Temporal Segmentation of Video Using Frame and Histogram Space

Robert A. Joyce, *Member, IEEE*, and Bede Liu, *Fellow, IEEE*

***Abstract*— Two algorithms are presented for the detection of gradual transitions in video sequences. The first is a dissolve detection algorithm utilizing certain properties of a dissolve's trajectory in image-space. It is implemented both as a simple threshold-based detector, and as a parametric detector by modelling the error properties of the extracted statistics. The second is an algorithm to detect a wide variety of wipes based on image histogram characteristics during such transitions. Both algorithms operate in the compressed domain, requiring only partial decoding of the compressed video stream. Experiments show the algorithms perform as well as—and in some cases, better than—full-frame methods, on a wide variety of gradual transitions, and can operate significantly faster than real-time.**

***Index Terms*— Compressed Video Processing, MPEG Video, Shot Detection, Wipe Detection, Dissolve Detection, Scene Change Detection, Gradual Transition Detection**

I. Introduction

CONTENT analysis of digital video is of central importance in the creation of indexing, browsing, and searching mechanisms for video databases. An essential first step is the segmentation of new streams via production cues such as scene and shot boundaries. As much of the video will be in compressed form, and computational expense is an important consideration, processing of the video in the compressed domain is desirable.

The detection of abrupt transitions ("cuts") between shots has been extensively studied in both the compressed and uncompressed domains. Gradual transitions, which are more likely to mark scene boundaries than are cuts, pose a much more difficult problem. Such transitions can be roughly divided into two classes: those that simultaneously but gradually affect every pixel of the image, and those that abruptly affect an evolving subset of the pixels, with the subset changing in each frame. Over a number of frames, the cumulative change—due to the summed gradual changes or to the union of pixel subsets—forms the gradual shot boundary.

Manuscript received July 12, 2001; revised September 13, 2003. The associate editor coordinating the review of this paper and approving it for publication was Dr. Wayne Wolf.

This work was supported in part by a New Jersey State R & D Excellence Grant, NSF Grant MIP-9408462, and an Intel Technology for Education 2000 Grant. This material is also based upon work supported in part by the U.S. Army Research Office under contract DAAG55-97-10208.

R. A. Joyce was with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA. He is now with ATC-NY, 33 Thornwood Drive, Suite 500, Ithaca, NY 14850 USA (e-mail: rob@atc-nycorp.com).

B. Liu is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: liu@princeton.edu).

EDICS 4-SEGM

Digital Object Identifier 10.1109/TMM.2005.861285

The first class includes dissolve and fade-in/out effects, and one could argue that dissolves and fades are the only members of this class. Much work has been done on dissolve and fade detection, particularly with the use of reduced-resolution frames and motion vectors gathered directly from the compressed stream [1]–[8].

What are commonly thought of as wipe effects are members of the second class, although for compactness the term "wipe" will herein be used to describe any transition abruptly affecting an evolving subset of pixels. Wipes are often used in television news and sports coverage, as well as in movies. In sports video, for example, wipes are generally used to denote the beginning and end of an instant replay; thus, detection of wipes would allow an indexing system to separate replays from live action, thereby preserving continuity in time. During newscasts, wipe transitions often signal a change in story or topic.

Qualitatively, wipe transitions are generally characterized by the slow sliding in or uncovering of an image from a new shot, while simultaneously covering up or sliding out the old shot. At any instant during the transition, the frame contains some of the old content, as well as some of the new. The "edge" of the wipe—the moving spatial boundary between the old and new shots—can be a single line or a complex pattern. Recently, there has been a trend toward using blurred wipe edges; attempting to detect the exact wipe edge can be difficult. Often, the transition is generated by computer, in which case three-dimensional projections or other special effects may be present. On occasion, computer-generated artwork will completely cover the image, creating an intermediate step in the wipe's progression.

One common method of wipe detection involves extracting and counting edges in the image; this statistic will monotonically change during a transition, from the old shot's value to the new shot's value [9], [10]. This generally must be performed on uncompressed video, and is computationally expensive. In the compressed domain, methods have been proposed which analyze a projection or subset of the DC DCT coefficients, looking for progressions of abrupt pixel changes [11]–[13]. Progressions of changes in encoder motion-prediction decisions have also been used [14], as have progressions in partial-frame histogram intersections [15]. A method has been proposed by which the statistical characteristics of wipe sequences are detected [16]. Finally, the Hough transform can be used on spatially-reduced frames to detect and characterize the style of certain types of wipes [17]. The majority of these algorithms depend on the video producer using only a limited amount of computer graphics or artwork,

and assume little motion adjacent to and during the wipe. With the prevalence of computer-generated wipes, assumptions of sharp boundaries and simple one-directional wipe models are likely to fail on modern video; what is needed is a more general method, independent of the direction or style of wipe, and independent of any reasonable amount of producer-added effects (for instance, blurring, page-turning, shadows, and projections).

This paper begins an explanation of and motivation for the compressed-domain techniques used throughout; these are presented in Section II. Section III describes a dissolve and fade detection algorithm which, while not too different from conventional methods, eliminates some restrictions imposed by available dissolve detectors. A general method of detecting transitions from the wipe class, which circumvents some of the limitations imposed by current algorithms, is presented in Section IV. Both the wipe and dissolve detection algorithms can be easily modified to detect partial-frame transitions (for example, caption appearances). The statistics computed by both algorithms are examined in further detail in Section V. Experimental results are described in Section VI, and conclusions, as well as ideas for further study, comprise Section VII.

II. Compressed domain processing

Ideally, the indexing process would be done in real-time, either from a live (streaming) feed or a single pass of a videotape. However, the computation time required to decode MPEG video and perform image-processing operations on full frames, while decreasing with progress in processor design, remains significant[1]. This complexity constraint becomes even more troublesome when considering that shot and scene decomposition are only the first steps of the indexing process; there is much yet to do. In addition, most sizable digital video libraries are likely to be in compressed form, if only for economic reasons. For these reasons, analyzing video streams directly in the compressed domain is advantageous.

One natural technique of compressed-domain analysis is reduced-resolution processing: using a subset of the block DCT coefficients to reconstruct thumbnail-sized images. Of particular interest is the construction of so-called "DC frames," which are comprised of the lowest-order DCT coefficients of each block (and are therefore one eighth the size of the full frames). For intracoded (I) frames, construction of DC frames is trivial. Intercoded (P,B) frames require full decompression of their reference frames for exact DC reconstruction. Instead, rapid first-order estimation techniques are used to construct DC frames for intercoded compressed frames [5]. If computation time is very critical, a slight speedup can be gained by resorting to the simpler zero-order estimation techniques; the negative impact on the final results is small. Similar methods can be used to construct DC+2AC frames, which are formed from the DC and two lowest-order AC coefficients of each block. The 2×2 IDCT then required for each block is simple to compute.

[1]While we will concentrate on MPEG-1 video, the techniques presented apply equally well to other block/transform-based compression schemes.

Aside from their computational advantages, DC sequences are more suitable for video analysis in many respects. Primarily, the artifacts of MPEG compression and video noise are significantly reduced at the lower resolution. In addition, small amounts of camera or object motion, which dramatically affect the registration of adjacent frames' pixels at the full-frame level, are obscured at such a low resolution.

Displaced frame differences ("DFD's"), which are the pixel-by-pixel differences between frames after any motion compensation, can be computed for P frames without full decompression. DC DFD's require no computation at all, as they are just the lowest-order DCT coefficients of the residue frame, which are available directly in the coded stream. Other reduced-resolution DFD's can be computed via low-order inverse DCT's.

Unless otherwise noted, first-order estimated DC sequences are used in all calculations for the remainder of the paper. For MPEG-1 sequences, the DC frames are typically 44×30 pixels in size.

III. Dissolve Detection

At its most basic, a dissolve or fade is a time-varying superposition of two video streams. Let $f_k(x, y)$ denote the value of pixel (x, y) in frame k of sequence f, with $g_k(x, y)$ and $h_k(x, y)$ defined similarly. A dissolve from sequence g to sequence h, lasting from frame m to frame n, can therefore be described by

$$f_k(x, y) = \alpha_k h_k(x, y) + (1 - \alpha_k)\, g_k(x, y) \qquad (1)$$

where α_k is an increasing sequence, with $\alpha_m = 0$ at the beginning of the dissolve and $\alpha_n = 1$ at the end. It is often assumed that the sequence α_k increases linearly, but this is not necessarily the case; particularly artistic dissolves may have a pause, a long lead-in time, or some other non-linearity in α_k.

For the moment, we assume there is negligible motion in the sequences g and h. For compactness, we denote by f_k the vector formed by all the pixels of frame k (the ordering is irrelevant, as long as it is consistent). For color video, each pixel has three dimensions in color space; f_k then contains three times as many elements as there are pixels in a frame. Consider the trajectories formed by $f_b - f_a$ and $f_d - f_c$, where $m < a < b < n$ and $m < c < d < n$. Substituting the model in (1) yields

$$f_d - f_c = (\alpha_d - \alpha_c)\,(\alpha_b - \alpha_a)^{-1}\,[f_b - f_a] \qquad (2)$$

during a dissolve. As α_k is an increasing sequence, $(\alpha_d - \alpha_c)\,(\alpha_b - \alpha_a)^{-1} > 0$. This condition is equivalent to the statement that, during a dissolve, the normalized correlation, ρ, between any two trajectory vectors is 1. If one considers each vector f_k as being in a frame-space, then the video's trajectory in this space will be a straight line during a dissolve, as shown in Figure 1. Natural, non-dissolve motion in a stream generally does not have this characteristic; it is uncommon for all the pixels in the image to evolve in the same way, frame after frame. Note that linearity in frame space is distinct from the condition that α_k increases linearly; we make no such assumption about the time progression of the dissolve.

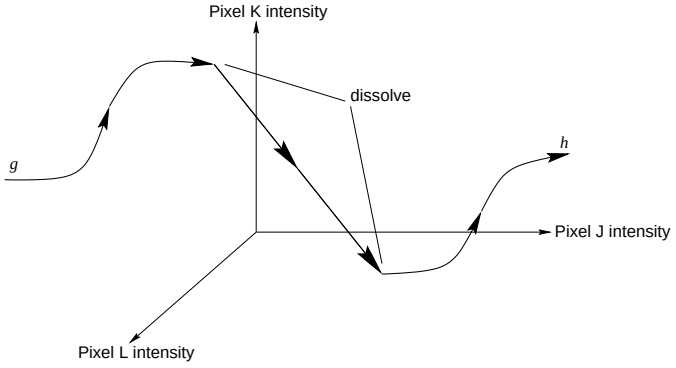

Fig. 1. Three-dimensional representation of a video sequence f_k in frame-space during a dissolve.

In order to check this condition, we are faced with four concerns: limited memory (we cannot store all the frames), limited computation time, no *a priori* knowledge of the start or end of the dissolve, and the fact that there may be some object or camera motion in the frame. Analysis of three nearby frames at a time offers a good compromise among these considerations. Using frames $k-L$, k, and $k+L$, we can compute two length-L frame differences, where

$$d_k^L(x,y) = f_k(x,y) - f_{k-L}(x,y) \quad \forall x,y \tag{3}$$

is the k-th difference frame, and d_k^L is the corresponding vector in frame space. The correlation, as a function of k and L, is then

$$\rho(k,L) = \frac{\left\langle d_{k+L}^L, d_k^L \right\rangle}{\sqrt{\left\| d_{k+L}^L \right\|^2 \left\| d_k^L \right\|^2}} \tag{4}$$

(where $\langle \cdot, \cdot \rangle$ represents inner product). A 'straight' triplet of frames is declared if the the correlation is high enough, i.e., if

$$\rho(k,L) \geq T_{corr} \tag{5}$$

for an appropriate threshold T_{corr}.

In order to declare a dissolve, we require that condition (5) hold for every k in some sequence of frames, say from m to n. A condition on the length of this line in frame space is also needed; we require that

$$\|f_n - f_m\| \geq T_{dist}. \tag{6}$$

The length condition is necessary because small changes (e.g., in frame brightness) can lead to the correlation condition being met for an isolated triplet or two. Instead of (6), a simpler threshold on the length in number of frames can be used, but the frame-space length condition is more robust in eliminating false detections.

The testing of (5) and (6) can be done sequentially, with no knowledge of future frames beyond $k+L$, according to the following algorithm:

```
while (there are new frames)
    if (ρ(k,L) ≥ T_corr)
        n = k + L
        if (m not yet set)
            m = k − L
        else if (‖f_n − f_m‖ ≥ T_dist)
            declare dissolve
    else
        unset m
    k = k + L
end
```

Regarding the selection of L, we note that the effects of motion diminish as $L \to 1$, but decreasing L leads to more false alarms, as it is possible to construct a long non-straight line in frame space which has local correlations near 1. As L is increased, computational requirements are lessened, but it becomes more likely that outliers (from a straight line) will be obscured by the coarse granularity of sampled frames. Selecting $L = 3$, which means only the I and P frames in many MPEG-1 streams, provides a resonable compromise: slow motion is not destructive, and the computation time and number of false alarms are both reasonable.

While many dissolves do indeed have little motion, this is not universally true; any rapid object or camera motion during the transition will prevent the frame-space linearity condition from holding. (Local linearity might still hold though, if L is small.) By using DFD's, described in Section II, instead of the true frame differences, simple object or camera motion can be compensated for. Unfortunately, this places a dependency on how the particular MPEG encoder was designed; to maintain some consistency among computed correlation values, we restrict analysis to only P frames[2]. In addition, much computation is eliminated, due to the ease of extracting (DC) DFD's. If we denote the DFD between frame f_k and frame f_{k-L} as $\tilde{d}_k^L$, the correlation calculation in (4) becomes

$$\rho_{dfd}(k,L) = \frac{\left\langle \tilde{d}_{k+L}^L, \tilde{d}_k^L \right\rangle}{\sqrt{\left\| \tilde{d}_{k+L}^L \right\|^2 \left\| \tilde{d}_k^L \right\|^2}}. \tag{7}$$

A plot of this ρ_{dfd} sequence for a sample documentary clip with two dissolves is shown in Figure 2; note the sharp increase during the dissolve frames.

Values for T_{corr} and T_{dist} should be set based on the desired false alarm rate or detection accuracy. In many cases, false alarms are not as detrimental as missed events in shot decomposition; detection accuracy can be improved if some false alarms are allowed. As the values of the frame-space correlations can depend on non content-related factors such as frame size, video noise, and compression artifacts, the mean of the past M values of $\rho_{dfd}(k,L)$ is subtracted before the T_{corr} comparison is made. (Subtracting this mean is equivalent to gently high-pass filtering the ρ_{dfd} sequence.) More thorough post-processing of the correlation sequence is detailed in Section V, and specific experimental results are presented in Section VI.

IV. Wipe Detection

One can imagine many effects in which an evolving subset of pixels changes abruptly in each frame. The simplest wipes

[2]This requires us to ignore two triplets per GOP, namely the PPI and PIP, because one of the two required DFD's cannot be reliably computed in each case.

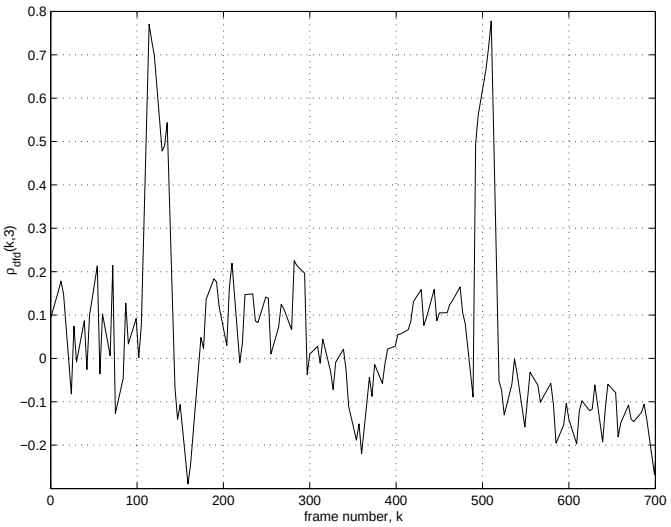

Fig. 2. The DFD-based correlation sequence $\rho_{dfd}(k, L)$ for a segment of documentary video, with $L = 3$; dissolves occur during frames 115–140 and 492–516.

Fig. 3. Sample wipe sequences from network television, showing the wide variation possible in computer-generated effects.

are those in which one sequence gradually covers or replaces another, with no global movement of either sequence. More complicated wipes can involve one stream "sliding" in over another, or one "pushing" another aside. "Zoom" based wipes can also be created in this manner, with a new stream appearing from the center of the old one, expanding to fill the whole frame. Finally, complex computer-generated wipes can include page-turning effects, projections, or artistic wipe boundaries (for a few examples, see Figure 3). One or more frames may not even contain content from either adjacent shot; this is particularly common in sports video, where a large computer-generated "object" passes across the field of view to effect a transition using two back-to-back wipes. Due to the broad range of gradual transitions that fall within the wipe class, a detection method tailored to a specific wipe is likely to miss many other kinds of wipes.

As in the dissolve case, we assume a wipe transition from sequence g to sequence h, from frame m to frame n. A simple, overlap-based wipe can be described as

$$f_k(x,y) = I_k(x,y)h_k(x,y) + [1 - I_k(x,y)]\, g_k(x,y) \quad (8)$$

where f_k is the resulting frame k, and I_k is either 0 or 1 for each k, x, and y. $I_k(x,y) = 0$ for all x and y when $k < m$ (i.e., before the wipe), and $I_k(x,y) = 1$ when $k > n$ (i.e., after the wipe). In the case where one or both sequences slide in or out of the frame, (8) becomes

$$\begin{aligned} f_k(x,y) \;=\; & I_k(x,y)h_k\left(x + x_{h,k}, y + y_{h,k}\right) \\ & + [1 - I_k(x,y)]\, g_k\left(x + x_{g,k}, y + y_{g,k}\right) \end{aligned} \quad (9)$$

where the wipe-induced motion of the sequences is described by $x_{g,k}$, $y_{g,k}$, $x_{h,k}$, and $y_{h,k}$. These two models are more restrictive than one would like; they preclude the detection of many artistic wipes, for example. Natural object motion in video typically fits these models as well, yielding only limited usefulness. The important information of each model is that contained in the sequence $I_k(x,y)$; as such, we will concentrate on $\|I_k\|$. This sequence should increase from 0 to N, the number of pixels in the image, as k increases from m to n. For most wipes, $\|I_k\|$ will increase linearly or quadratically.

One representation of a video sequence that allows us to examine the $\|I_k\|$ sequence, without the restrictions of specific wipe models, is the histogram. We denote the p-th bin of frame f_k's histogram as $F_k(p)$ (the number of bins, P, is a free parameter); we will use the same vector shorthand of F_k, for some arbitrary ordering of bins. Assuming for the moment that each frame's histogram is fairly uniform across different portions of the image, the histograms during a wipe can be expressed as

$$\begin{aligned} F_k(p) \;=\; & \left(\frac{\|I_k\| + E_{G,k}(p)}{N}\right) G_k(p) \\ & + \left(1 - \frac{\|I_k\| + E_{H,k}(p)}{N}\right) H_k(p), \end{aligned} \quad (10)$$

where $E_{G,k}(p)$ and $E_{H,k}(p)$ are error terms resulting from the spatial nonuniformity of the histograms of g and h, respectively. Note that this histogram-based wipe model has the same form as the frame-space model (1) for a dissolve! If the values of $E_{G,k}$ and $E_{H,k}$ are small and fairly constant in k, it also meets the conditions we imposed on the coefficients α_k from the dissolve case. Specifically, the quantity

$$\beta_k = \frac{\|I_k\| + E_{G,k}(p)}{N} \quad (11)$$

will be increasing in k from 0 to 1, and

$$1 - \frac{\|I_k\| + E_{H,k}(p)}{N} \approx 1 - \beta_k. \quad (12)$$

Such a parallel immediately suggests a wipe detection algorithm. As in the dissolve case, the correlation between any two histogram difference vectors ($F_b - F_a$ and $F_d - F_c$) will be 1 during an ideal wipe. Moreover, a wipe will appear

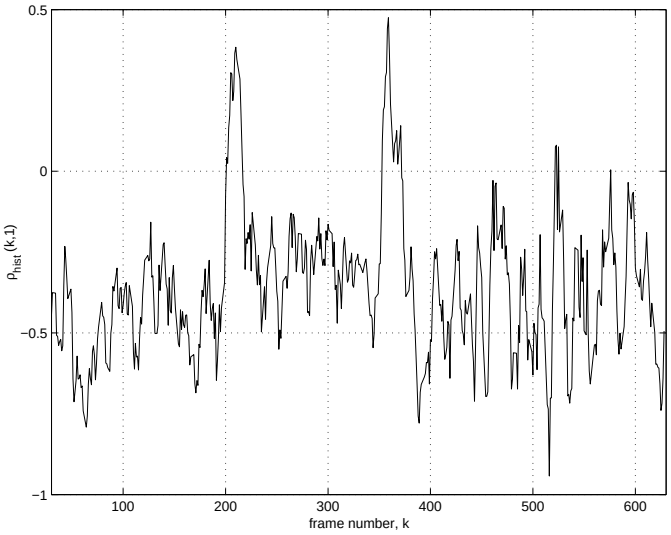

Fig. 4. The sequence $\rho_{hist}(k, L)$ for a segment of news video containing two wipes. $L = 1$ in this case, and wipes occur in frames 197–209 and 352–364.

as a straight line in a histogram-space, where each dimension corresponds to one bin of the histogram. (This linearity is independent of any nonlinearity in the time progression of the wipe.) In the same manner as the dissolve case, we define the l-frame histogram difference $D_k^l(p)$ as

$$D_k^L(p) = F_k(p) - F_{k-L}(p) \quad \forall p. \tag{13}$$

We compute the correlation sequentially, from triplets of frames:

$$\rho_{hist}(k, L) = \frac{\left\langle D_{k+L}^L, D_k^L \right\rangle}{\sqrt{\left\| D_{k+L}^L \right\|^2 \left\| D_k^L \right\|^2}}. \tag{14}$$

This value is compared to a threshold, and following the pseudocode presented for the dissolve algorithm, the value of (6) is computed to determine the length of the candidate wipe; a wipe transition is declared if both thresholds are met. Note that a condition similar to (6) could be computed in the histogram space; we have not done so, due to the unwanted constraint this imposes that the two adjacent shots must have sufficiently different histograms. The sequence ρ_{hist} for a sample stream is given in Figure 4.

As in the dissolve case, T_{corr} and T_{dist} should be chosen to achieve the desired weighting between detection probability and false alarm rate. Once again, to counter the fact that the mean value of ρ_{hist} is somewhat dependent on the type of video and the recording/compression quality, the mean of the last M values is subtracted before thresholding with T_{corr}.

Any natural change in g or h's histograms through time (due to motion or other effects) introduces deviation from $\rho_{hist} = 1$ in the same manner as motion in g or h did in the dissolve case. As L is decreased, $G_k - G_{k-L} \to 0$ and $H_k - H_{k-L} \to 0$, because the g and h histograms are very unlikely to change abruptly (except in the case of a scene cut).

In addition, as L is decreased, the effect of error terms $E_{G,k}$ and $E_{H,k}$ in (10) diminishes. This is due to the use of triplets: the deviation from spatial uniformity in the histogram is only important in the region that actually changes over the frame interval $k - L$ to $k + L$. In a wipe, the size of this region (a vertical slice of the image, for example) vanishes as $L \to 0$. For these reasons, we set $L = 1$ from here on; this agrees with experimental results obtained by varying L. In cases where the histogram non-uniformity is caused by an object's edge entering or exiting the region/slice of significance, the effect on ρ_{hist} will be impulsive—the straight line in histogram space will now be piecewise linear, with some small number of vertices.

Equation (10) makes a computational assumption: the number of pixels in any histogram must be an integer, yet the coefficient β_k may be such that the equation requires a non-integral number of pixels in a particular bin. This quantization error, if significant, can reduce the correlation among the adjacent pair of vectors in a triplet. The error can be reduced by using fewer histogram bins, as well as by increasing the spatial resolution at which one operates (using a low resolution or a large number of bins would force very small quantities of pixels into many bins, making any quantization errors in the intermediate frame of a triplet more significant). For this reason, we perform the histograms on DC+2AC frames and use 2 to 4 histogram bins per color dimension (8–64 total).

While better characterizing these data-dependent quantization and histogram non-uniformity errors remains an open problem, their effects on ρ_{hist} can be reduced by low-pass filtering or otherwise post-processing the resulting correlation sequence (the assumption here being that the errors in $\rho_{hist}(k, L)$ are approximately independent in k). The generally impulsive errors due to $E_{G,k}$ and $E_{H,k}$ suggest the use of a median filter (or, more generally, an n^{th}-largest filter), which while nonlinear, does have a sufficiently low-pass characteristic to help with the quantization noise. (As an added benefit, low-pass filtering helps alleviate the time-varying histogram distortions that MPEG compression and video noise can introduce.)

One issue has not yet been addressed: can natural motion in video have this linear histogram-space characteristic? Pathologically-structured object motion into or out of a frame can cause a straight line in the histogram-space, as can panning the camera if the image contents and histograms change radically during the pan. Experimentally, the number of false alarms attributed to object motion has been shown to be fairly small in natural video, provided the image histogram does not change radically during pans. False detections due to panning can only be eliminated at the expense of missing "push" type wipes (which are arguably a type of panning). This can be done by computing the temporal variance of each macroblock's motion vectors—low variance corresponds to constant motion in some direction, through time. In this case, a wipe is declared only if there are enough macroblocks with high temporal variance over the period of the candidate wipe. False alarms could be further reduced by adding additional constraints; one example is requiring the ratio of $\left\| D_k^L \right\|$ to $\left\| D_{k+L}^L \right\|$ to be either constant or linear.

In practice, we find that dissolves are often falsely detected as wipes by our algorithm. A dissolve does not have the linearity property in histogram space; rather, the histogram

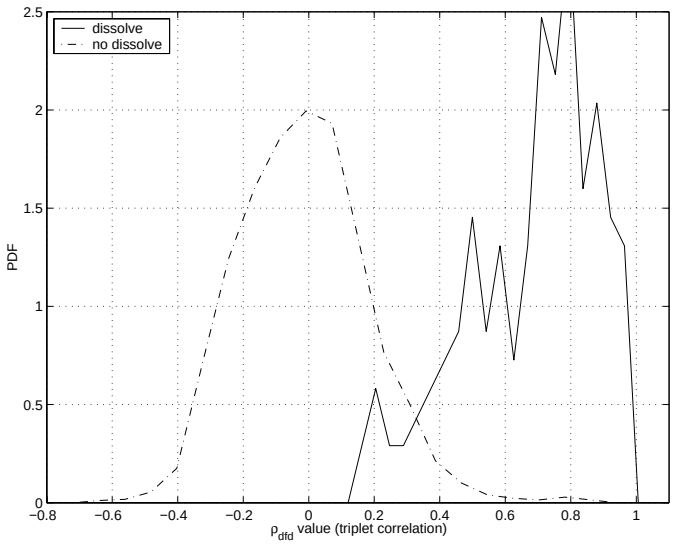

Fig. 5. Triplet correlation values ρ_{dfd} for dissolve (solid line) and non-dissolve (dashed line) segments. The sample variance for the dissolve segments is 0.0406; for the non-dissolve segments, 0.0387.

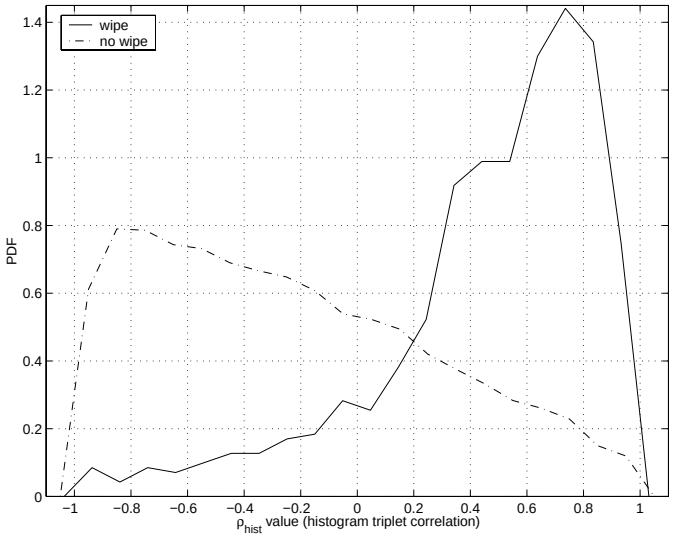

Fig. 6. Triplet histogram correlation values ρ_{hist} for wipe (solid line) and non-wipe (dashed line) segments. The sample variance for the wipe segments is 0.1386; for the non-wipe segments, 0.2175.

of the old shot is progressively shifted, bin by bin, toward all pixels being in the "black" bin; the new shot is correspondingly shifted binwise from black to its final histogram. However, if the shots' histograms are fairly continuous from bin to bin (i.e., there are no spikes in particular bins, while other bins are nearly empty), then the dissolve can masquerade as a linear change in histogram space. This is particularly a problem when the number of bins is small; spikes are very unlikely when there are only a few bins. The simplest solution is to cascade the transition detectors: only try to detect wipes in areas previously declared not to be dissolves. While this introduces a two-level detection dependency—dissolve misses will contribute to wipe false alarms—the result is well worth it.

V. Analysis of the Correlation Statistic

Given that the detection algorithms introduced in Sections III and IV are so similar once the correlation statistics are computed, it is useful to study the ρ sequences' statistics. Any information gained can be used to derive a more optimal (yet computationally expensive) detector.

Figure 5 shows the distributions of ρ_{dfd} values for 13 minutes of video, separated into dissolve and non-dissolve segments (after filtering out cuts); Figure 6 is the wipe case. The overlap in densities is not as bothersome as it might appear, because detection is done on variable-length sets of frame triplets (using the algorithm presented in Section III), not on individual triplets.

One interpretation of the distributions, particularly those of ρ_{dfd}, is that of a signal+noise detection problem, where the signal of interest (denoted $s(k)$) is a binary indicator of whether there is a transition during the triplet. (In the wipe case, the clipping of the values to ± 1 causes the noise distribution to not be independent of $s(k)$; this could be alleviated by more sophisticated detection-theoretic techniques.) If we subtract $s(k)$ from $\rho_{dfd}(k)$, we find the measurement error,

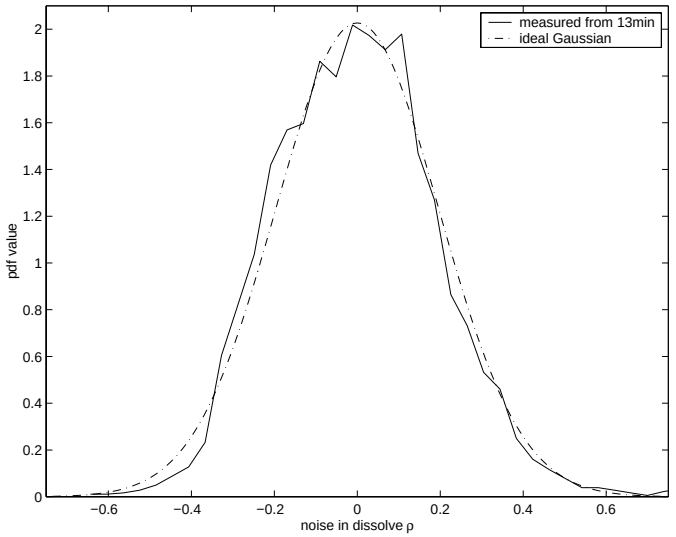

Fig. 7. ρ_{dfd} "noise", after the dissolve indicator signal $s(k)$ is subtracted.

due to motion, compression, etc., is nearly an ideal Gaussian process with sample variance 0.0387 (Figure 7). This process $n(k)$, where $n(k) = \rho_{dfd}(k) - s(k)$, is not white noise—its power spectrum is tilted toward DC—but is fairly independent of $s(k)$. (The coarseness of Figure 5's distribution within the dissolve region is due to the relatively small number of triplets in it.)

The length of the transition is unknown during the detection process, yet we would like to take advantage of the interdependence of the $s(k)$ indicator values; this can be done using a parametric detector, which averages over some given distribution of test signals. In this case, the parameter is the transition length; an estimated distribution of dissolve lengths, measured from 95 transitions, is shown in Figure 8. We call this PMF $w(m)$. Note that, even within the framework of the simple detector presented in Section III, this distribution could

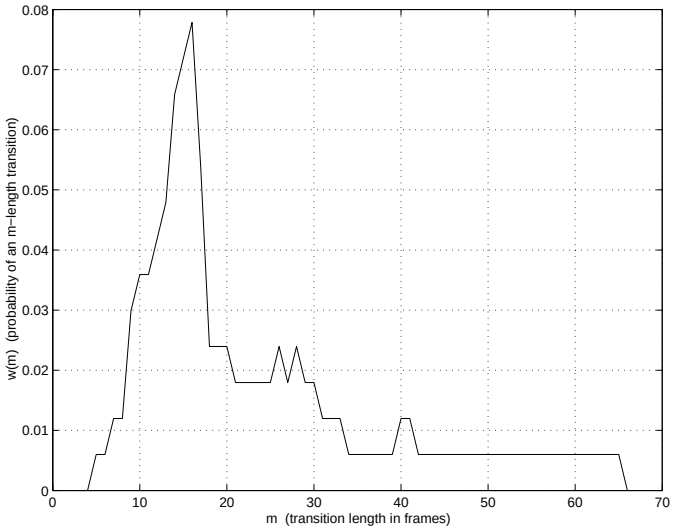

Fig. 8. Distribution of dissolve transition lengths, measured from 95 dissolves. Some values are interpolated from neighboring samples.

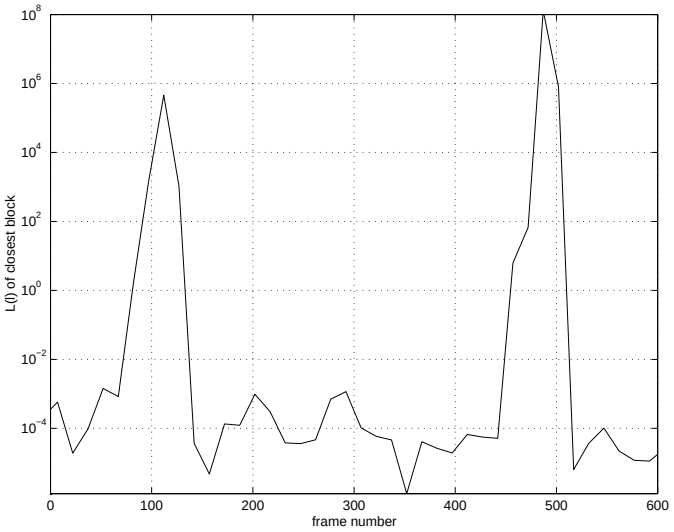

Fig. 9. Dissolve likelihood function $\mathcal{L}(l)$, expressed in terms of the frame numbers beginning each block of triplets l; note the logarithmic scale. The MPEG video stream is the same as in Fig. 2.

be used to eliminate inordinately short or long false detections.

Optimum parametric detectors in non-i.i.d. Gaussian noise are well known [18]. Such detectors by necessity work on blocks of input data, where the block in our case must be longer than the support of the parameter's PMF. Denote this block length K; $w(m) = 0$ for $m > K$. In order to make each block's noise statistics consistent, we require that blocks begin at a GOP boundary; otherwise, we have a non-stationary noise source as the blocks' starting points shift through one GOP. Block l therefore contains time indices (frames) lG through $lG + K - 1$, where G is the length of a GOP.

We begin by setting μ_0 to be the (estimated) mean of the ρ_{dfd} values when no transition is present; μ_1 is the mean during dissolves and fades. Denote by $R(l)$ the column vector of measurements formed by $\rho_{dfd}(k) - \mu_0$, where $lG \le k \le lG + K - 1$. We then construct a parameterized set of K-length test signal vectors, $S(p, q)$, $0 \le p \le G - 1$ and $1 \le q \le K - p$, where G is the length of a GOP:

$$S(p,q) = [\underbrace{0 \cdots 0}_{p} \underbrace{\mu \cdots \mu}_{q} 0 \cdots 0]^T \tag{15}$$

where $\mu = \mu_1 - \mu_0$. Essentially, S contains every possible transition length of interest, with starting points anywhere within the first GOP (starting points within later GOP's will be detected in subsequent blocks' tests). The stationary block-based problem can then expressed as $R(l) = N(l) + S(p, q)$ for some p and q. The density $w(m)$ must mapped into $W(p, q)$ according to the length of the transition tested in $S(p, q)$, giving[3] $W(p, q) = w(q)$.

As the noise sequence $n(k)$ is not i.i.d., neither is the noise vector $N(l)$, so both the received signal blocks $R(l)$ and the test signal vectors $S(p, q)$ must be prewhitened; denote the whitened vectors $\bar{R}(l)$ and $\bar{S}(p, q)$, respectively (the whitening can be done by estimating the covariance matrix Σ_N and multiplying by one of its Cholesky factors) [18].

Given the set of $\bar{S}$ vectors, the block PMF $W(p, q)$, and the prewhitening matrix (C^{-1}), the transition likelihood function can be calculated for each block of correlations $R(l)$ as follows:

$$\mathcal{L}(l) = \sum_{p,q} W(p,q) \exp\left[\left(\bar{S}^T(p,q) C^{-1} R(l)\right) - \frac{1}{2}\left(\bar{S}^T(p,q)\bar{S}(p,q)\right)\right] \tag{16}$$

A sample plot of $\mathcal{L}(l)$ in Figure 9 shows that this approach extracts dissolves quite well from the ρ_{dfd} sequence. A histogram of $\mathcal{L}(l)$ for dissolve and non-dissolve segments is shown in Figure 10 (note the log scale); when compared to Figure 5, the improvement is clear. As argued in Section III, it is also necessary to test the L^2 length of a candidate dissolve in frame space before declaring a transition. Specifically, a dissolve transition is declared if both the following inequalities hold:

$$\begin{aligned} log_{10}\mathcal{L}(l) &> T_{\mathcal{L}} \\ \text{and} \quad \|f_{lt+Q-1} - f_{lt}\| &\ge T_{dist}, \end{aligned} \tag{17}$$

where Q is the largest term (over values of q) in the sum (16). In practice, transitions often span multiple GOPs, so that $log_{10}\mathcal{L}(l) > T_{\mathcal{L}}$ for two or more consecutive l values; in such cases, we declare the frames corresponding to the largest $\mathcal{L}(l)$ value as the dissolve transition. Given our "on-then-off" prototype sequences $S(p, q)$, the (locally) largest $\mathcal{L}(l)$ during a dissolve is generally the first one.

Recall that, in order to gain the stationarity required to make this parametric detector reasonable, we dropped the temporal resolution of the detector to the length of a GOP by using blocks of ρ_{dfd} values. If a greater temporal resolution is required, the largest term of the sum in (16) can be used as an estimate of the parameters p and q best fitting the transition; p and $p + q$ give the exact start and ending frames. Section VI shows some results of this more complex detection algorithm.

[3]If certain triplets are skipped, such as the GOP boundary triplets mentioned in Section III, this equation must be modified to account for the non-constant time increments. This complication is ignored for clarity's sake in the paper.

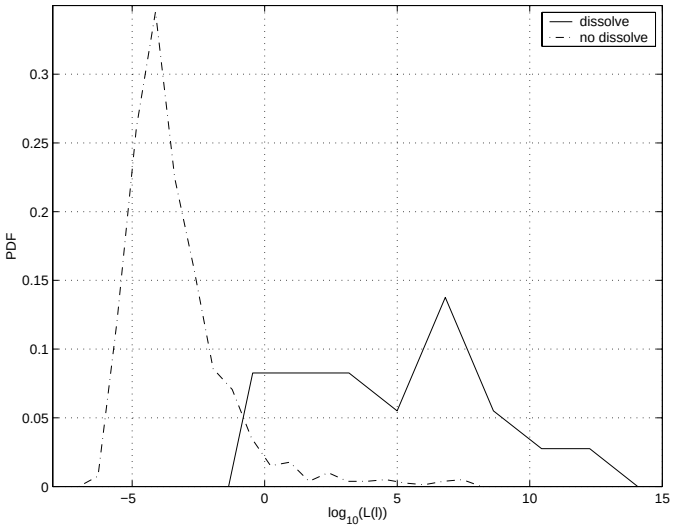

Fig. 10. Distribution of $\mathcal{L}(l)$ values for dissolve and non-dissolve segments of 13 minutes of video; again note the log scale.

This parametric detection structure could also be useful for the more general case of gradual transition detection, where some statistic is computed per frame (or set of frames) and the probability density of transition lengths is known or estimated, provided that the "noise" in measurements is approximately Gaussian and, more crucially, independent of $s(k)$. As can be seen in Figure 6, the noise in ρ_{hist} is highly dependent on $s(k)$ (and is not Gaussian); applying the parametric detector unaltered indeed yields poor results. In the absence of tractable independent noise models, the only way to improve detection with ρ_{hist} is to compare sample densities to the two in Figure 6. Such a comparison is likely to be problematic, as the number of samples within a given block is small (and there is no way to refine the temporal resolution beyond the block level). ρ_{hist} could certainly benefit from further analysis, however; it is possible for example that some neural network or adaptive filter structure exists which will improve the results.

VI. Experimental Results

Each algorithm was tested with "natural" television and film footage, digitized from VHS tape sources with a hardware MPEG-1 encoder. The resulting video quality is hardly perfect, making for a good workout of each transition detector. Each test stream was digitized at a resolution of 352×240 and a frame rate of 29.97 fps. The test sets consisted of news video from different networks, documentary footage, and other material. All computation was performed on a 350 MHz Sun workstation.

A. Dissolve Detection with Simple Detector

A thirteen minute set of video was used as a "training" set, on which parameter values were selected. The training set's 23 700 frames contained 59 dissolves, 115 cuts, and 6 wipes, as well as some significant object and camera motion. Most of the dissolves were clearly visible, but three were between images so similar that a casual human viewer likely would not notice the transition. The dissolves ranged in length from 12 to 65 frames, and a number of the transitions contained motion of some sort.

Testing yielded good results with $T_{corr} = 0.15$, after the mean of the past 125 values was subtracted. (Depending on the video, the effective T_{corr} was between 0.4 and 0.8.) $T_{dist} = 55000$ (normalized to the number of macroblocks per frame) and at least 3 successive above-threshold triplets were required in order to declare a dissolve. With these values, 52 out of the 59 dissolves were properly detected, with 24 false alarms (a rate of one per 2633 frames). Different thresholds can be selected to yield different detection probability versus false alarm tradeoffs. In most cases, the detector correctly identified the locations of the transition start and end to within four frames (in ignoring B frames, the theoretical best accuracy is three frames).

These results confirm that frame-space correlation is a reasonable statistic to use for dissolve detection, and that using DFD's is an effective way to combat the errors in correlation introduced by shot motion. By visual inspection of $\rho_{dfd}(k)$ plots, one can generally pick out all of the dissolves (even the ones that are missed); this leads us to believe that a more sophisticated detector could produce better results using the same ρ_{dfd} sequence. Results for such a detector, that described in Section V, are presented in the next sub-section.

Including the overhead due to DC frame extraction, our algorithm processed video at about 170 frames per second. In fact, about 95% of the processing time is spent parsing the MPEG stream and calculating the DC frames; once the DC frame is available, our algorithm takes only an additional 0.3 ms/frame on the test machine. Speed in parsing could likely be improved through better optimization of our partial MPEG decoder.

B. Dissolve Detection via Parametric Detector

Once the $\rho_{dfd}(k)$ sequence is computed, the detector outlined in Section V can be applied as an alternative to the simple detector. Using the same 23 700-frame test sequence, $T_{\mathcal{L}} = 0.1$, and $T_{dist} = 55000$, the parametric detector correctly finds 53 out of 56 dissolves, with 12 false alarms[4]. Of the three missed dissolves, one falls at the very end of a GOP and is short enough to not affect any DFD's; another overlaps slightly with a wipe transition. The final missed transition is likely due to unfortunate motion-compensation decisions on the part of the encoder, such that the DFD's did not suggest gradual changes in each pixel. Seven of the 12 false alarms are due to cuts, which could be eliminated if a cascaded "cut-dissolve-wipe" detector were implemented; three more are semantically "dissolve-like" operations, such as captions or computer-graphic effects fading away. The total number of false alarms, if one discounts those due to cuts and dissolve-like effects, is 2 per 13 minutes of video.

[4]The total number of dissolves is 56, not 59 as in the previous test, because 3 of the dissolves occur within K frames of the end of a stream, thus are never properly tested against the $S(p, q)$ sample vectors.

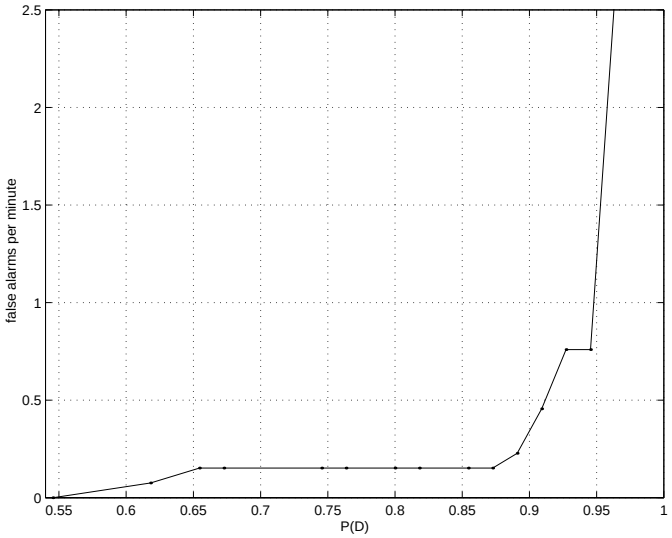

Fig. 11. ROC plot for the parametric dissolve detection method on a 13 minute (23 700 frame) television video sequence. The false alarm rate is the number of false detections divided by the total sequence length; caption fades and similar effects, when detected, were counted as false alarms.

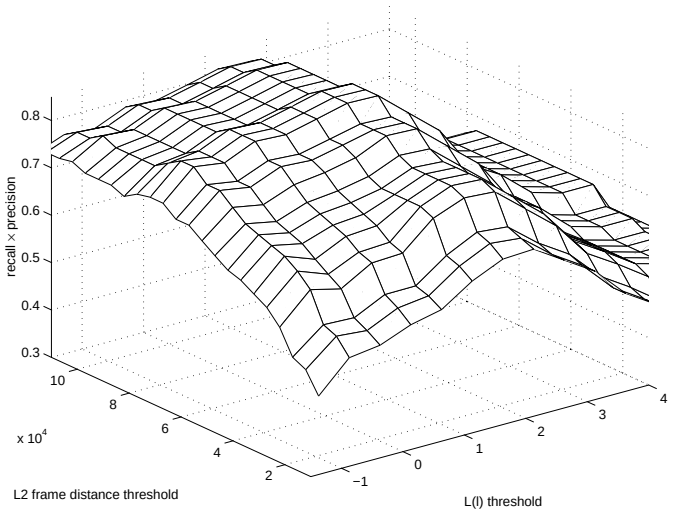

Fig. 12. recall × precision for the parametric dissolve detector, over range of $T_{\mathcal{L}}$ and T_{dist}; the maximum, 0.8395, occurs at $T_{\mathcal{L}} = 1.5$ and $T_{dist} = 67000$.

At the expense of a higher false alarm rate, the detection probability can be pushed up to 0.982; 31 false alarms are produced in this case. Discounting caption fades and cuts, the false alarm count falls to 15 (slightly more than one per minute). Different detection/false alarm tradeoffs are possible; Figure 11 shows the raw false alarm rate corresponding to a number of detection probabilities.

An alternative quality measure is to find the maximum achievable $Q = (\text{recall} \times \text{precision})$ value, where recall is the detection probability, and precision is the number of correct detections divided by the total number of detections (correct or not). A perfect detector is one with $Q = 1$. (In practical applications, the quantity of false alarms is not as critical as the weighting given to them in the Q measure defined here, but this metric is a reasonable basis for comparison.) Automated detection and false alarm counting techniques are used to iteratively find the maximum Q for the parametric detector; the maximum, 0.8395, occurs when $T_{\mathcal{L}} = 1.5$ and $T_{dist} = 67000$, yielding a detection probability of 0.891 and a false alarm rate of 0.228 per minute. The Q values over a small range of $T_{\mathcal{L}}$ and T_{dist} are shown in Figure 12.

The parametric detector was also tested on longer streams, a total of 23 additional minutes; in addition to news and documentary footage, the longer streams contained a number of commercials. Blindly using $T_{\mathcal{L}} = 0.1$ and $T_{dist} = 55000$, 108 out of 149 dissolves are detected, with 36 false alarms (1.57 per minute). Discounting false alarms due to cuts and caption fades, the number drops to 24 (1.04 per minute). If necessary, further processing can be done to cut the number of false alarms (for example, requiring each triplet's frame space vectors have at least a certain L^2 length during a dissolve, or utilizing other statistical properties dissolve transitions must have). Most of the missed detections occur during the commercial segments, where large numbers of dissolves occur right after one another (often, with very few non-dissolve frames in between). In fact, as designed, our parametric detector will not detect more than one transition within K frames (which alone accounts for 10 misses); by adding more complicated 1/0 sequences to the test set $S(p, q)$, these could be detected at some computational expense. (One could make an argument that the short non-dissolve segments in cases like this are not "shots" in the same sense as those in non-commercial segments.)

In most cases, within each test segment, the false alarms have lower likelihood function values than the real transitions (yet still above threshold); very few non-transition regions induce higher likelihood function values (16) than the smallest dissolve segment's value. The cutoff region in $T_{\mathcal{L}}$ varies somewhat by stream, generally according to the content and the MPEG encoder. (This variation is significantly less than the stream-dependent variation in the simple detector's optimal T_{corr}.) One way to both address this issue and to provide the user with more control over the coarseness of the temporal segmentation is to have a presentation-time control of $T_{\mathcal{L}}$. The user would increase the $T_{\mathcal{L}}$ "knob" for coarser segmentation (avoiding caption fades, etc), and decrease it to see the less dramatic gradual transitions (for instance, those involving only a portion of the screen). This operation would require efficient storage of the $\mathcal{L}(l)$ sequence, and fairly rapid threshold testing and presentation, as the threshold would be unknown at the time of analysis.

When the largest term of the sum in (16) is examined as suggested at the end of Section V, the temporal accuracy of the derived start and stop points for each dissolve is generally better than 4 frames. Given that the B frames are ignored, it is impossible to be more accurate than 3 frames on average with our test streams.

As most of the computation time is spent extracting DC frames and DFD's, the increased computational burden of this detector is minimal (on the order of K additional multiply-adds per GOP).

Applying the parametric detector to other dissolve detection

techniques is of interest as future work. Such a test—using the alternative statistics extracted from the raw video stream, then the same parametric detector—would be a fairer comparison of our scheme with existing algorithms than would be a simple test with empirically-determined absolute thresholds. The test would then "factor out" the effects of threshold selection, comparing only the underlying dissolve models and extraction of dissolve-related statistics.

C. Wipe Detection

In testing the wipe algorithm, the training video set was augmented by short clips with artificial wipes (between TV news shots) created with Adobe Premiere 4.2.1. Forty test clips, with varying parameters and styles of wipes, were created. The first shot of each clip contains mild object motion, and the second shot of each is a slow zoom; neither shot is motionless during the wipe. The combined length was 42 100 frames, or 23.5 minutes.

Sixty out of 62 wipes were detected, with 35 false alarms, when using the parameters $T_{corr} = 0.25$ (after the running mean was subtracted), $T_{dist} = 61000$, and 2 histogram bins per color dimension (for a total of 8 bins). The misses were mainly due to the adjacent shots having very similar histograms: one example is a wipe between two close-up views of a basketball play, having very similar histograms; except for its white boundary, the transition was barely visible to the eye. Most of the false alarms were due to close-up panning during a tennis segment, where the histograms changed wildly.

Using the same parameter set, the algorithm was tested on a feature-length (135.5 minute) movie containing 28 wipes of varying styles, along with significant motion and special effects. 14 of these wipes were detected, with a false alarm rate of 4.0 per minute. Again, most of the false alarms were due to rapid camera motion in action sequences. The wipes that were not detected were ones with very broad borders, within which the two images were blurred together; in this region, the histograms will not be linearly combined (some of the transitions were midway between a wipe and a dissolve). Naturally, the results will improve if the parameters are tailored to this particular stream, and different detection versus false alarm tradeoffs can be reached.

The wipe algorithm requires about 2.9 ms/frame of computation time; when added to the 18.1 ms/frame required to extract the DC+2AC frames, the algorithm runs at a rate of nearly 48 frames per second. If the dissolve algorithm is cascaded with the wipe algorithm, the overall processing speed is 46 frames per second.

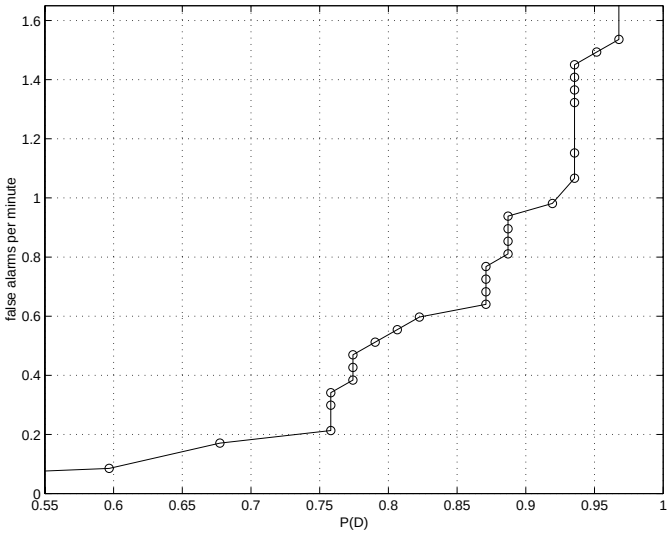

Fig. 13. ROC plot for our wipe detection algorithm over 23.5 minutes (42147 frames) of video, including some synthetic wipes. The false alarm rate is the number of false detections divided by the sequence length.

VII. Conclusions and Future Work

In this paper, we have presented a novel approach to the detection of gradual transitions in compressed video streams. The gradual transitions are grouped into two categories: those that affect every pixel simultaneously, but only by a small amount in each frame, and those that affect the pixels in some sequence, each pixel being changed abruptly. Dissolves and fades are members of the former class, and an algorithm to detect their presence is proposed based on their properties in a reduced-resolution, motion-compensated, frame space. Wipes and related computer effects are members of the latter class, and parallels are drawn with the dissolve case to develop a detection algorithm using histogram-space properties. Both algorithms have very good detection performance and run quickly enough to enable analysis of real-time streaming video (with ample headroom to allow further processing). Performance is improved, at least in the dissolve case, by employing a more sophisticated detector based on the same extracted statistics.

The algorithms described above can also be extended to the detection of partial-frame gradual transitions: the appearance of captions, graphic effects, or other spatially localized events. The correlations and histograms would be taken over the region of interest, and any time they meet the required thresholds, the same tests would be performed on the remainder of the frame, to ensure that it does *not* meet the thresholds (thereby confirming a partial frame transition). In the dissolve algorithm, a frame of correlations—in this case, taken pixelwise in color space—can even be constructed to visualize which regions best meet the dissolve model.

Comparison of our dissolve and wipe detection algorithms with others' methods *on the same streams* would be useful to the research community. Each algorithm's parameters and thresholds would need to be similarly optimized (or similarly not optimized) for the data set, which should contain a wide variety of video material. Further, a method is needed for characterizing thresholds' dependence on video sources. Use of our parametric detector may be helpful in factoring out the varied thresholding techniques employed by existing algorithms.

Histogram- and frame-space trajectories of video streams potentially hold significantly more information than the simple presence or absence of a line reveals. We are exploring exactly what information can be gleaned from joint study of these trajectories, possibly with the aid of additional cues, such as

audio, motion vectors, or other available data. By performing joint processing, with all available tracks (audio, video, and data), high-level content information can be heuristically inferred from "mid-level" statistics such as transition detections [19]. Segments of news broadcasts could be identified, voiceovers or interviews detected, and important replays from a sports event could be extracted, as just a few examples. With this high-level information, the stream can be inserted appropriately into a browseable or searchable database.

REFERENCES

[1] H. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," *Multimedia Systems*, vol. 1, pp. 10–28, 1993.

[2] J. Meng, Y. Juan, and S.-F. Chang, "Scene change detection in a MPEG compressed video sequence," in *Digital Video Compression: Algorithms and Technologies*, vol. 2419. Proceedings of the SPIE, February 1995, pp. 14–25.

[3] H.-C. H. Liu and G. L. Zick, "Scene decomposition of MPEG compressed video," in *Digital Video Compression: Algorithms and Technologies*, vol. 2419. Proceedings of the SPIE, February 1995, pp. 26–37.

[4] K. Shen and E. J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, October 1995, pp. 252–255.

[5] B.-L. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 533–544, December 1995.

[6] Y. Nakajima, K. Ujihara, and A. Yoneyama, "Universal scene change detection on MPEG-coded data domain," in *Visual Communications and Image Processing*, vol. 3024. Proceedings of the SPIE, February 1997, pp. 992–1003.

[7] M. Sugano, Y. Nakajima, H. Yanagihara, and A. Yoneyama, "A fast scene change detection on MPEG coding parameter domain," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, October 1998, pp. 888–892.

[8] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull, "Fade and dissolve detection in uncompressed and compressed video sequences," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, October 1999, pp. 299–303.

[9] H.-H. Yu and W. Wolf, "A multi-resolution video segmentation scheme for wipe transition identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, 1998, pp. 2965–2968.

[10] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying production effects," *Multimedia Systems*, vol. 7, no. 2, pp. 119–128, 1999.

[11] M. Wu, W. Wolf, and B. Liu, "An algorithm for wipe detection," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, October 1998, pp. 893–897.

[12] H. Kim, S.-J. Park, J. Lee, W. M. Kim, and S. M.-H. Song, "Processing of partial video data for detection of wipes," in *Storage and Retrieval for Image and Video Databases VII*, vol. 3656. Proceedings of the SPIE, January 1999, pp. 280–289.

[13] C. W. Ngo, T. C. Pong, and R. T. Chin, "Camera break detection by partitioning of 2D spatio-temporal images in MPEG domain," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, vol. 1, June 1999, pp. 750–755.

[14] S.-C. Pei and Y.-Z. Chou, "Efficient and effective wipe detection in MPEG compressed video based on the macroblock information," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, September 2000, pp. 953–956.

[15] M. S. Drew, Z.-N. Li, and X. Zhong, "Video dissolve and wipe detection via spatio-temporal images of chromatic histogram differences," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, September 2000, pp. 929–932.

[16] A. M. Alattar, "Wipe scene change detection for use with video compression algorithms and MPEG-7," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 43–51, 1998.

[17] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull, "Wipe scene change detection in video sequences," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, October 1999, pp. 294–298.

[18] H. V. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. Springer-Verlag, 1994.

[19] R. A. Joyce, "Content-based temporal processing of video," Ph.D. dissertation, Princeton University, Department of Electrical Engineering, November 2002. Available at http://www.atc-nycorp.com/papers/rjoycethesis.pdf.

Robert A. Joyce (S'95–M'02) received the B.S.E.E. degree in electrical engineering (with honors) from Cornell University, Ithaca, NY, in 1997, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, in 1999 and 2002, respectively. He is currently a Principal Scientist with ATC-NY, specialists in research and development for information assurance.

His research interests include image and video processing, network and media security, visualization and design, and human-computer interaction.

Bede Liu (S'55–M'62–F'72), Professor of Electrical Engineering at Princeton University, was born in China. He studied at the National Taiwan University (B.S.E.E. 1954) and the Polytechnic Institute of Brooklyn (D.E.E. 1960). Prior to joining Princeton University in 1962, he had been with Bell Laboratories, Allen B. DuMont Laboratory, and Western Electric Company. He has also been a visiting faculty member at several universities in the U.S. and abroad.

His current research interest lies mostly with multimedia technology, with particular emphasis on digital watermarking and video processing. He and his students have received the C&S Video Technology's best paper award for 1994, 1996, and 2003. His other IEEE awards include Centennial Medal (1984) and Millennium Medal (2000), Signal Processing Society's Technical Achievement Award (1985) and Society Award (2000), Circuit and Systems Society's Education Award (1988) and Mac Van Valkenburg Award (1997). He is a member of the National Academy of Engineering.

His was the President of the Circuit and Systems Society (1982), and the IEEE Division I Director (1984, 1985). He also served as the 1978 ISCAS Technical Program Chair the General Chair of the 1995 ICIP.