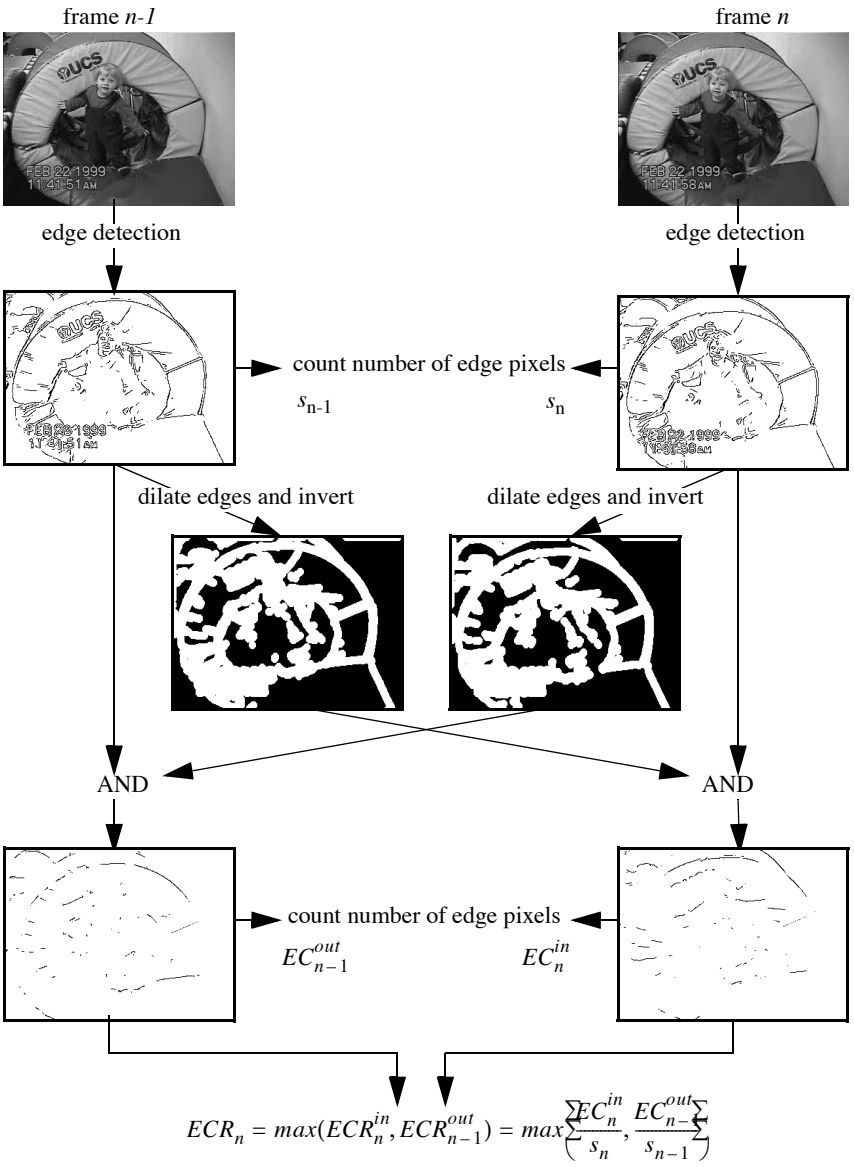

Fig. 1. Calculation Graph of the Edge Change Ratio (ECR)

not do well compared to intensity/color histogram-based algorithms. Similar results are reported in [18] as well. The core problem with all motion features arises from the fact that reliable motion estimation is far more difficult than detecting visual discontinuity, and thus less reliable. A simple detection problem is basically replaced by a much more complicated one.

2.2. Classification of discontinuity

2.2.1. Global threshold

The input to a global thesholding technique is a time series of feature values of a measure of discontinuity, which in the ideal case is supposed to show a single large peak at hard cut locations. A hard cut is declared each time the feature value *f(t)* surpasses a globally fixed threshold.

A common problem of global thresholding is that in practice it is impossible to find a single global threshold that works with all kinds of video material[14]. Therefore, global thresholds should be avoided.

2.2.2. Adaptive threshold

The input to an adaptive thesholding technique is a time series of feature values of a measure of discontinuity, which in the ideal case is supposed to show a single large peak at hard cut locations. A hard cut is detected based on the difference of the current feature values *f(t)* from its local neighborhood. Usually a temporal sliding window of size $2w+1$ centered around the current time instance t is chosen to represent the local neighborhood. A hard cut is declared if all the following conditions are satisfied:

(i) *f(t)* takes on the maximum value inside of the window, i.e., $f(t) \geq f(x) \quad \forall x \in [t-w, t+w]$ [25, 24].

(ii) The ratio between *f(t)* and the second largest value $f(t_2)$ ($f(t_2) \geq f(x) \quad \forall x \in [t-w, t+w] \backslash \{t\}$) surpasses a second threshold th_2 [25]. Alternatively Truong et al. propose to use the following criterion:

$$ratio = \frac{(f(t)+c)}{c + \sum_{x \in [t-w, t+w] \backslash \{t\}} f(x)'} . \qquad \text{(eq. 2)}$$

If the *ratio* surpasses a given threshold, a hard cut is declared. The constant c is added to the ratio calculation in order to deal with freeze frames, in which case the measure of discontinuity would be almost zero, thus making the determination of a good adaptive threshold difficult. Note that this constant c could also be added to the first criterion by requiring that the ratio between the largest value *f(t)* and the maximum of the second largest value $f(t_2)$ and c surpasses th_2.

Both adaptive thresholding techniques in combination with color histogram differences between frames have been proven to lead to high performance[25, 24, 11].

3. Fade Detection

A fade sequence *S(x,y,t)* of duration *T* is defined as scaling the pixel intensities/colors of a video sequence $S_1(x, y, t)$ by a temporally monotone scaling function *f(t)*:

$$S(x,y,t) = f(t) \cdot S_1(x,y,t)\ ,\ t \in [0,T] \qquad \text{(eq. 3)}$$

For a fade-in it is additionally required that *f(0)=0* and *f(T)=1*, while a fade-out requires *f(T)=0* and *f(0)=1*. Often *f(t)* is linear, i.e. $f(t) = t/T$ for a fade-in and $f(t) = (T-t)/T$ for a fade-out. Also often a fade-out is directly followed by a fade-in. Such a sequence is called a fade group and often considered as one transition.

3.1. Standard deviation of pixel intensities

During fades the pixels' intensity/color scaling is clearly visible in the time series of its standard deviation as depicted in Figure 2[14]. Assuming that S_1 is (roughly) an ergodic random process, the variance of S_1 becomes independent of t and it follows[14]:

$$\begin{aligned} Var(S(x,y,t)) &= Var(f(t) \cdot S_1(x,y,t)) \\ &= f^2(t) \cdot Var(S_1(x,y,t)) \\ &= f^2(t) \cdot Var(S_1(x,y)) \end{aligned} \qquad \text{(eq. 4)}$$

.

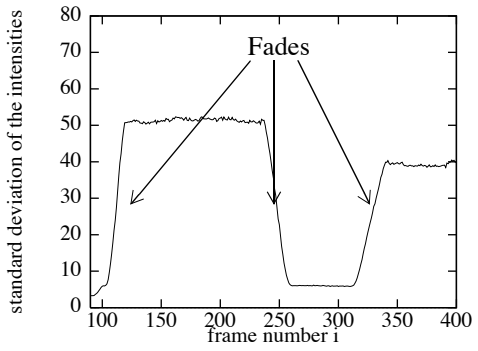

Fig. 2. Characteristic pattern of the standard deviation of pixel intensities during a fade-in and fade-out[14].

For the pixels' intensity/color standard deviation we get

$$\sigma(S(x,y,t)) = f(t) \cdot \sigma(S_1(x,y))\ . \qquad \text{(eq. 5)}$$

Thus the scaling function *f(t)* is directly revealed in the standard deviation of the pixels' intensity/color distribution.

Lienhart[14] proposes to first locate all monochrome frames in the video as potential start/end points of fades. Monochrome frames are identified as frames with $\sigma(S(x,y,t))$ close to zero. Fades are then detected by starting to search in both directions for a linear increase in the pixels' intensity/color standard deviation. Linear increase is detected by means of evaluating the error introduced by approximating the curve through a straight line (linear regression). Only if the absolute value of the correlation and the slope of the calculated best fitting line surpasses certain thresholds, and only if the detected duration falls into the typical range, is a fade declared. On a large and very difficult video test set, an average hit rate of 87% has been reported at a small false alarm rate of 30%[14].

An alternative approach also based on the variance of pixel intensities is proposed by Alattar[3]. Fades are detected first by recording all negative spikes in the second order difference of the pixel intensity variance time series, and then ensuring that the first order difference of the mean curve is relatively constant next to the negative spike.

A combination of both approaches is presented by Truong et al.[24]. They report a very high

recall rate of 93.3% at a precision rate of 82.4% on their large video test set containing 111 fades. Their algorithm works as follows:

(i) Determine all monochrome frames

(ii) Keep all monochrome frames which are next to a large negative spike in the second order difference curve of the intensity variance. Note that such a spike can be caused by motion also.

(iii) Check whether the smoothed first order difference curve of the mean intensity remains relatively constant and does not change its sign during a fade sequence. It is also required that the absolute value of the slope of the smoothed first order difference exceeds some large threshold, and that the intensity variance of the first and last frame of a fade-out and fade-in, respectively, exceeds some threshold.

3.2. Edges/contours

During a fade-out object contours (i.e., object edges) gradually disappear, while during a fade-in they gradually show up. One measure of the change of contours is the edge change ratio (*ECR*) as proposed by Zabih et al.[28] (see Section 2.1.2 for a definition of *ECR*). During a fade-in/out the number of entering/exiting edges predominates[28,29] the exiting/entering edges. In other words, during a fade-in $ECR^{in} \gg ECR^{out}$, and the reverse $ECR^{in} \ll ECR^{out}$ is true during a fade-out. In addition, and unlike hard cuts which only lead to a single peak in the *ECR* time series, fades and other gradual transitions lead to an interval where the *ECR* is elevated. Zabih et al. exploited these features directly in their shot detection approach[28,29]. Two variations of this approach can be found in [18] and [27].

In practice, the contour-based approach again did not perform as well as the approaches based on the intensity standard deviation[14,18].

4. Dissolve Detection

A dissolve sequence *D(x,y,t)* of duration *T* is defined as the mixture of two video sequences $S_1(x, y, t)$ and $S_2(x, y, t)$, where the first sequence is fading out while the second is fading in:

$$D(x, y, t) = f_1(t) \cdot S_1(x, y, t) + f_2(t) \cdot S_2(x, y, t), \; t \in [0, T] \qquad \text{(eq. 6)}$$

Therefore, sometimes the term cross-fade is also used. The most common dissolve types are cross-dissolves with

$$\begin{aligned} f_1(t) &= \frac{T-t}{T} = 1 - f_2(t) \\ f_2(t) &= \frac{t}{T} \end{aligned} \quad , t \in [0, T] \qquad \text{(eq. 7)}$$

They are much more common than the second most common dissolve type, which are additive dissolves with

$$f_1(t) = \begin{cases} 1 & if(t \le c_1) \\ \frac{T-t}{T-c_1} & else \end{cases}, \; t \in [0,T], \; c_1 = \;]0,T[\qquad \text{(eq. 8)}$$

$$f_2(t) = \begin{cases} \frac{t}{c_2} & if(t \le c_2) \\ 1 & else \end{cases}, \; t \in [0,T], \; c_2 = \;]0,T[\qquad \text{(eq. 9)}$$

Their intensity scaling functions are depicted in Figure 3. Since cross-dissolves are so much more common than additive dissolves, we will restrict the subsequent discussion to cross dissolves. This is also in line with nearly all dissolve detection work. Only [16] explicitly tries to take additive dissolves into account as well.

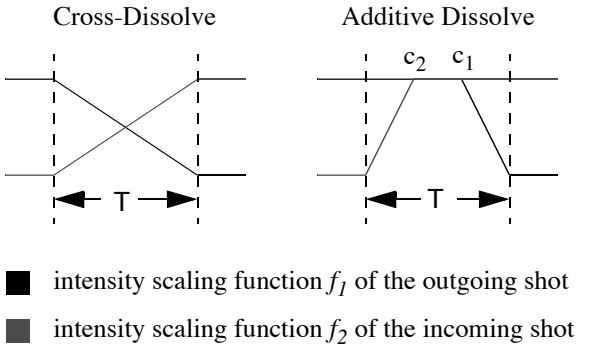

Fig. 3. Intensity scaling functions of two popular dissolve types[14,16]

Types of dissolves. Basically three different kinds of dissolves can be distinguished based on the visual difference between the two shots involved:

(i) The two shots involved have different color distributions. Thus, they are different enough so that a hard cut would be detected between them if the dissolve sequence were removed.

(ii) The two shots involved have similar color distributions which a color histogram-based hard cut detection algorithm would not detect, however, the structure between the images is different enough to be detectable by an edge-based algorithm. An example is a transition from one cloud scene to another.

(iii) The two shots involved have similar color distributions and similar spatial layout. This type of dissolve represents a special type of morphing. One example of such a dissolve can be found in the movie Aliens where Ripley's sleeping face matches graphically the curve of the earth[4].

Usually dissolve detection approaches concentrate only on the first two types of dissolves,

since they clearly mark transitions between semantically different shots. The morphing-like dissolves are ignored since they only represent a transition from a technical point of view and not from a semantic point of view.

4.1. Temporal change of the pixel intensities

Assuming a cross-dissolve without motion in S_1 and S_2 during the dissolve (i.e., the frames of the two shots involved are basically frozen), the following holds true

$$\frac{\partial D(x,y,t)}{\partial t} = \frac{S_2(x,y) - S_1(x,y)}{T}\ ,\ t \in [0,T] \qquad \text{(eq. 10)}$$

Thus, a dissolve with little motion can be detected by searching for temporal ranges in a video where most of the pixels in the video change roughly linearly over time[13]. Unfortunately, this feature is very noise and motion sensitive, so that dissolve detectors based on this feature tend to miss some dissolves at a very large false alarm rate.

Gu et al.[12] present several techniques to make this approach more motion and noise tolerant. They propose to substantially smooth the images in order to reduce the impact of noise and motion. In fact, they use the DC values of the luminance blocks in the MPEG videos. Then the percentage of blocks is calculated those temporal absolute DC differences fall into ranges, which are typical for dissolves[12]. A dissolve is declared if the percentage consistently exceeds a threshold over a duration which is typical for dissolves (10-60 frames at 29.95 fps).

It it obvious that the cross-dissolve detection approach can be extended to additive dissolves. For an additive dissolve we have:

$$\frac{\partial D_{add}(x,y,t)}{\partial t} = \begin{cases} S_2(x,y)/c_2 & if(t \le c_2) \\ 0 & else \\ -S_1(x,y)/(T-c_1) & if(t \ge c_1) \end{cases},\ t \in [0,T]\ ,\ c_2 \le c_1 \qquad \text{(eq. 11)}$$

and

$$\frac{\partial D_{add}(x,y,t)}{\partial t} = \begin{cases} S_2(x,y)/c_2 & if(t \le c_1) \\ S_2(x,y)/c_2 - S_1(x,y)/(T-c_1) & else \\ -S_1(x,y)/(T-c_1) & if(t \ge c_2) \end{cases},\ t \in [0,T]\ ,\ c_2 > c_1\ , \qquad \text{(eq. 12)}$$

respectively.

Another very interesting approach was proposed by Nam et al.[21]. They correctly observed that the features derived from actual dissolve transitions behave in a far more complicated manner due to motion and/or post-processing operations than what the analysis under the simplified assumptions made above actually suggests. However, independent of what edit function was used to generate the dissolve effect and how it was degraded, the artificial character of the temporal development of the pixels should still be visible in the feature's

time series. Nam et al. propose to approximate the temporal development of each DC pixel in the video by calculating the fitness of a B-spline approximation over a window of L=31 frames. During dissolves the standard deviation in the temporal development of the DC pixels should be large; however, the error introduced by the B-spline approximation should be small due to the artificial character of the transition. In contrast, for motion scenes and other post-processing noise, the approximation error should be large. In detail, the inter-frame variance is defined as

$$Var(x,y,t) = \frac{1}{L}\sum_{i=t-\frac{L}{2}}^{t+\frac{L}{2}} [D(x,y,i) - E(x,y,t)]^2 \text{ where } E(x,y,t) = \frac{1}{L+1}\sum_{i=t-\frac{L}{2}}^{t+\frac{L}{2}} D(x,y,t) \quad \text{(eq. 13)}$$

Then, the error introduced by the B-spline approximation becomes

$$e(x,y,t) = \frac{1}{L+1}\sum_{i=t-\frac{L}{2}}^{t+\frac{L}{2}} [D(x,y,t) - D'(x,y,t)]^2 \quad \text{(eq. 14)}$$

if $D'(x,y,t)$ denotes the B-spline approximation. During dissolves the fitting error should be small due to the linear-like development, while the inter-frame variance is high. For static scenes the variance as well as the fitting error are small, and sequences with camera and/or object motion will have a high variance as well as a high fitting error. A formula to set the thresholds adaptively based on the fitting error and variance is provided in [21].

4.2. Temporal change of the frame-based intensity variance

Given the above dissolve definition and assuming S_1 and S_2 to be statistically independent with intensity variances $\sigma_1(t) = Var(S_1(x,y,t))$ and $\sigma_2(t) = Var(S_2(x,y,t))$, the intensity variance $\sigma(t) = Var(D(x,y,t))$ is given by

$$\begin{aligned} Var(D(x,y,t)) = \sigma(t) &= Var(f_1(t)\cdot S_1(x,y,t) + f_2(t)\cdot S_2(x,y,t)) \\ &= f_1^2(t)\cdot Var(S_1(x,y,t)) + f_2^2(t)\cdot Var(S_2(x,y,t)) \\ &= f_1^2(t)\cdot \sigma_1(t) + f_2^2(t)\cdot \sigma_2(t) \end{aligned} \quad . \quad \text{(eq. 15)}$$

Assuming further that S_1 and S_2 are (roughly) ergodic random processes[2], the variance of S_1 and S_2 become independent of t:

$$\sigma(t) = f_1^2(t)\cdot \sigma_1 + f_2^2(t)\cdot \sigma_2 \, . \quad \text{(eq. 16)}$$

For cross-dissolves it follows:

$$
\begin{aligned}
\sigma(t) &= \frac{(T-t)^2}{T^2}\cdot\sigma_1 + \frac{t^2}{T^2}\cdot\sigma_2 \\
&= c(t-a)^2 - b \\
\frac{\partial\sigma(t)}{\partial t} &= 2c(t-a) \\
\frac{\partial^2\sigma(t)}{\partial^2 t} &= 2c
\end{aligned}
\qquad \text{with} \qquad
\begin{aligned}
a &= \frac{T\sigma_1}{\sigma_1+\sigma_2} \\
b &= \sigma_1 - \frac{\sigma^2{}_1}{\sigma_1+\sigma_2} = \frac{\sigma_1\sigma_2}{\sigma_1+\sigma_2} \\
c &= \frac{\sigma_1+\sigma_2}{T^2}
\end{aligned}
\qquad \text{(eq. 17)}
$$

From this it can be seen directly that the frame-based intensity variance curve of an ideal dissolve has a parabolic shape. Thus, the first order derivative before and after a dissolve is zero and a positive constant during a dissolve.

This insight was first published by Alattar[2] and many other researchers have built on it [8,12,19,23,24]. Alattar proposed to exploit this feature by approximating the second derivative by means of the second-order difference. In this approximation, two large negative spikes are introduced at the boundaries of the dissolves (discontinuities in the continuous case). He used these large negative spikes to detect dissolve candidates as two large negative spikes which are less apart than a given constant. In addition, Alattar required the average value of the second order difference to be above a given positive constant within a dissolve.

In practice, however, the two large negative spikes at the beginning and end of a dissolve are often not that pronounced due to noise and motion in the video. Therefore, Truong et al. propose to exploit the facts that[23,24]

(i) the first derivative during a cross-dissolve should be monotonically increasing from a negative value up to a positive value,

(ii) the intensity variances of the two shots involved should be larger than a minimal threshold T_{minVar} (One of the variances σ_1 and σ_2 can be close to zero only in the case of a fade), and

(iii) the actual dissolve duration usually falls between two well defined thresholds T_{Dmin} / T_{Dmax}.

Given the above thresholds it can be directly derived that the first derivative will change linearly from

$$
-\frac{2\sigma_1}{T^2} \le -\frac{2T_{minVar}}{T_{Dmax}} \text{ to } \frac{2\sigma_2}{T^2} \ge \frac{2T_{minVar}}{T_{Dmax}}\,. \qquad \text{(eq. 18)}
$$

Truong et al. propose to trigger the detection of dissolve candidates by all zero crossing sequences in the smoothed first order difference which comply with the above upper and lower threshold bounds and which are almost monotonically increasing. In addition, relaxations of the following conditions are checked too, and can be derived from an ideal cross-dissolve:

$$\frac{\sigma_1}{(\sigma_1+\sigma_2)} = \frac{\partial\sigma(0)}{\partial t} - \frac{\partial\sigma(t_0)}{\partial t}\,, \qquad \text{(eq. 19)}$$

$$\frac{\sigma_2}{(\sigma_1+\sigma_2)} = \frac{\partial\sigma(T)}{\partial t} - \frac{\partial\sigma(t_0)}{\partial t}\,, \text{ and} \qquad \text{(eq. 20)}$$

$$\frac{\partial\sigma(0)}{\partial t} + \frac{\partial\sigma(T)}{\partial t} - 2\frac{\partial\sigma(t_0)}{\partial t} = \frac{\sigma_1^2+\sigma_2^2}{\sigma_1+\sigma_2} > \frac{\sigma_1+\sigma_2}{2} \qquad \text{(eq. 21)}$$

The approach was tested thoroughly on two hours of video, and a recall of 82.2% at a precision of 75.1% has been reported[24].

4.3. Edges/contours

During dissolves object contours gradually disappear and new object contours gradually show up. As a consequence, the perceived contrast decreases toward the center of a dissolve.

One measure of contour changes is the edge change ratio (*ECR*), as already mentioned[28]. During the first half of a dissolve the number of exiting edge pixels is large, while the number of entering edge pixels is large during the second half. Zabih et al. exploit this feature in their shot detection approach[28,29].

Although good hit rates have been reported in practice for the *ECR*-based approaches, the false alarm rate was usually very large (>100%) due to the feature's sensitivity to object motion. Also, the method does not seem to be very suitable for finding the actual boundaries of dissolves since the values of ECR^{in} and ECR^{out} return to normal values much earlier than the boundaries are reached.

Lienhart[14] tried to avoid the motion sensitivity by analyzing the change in the average edge strength over time, which can be calculated on a per frame basis. The so-called *edge-based contrast* (*EC*) captures and amplifies the relation between stronger and weaker edges. Given the edge map $K(x,y,t)$ of frame $S(x,y,t)$ as well as a lower threshold value θ_w for weak and a higher threshold value θ_s for strong edges, the following formula defines the edge-based contrast:

$$EC(K) = 1 + \frac{s(K)-w(K)-1}{s(K)+w(K)+1} \;\;,\; EC(K)\in[0,2] \qquad (1.2)$$

where

$$w(K) = \sum_{x,y} W_K(x,y) \;\;,\;\; s(K) = \sum_{x,y} S_K(x,y) \qquad \text{(eq. 22)}$$

and

$$W_K(x,y) = \begin{cases} \sum K(x,y) & if\ \theta_w \le K(x,y) < \theta_s \\ \sum 0 & else \end{cases} \;,\; S_K(x,y) = \begin{cases} \sum K(x,y) & if\ \theta_s \le K(x,y) \\ \sum 0 & else \end{cases} \;. \qquad (1.3)$$

Figure 4 depicts some examples of how dissolves temporally influence the *EC*. It can easily be recognized that a dissolve coincides with distinct local minima, surrounded by steep flanks. The boundaries of a dissolve occur in company with the abrupt end of the steep flanks. Dissolve locations and their extent are thus detected by searching first for those

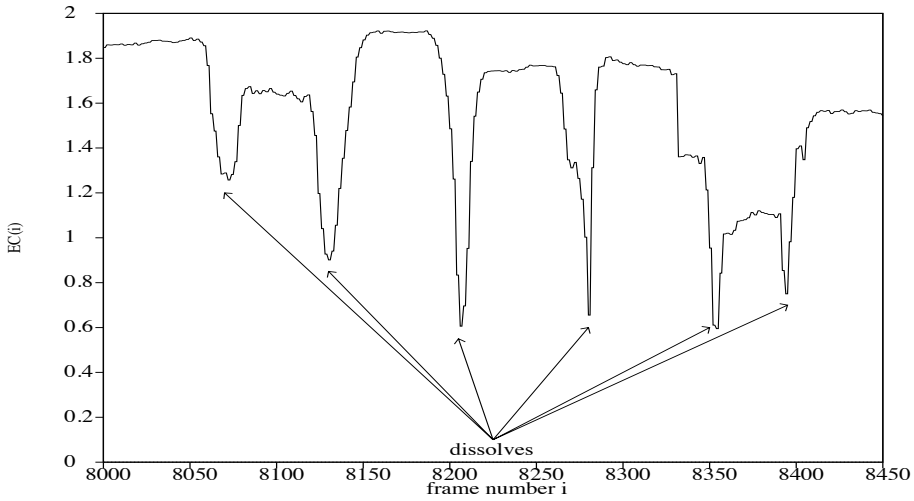

Fig. 4. Some examples of how dissolves temporally influence the *EC*[16]

distinct local minima, and then extending the detection to both sides until the error produced by linear approximation of each flank surpasses a threshold. If the resulting detection duration falls inside the target dissolve duration range, a dissolve is recognized.

On a difficult video test set, this dissolve detection approach outperformed the *ECR*-based approach at a much lower false alarm rate. However, the reported average false alarm rate of 59% is still too high for most practical cases[14].

4.4. Multi-resolution pattern recognition

A novel, and conceptually somewhat different, approach from the ones presented so far has been taken in [16]. It is less concerned about the actual feature used for dissolve detection (all of the above mentioned can be used), but more with a general sound framework to recognize transition and special effects in general. The proposed overall system consists of two large components:

(1) A transition detector training system including a transition synthesizer and

(2) A fully multi-resolution transition detection system.

In principle, the transition synthesizer can create an infinite number of dissolve examples from a given video database. It is used in their approach to create a large training and validation set of dissolves with a fixed length and a fixed position of the dissolve center.

'Large' means tens of thousands of examples. These sets are then used to train iteratively with the so-called bootstrap method a heuristically optimal classifier. Trained artificial neural networks as well as support vector machines serve as classifiers, and the sum of the magnitude of the directional gradients $CS_{avg}(t)$ is used as a simple contrast strength measure:

$$CS_{avg}(t) = \frac{\sum_{x \in X} \sum_{y \in Y} \left| \frac{\partial}{\partial x} I(x, y, t) \right| + \left| \frac{\partial}{\partial y} I(x, y, t) \right|}{|X||Y|} . \qquad \text{(eq. 23)}$$

Dissolves of various durations are then detected by means of a multi-resolution search. In a first step, the frame-based CS_{avg} is derived (Figure 5(a)). They form a time series of feature values, which is re-scaled to a full set of time series at different sampling rates, creating a time series pyramid (Figure 5(b)). At each scale, a fixed-size sliding window runs over the time series, serving as the input to a fixed-scale fixed-position dissolve detector (Figure 5(c)). The fixed-scale fixed position dissolve detector outputs the probability that the feature sequence in the window was produced by a dissolve. This results in a set of time series of dissolve probabilities at the various scales (Figure 5(d)). For scale integration, all probability times series are rescaled to the original time scale (Figure 5(e)), and then integrated into a final answer about the probability of a dissolve transition at a certain location and temporal extent (Figure 5(f)).

Lienhart[16] reports a detection rate of 75% at an acceptable false alarm rate of 16% on a test video set for which, so far, the best reported detection and false alarm rate had been 66% and 59%, respectively, using the *edge change ratio*[29] or the edge contrast[14].

5. Summary and Conclusion

Automatic shot boundary detection algorithms, or equivalently, transition detection algorithms have reached a high level of maturity. In this survey we focused on the three most widely used video transition effects: hard cuts, fades and dissolves. The different core concepts underlying the different detection schemes were presented together with guidelines for practitioners in video processing. Future research will focus on the less frequently used transition effects such as wipes, doors, and iris effects and the recovering of their underlying transformation parameters.

6. References

1. A. Akutsu, Y. Tonomura, H. Hashimoto, and Y. Ohba. Video Indexing Using Motion Vectors. Proc. *SPIE Visual Communications and Image Processing*, Vol. 1818, pp. 1522-1530, 1992.
2. A. M. Alattar. Detecting and Compressing Dissolve Regions in Video Sequences with a DVI Multimedia Image Compression Algorithm. *IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 1, pp. 13 -16, May 1993.
3. A. M. Alattar. Detecting Fade Regions in Uncompressed Video Sequences. *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 3025-3028, 1997.
4. D. Bordwell, K. Thompson. *Film Art: An Introduction*. McGraw-Hill, Inc., 4th ed., 1993.

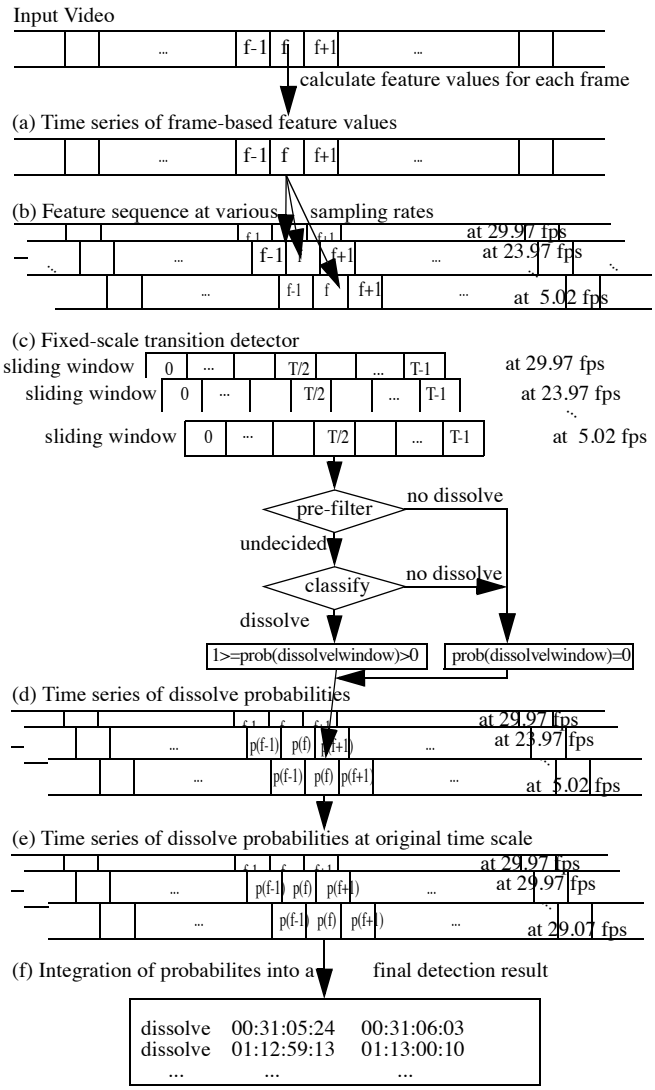

Fig. 5. System overview of the transition detection system

5. J. S. Boreczky and L. A. Rowe. Comparison of Video Shot Boundary Detection Techniques. *SPIE Storage and Retrieval for Still Image and Video Databases IV*, Vol. 2664, pp. 170-179, Jan. 1996.
6. J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 34-43, Nov. 1986.
7. A. Dailianas, R. B. Allen, and P. England: Comparison of Automatic Video Segmentation Algorithms. *SPIE Integration Issues in Large Commercial Media Delivery Systems*, Vol. 2615, pp. 2-16, Oct. 1995.

8. W.A.C. Fernando, C.N. Canagarajah, D.R. Bull. Fade and Dissolve Detection in Uncompressed and Compressed Video Sequences. *IEEE International Conference on Image Processing (ICIP)*, Vol. 3 , pp. 299-303, 1999.
9. S. Fischer, R. Lienhart, and W.Effelsberg. Automatic Recognition of Film Genres. *Proc. ACM Multimedia 95*, San Francisco, CA, Nov. 1995, pp. 295-304.
10. U. Gargi, R. Kasturi, and S. Antani. Performance Characterization and Comparison of Video Indexing Algorithms. *IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, pp. 559-565, June 1998.
11. U. Gargi, R. Kasturi, S. H. Strayer. Performance Characterization of Video-Shot-Change Detection Methods. *IEEE Transaction on Circuits and Systems for Video Technology*, Vol. 10, No. 1, Feb. 2000.
12. L. Gu, K. Tsui, D. Keightley. Dissolve Detection in MPEG Compressed Video. *IEEE International Conference on Intelligent Processing Systems (ICIPS '97)*, Vol.2, pp. 1692-1696, 1997.
13. A. Hampapur, R. Jain, and T. E. Weymouth. Production Model Based Digital Video Segmentation. *Multimedia Tools and Applications*, Vol.1, pp. 9-45, 1995.
14. R. Lienhart. Comparison of Automatic Shot Boundary Detection Algorithms. *SPIE Storage and Retrieval for Still Image and Video Databases VII 1999*, Vol. 3656, pp. 290-301, Jan. 1999.
15. R. Lienhart. Dynamic Video Summarization of Home Video. *SPIE Storage and Retrieval for Media Databases 2000*, Vol. 3972, pp. 378-389, Jan. 2000.
16. R. Lienhart. Reliable Dissolve Detection. *SPIE Storage and Retrieval for Media Databases 2001*, Vol. 4315, pp. 219-230, Jan. 2001.
17. R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video Abstracting. *Communications of the ACM*, Vol. 40, No. 12, pp. 55-62, Dec. 1997.
18. G. Lupatini, C. Saraceno, and R. Leonardi. Scene Break Detection: A Comparison. *Research Issues in Data Engineering, Workshop on Continuos Media Databases and Applications*, pp. 34-41.1998.
19. J. Meng, Y. Huan, and S.F. Chang. Scene Change Detection in a MPEG-Compressed Video Sequence. IS&T/SPIE Proceedings, Vol. 2419, San Jose, CA, Feb. 1995.
20. A. Nagasaka and Y. Tanaka. Automatic Video Indexing and Full-video Search for Object Appearances. *Visual Database Systems II*, pp. 113-127, 1992.
21. Jeho Nam and Ahmed H. Tewfik. Dissolve Transition Detection Using B-Splines Interpolation. *IEEE International Conference on Multimedia and Expo (ICME)*, Vol. 3, pp. 1349 -1352, July 2000.
22. B. Shahraray. Scene Change Detection and Content-Based Sampling of Video Sequences. *SPIE Digital Video Compression, Algorithm and Technologies*, Vol. 2419, pp. 2-13, 1995.
23. B. T. Truong, C. Dorai, and S. Venkatesh. Improved fade and dissolve detection for reliable video segmentation. *IEEE International Conference on Image Processing (ICIP2000)*, Vol. 3, pp. 961 -964, 2000.
24. B. T. Truong, C. Dorai and S. Venkatesh. New Enhancements to Cut, Fade, and Dissolve Detection Processes in Video Segmentation. *ACM Multimedia 2000*, pp. 219-227, Nov. 2000.
25. B.-L.Yeo and B. Liu. Rapid Scene Analysis on Compressed Video. *IEEE Transactions on Circuit and Systems for Video Technology*, Vol. 5, No. 6, Dec. 1993.
26. M. M. Yeung and B.-L. Yeo. Video Visualization for Compact Presentation and Fast Browsing of Pictorial Content. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 5, pp. 771-785, Oct. 1997.
27. H. Yu, G. Bozdagi, S. Harrington. Feature-based Hierarchical Video Segmentation. *IEEE International Conference on Image Processing (ICIP'97)*, Vol. 2, pp. 498-501, 1997.
28. R. Zabih, J. Miller, and K. Mai. A Feature-Based Algorithm for Detecting and Classifying Scene Breaks. *Proc. ACM Multimedia '95*, San Francisco, CA, pp. 189-200, Nov. 1995.

29. R. Zabih, J. Miller, K. Mai. A Feature-based Algorithm for Detecting and Classification Production Effects. *Multimedia Systems*, Vol. 7, pp. 119-128, 1999.
30. H.J. Zhang, A. Kankanhalli, S.W. Smoliar. Automatic Partitioning of Full-Motion Video. *Multimedia Systems*, Vol. 1, No. 1, pp. 10-28, 1993.

Photo and Biliography

RAINER LIENHART received the Diploma in Computer Science and Applied Economics from the University of Mannheim, Germany in 1994. In the same year he joined the Tenet Group at the International Computer Science Institute in Berkeley for six months and in 1997 he was a visiting researcher in the Visual Computing Lab of Professor Ramesh Jain. In 1998, he received the Doctoral degree in computer science from the University of Mannheim, Germany on 'Methods for Content Analysis, Indexing and Comparison of Digital Video Sequences'. He was a core member of the Movie Content Analysis Project (MoCA). Since 1998 he is a Senior Researcher at Intel's Microcomputer Research Lab in Santa Clara working on video technologies beyond simple video encoding/decoding, media management and multimedia peer-to-peer systems.