TEMPORAL SEGMENTATION OF VIDEO USING FRAME AND HISTOGRAM-SPACE

Robert A. Joyce and Bede Liu

Department of Electrical Engineering
Princeton University, Princeton, NJ 08544
{robjoyce, liu}@ee.princeton.edu

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. 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 well in detecting a wide variety of gradual transitions, and at a significant reduction in computation time when compared with full-frame methods.

1. 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 most of this 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 either affect all pixels simultaneously (a dissolve), or abruptly affect some evolving subset of the pixels (a wipe). Much work has been done on dissolve detection (of which fades are special cases), particularly with the use of reduced-resolution frames and motion vectors gathered directly from the compressed stream [1]–[7].

The detection of the latter class of gradual transitions, collectively known as "wipes", has not been as extensively studied. Wipe transitions are 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. More "artistic" wipes can be created by blurring or drop-shadowing the edge, or by adding further computer-generated effects during the transition; Figure 1 shows some examples. Wipes are often used in television news and sports coverage to denote stories or replays, as well as to set off scenes in movies (the *Star Wars* series, for example, uses wipes extensively). One application for the detection of wipes is the extraction of instant replays in sports video for automated summarization.

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.

One common method for wipe detection involves counting the number of pixels belonging to edges within the image; this statistic will monotonically change during a transition, from the old shot's quantity to the new shot's [8, 9]. 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 [10, 11]. A method has been proposed by which the statistical characteristics of wipe sequences are detected [12]. Most recently, an algorithm was proposed using the Hough transform on spatially-reduced frames to detect and characterize certain types of wipes [13]. With the prevalence of computer-generated wipes, sharp boundary assumptions 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 unaffected by any reasonable amount of producer-added effects (blurring, page-turning, shadows, etc.).

This paper begins a brief motivation for, and explanation of, the compressed-domain techniques used throughout; these are presented in Section 2. Section 3 describes a fast dissolve detection algorithm which eliminates some restrictions imposed by available dissolve detectors. A general method of wipe detection, which circumvents many of the limitations imposed by current algorithms, is presented in Section 4. Experimental results and conclusions are described in Sections 5 and 6, respectively.

2. COMPRESSED DOMAIN PROCESSING

One natural technique of compressed-domain analysis is reduced-resolution processing: using a subset of the block DCT coefficients to reconstruct thumbnail-sized images[1]. The effects of MPEG compression, video noise, and camera or object motion are far less significant at such reduced resolutions, facilitating analysis. Of particular interest is the construction of "DC frames," which are comprised of the lowest-order DCT coefficients of each block (and are therefore one sixty-fourth the size of the full frames). For intercoded compressed frames, the DC sequence can be easily estimated using available motion vectors [4]. 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.

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.

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

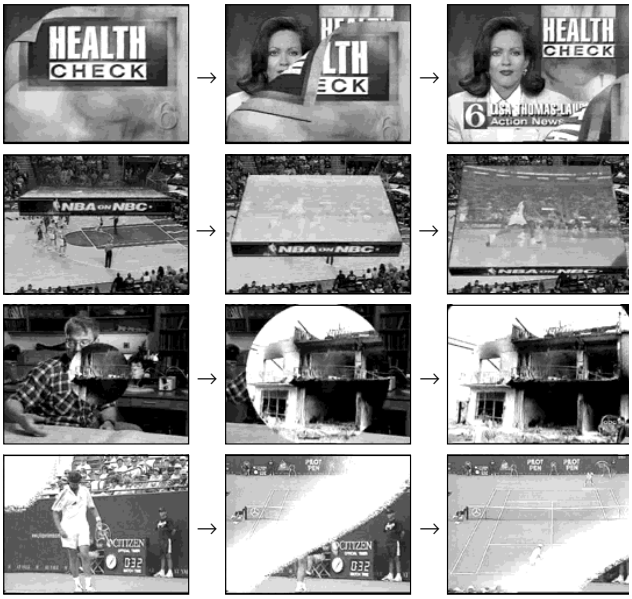

Fig. 1. Sample wipe sequences from network television, showing the wide variation possible.

In particular, DC DFD's require no computation at all, as they are just the lowest-order DCT coefficients of the residue frames.

3. DISSOLVE DETECTION

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. A dissolve from sequence g to sequence h, lasting from frame m to frame n, can therefore be described by

$$f_k(x, y) = \alpha_k h_k(x, y) + (1 - \alpha_k)\, g_k(x, y) \tag{1}$$

where the sequence α_k increases from $\alpha_m = 0$ at the beginning of the dissolve to $\alpha_n = 1$ 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 contain 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. Consider the trajectories formed by $f_b - f_a$ and $f_d - f_c$, where $m < a < b < n$ and $m < c < d < n$, and f_k is the vector formed by all the DC pixels of frame k. Substituting the model in (1) yields

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

during a dissolve. As α_k is an increasing sequence, we know $(\alpha_d - \alpha_c)\,(\alpha_b - \alpha_a)^{-1} > 0$. This condition is equivalent to the statement that, during a dissolve, the 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 2. Natural, non-dissolve motion in a stream generally does not have this characteristic.

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 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: $d^l_k = f_k - f_{k-l}$ and $d^l_{k+l} = f_{k+l} - f_k$. The correlation of frame-space vectors d^l_k and d^l_{k+l}, as a function of k, is then

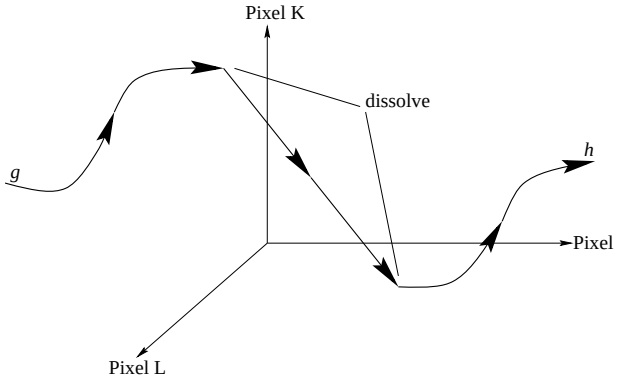

Fig. 2. Three-dimensional representation of a video sequence f_k in frame-space during a dissolve.

$$\rho_k = \frac{\left\langle d^l_{k+l}, d^l_k \right\rangle}{\sqrt{\left\| d^l_{k+l} \right\|^2 \left\| d^l_k \right\|^2}}. \tag{3}$$

A 'straight' triplet of frames is declared if the correlation is greater than a threshold T_{corr}. In order to declare a dissolve, we require that the threshold be met for every triplet in some sequence of frames, say from m to n. We also require that

$$\|f_n - f_m\| \geq T_{dist}. \tag{4}$$

The length condition is necessary because small changes (eg, in frame brightness) can lead to the correlation condition being met for an isolated triplet or two. Both threshold tests can be done sequentially, with no knowledge of future frames beyond $k + l$. If we use DFD's, instead of the true frame differences, simple object or camera motion can be compensated for; a plot of this DFD-based correlation sequence is shown in Figure 3.

Values for T_{corr} and T_{dist} should be set based on the desired false alarm rate or detection accuracy. 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 value of the past N ρ_k values is subtracted before the T_{corr} comparison is made. Specific experimental results are presented in Section 5.

4. WIPE DETECTION

As in the dissolve case, assume we have a wipe from frame m to frame n, a transition between sequence g and sequence h. While simple wipes can be modeled with frame-space equations, complex computer-generated wipes do not admit a single, general model. Instead, we focus on which pixels belong to g and which to h in each frame. The wipe is then characterized by an indicator sequence $I_k(x, y)$, which is 0 for all x and y at the beginning of the wipe, and 1 for all at the end. $\|I_k\|$ will increase from 0 to

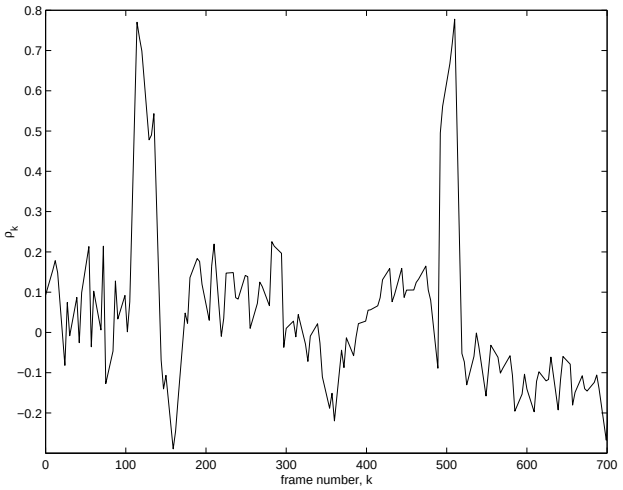

Fig. 3. The sequence ρ_k for a segment of documentary video, using DFD's and $l = 3$; dissolves occur during frames 115–140 and 492–516.

the number of pixels in the (reduced-resolution) image, as k goes from m to n.

One representation of a video sequence that allows us to examine the $\|I_k\|$ sequence, without imposing a specific wipe pattern, is the histogram. We denote the p-th bin of frame f_k's histogram as $F_k(p)$; during a wipe, we have

$$F_k(p) = \left(\frac{\|I_k\| + E_{G,k}(p)}{N}\right) G_k(p) + \left(1 - \frac{\|I_k\| + E_{H,k}(p)}{N}\right) H_k(p), \quad (5)$$

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

$$\beta_k = \frac{\|I_k\| + E_{G,k}(p)}{N} \quad (6)$$

will be increasing in k from 0 to 1, and

$$1 - \frac{\|I_k\| + E_{H,k}(p)}{N} \approx 1 - \beta_k. \quad (7)$$

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

We compute the correlation sequentially, from pairs of histogram differences:

$$\rho_{hist,k} = \frac{\left\langle D_{k+l}^l, D_k^l \right\rangle}{\sqrt{\left\|D_{k+l}^l\right\|^2 \left\|D_k^l\right\|^2}}. \quad (8)$$

This value is compared to a threshold, and the value of (4) is computed to determine the length of the candidate wipe; a wipe transition is declared if both thresholds are met.

Equation (5) 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. For this reason, we chose to use on the order of 2 to 4 histogram bins per color dimension, and perform the histograms on DC+2AC frames. While characterizing these data-dependent errors and their effects on ρ_{hist} remain open problems, their effects on ρ_{hist} can be reduced by low-pass filtering the resulting correlation sequence (the assumption here being that the errors in $\rho_{hist,k}$ are approximately independent in k). Low-pass filtering also helps alleviate the time-varying histogram distortions that MPEG compression and video noise can introduce. The quantization issues can also be ameliorated by using the inner product, instead of the correlation, between histogram vectors; this benefit comes at the cost of additional false alarms in non-wipe segments with large histogram changes.

One issue has not yet been addressed: can natural motion in video cause a linear trajectory in histogram-space? 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 the movement. 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.

5. EXPERIMENTAL RESULTS

Each algorithm was tested with "natural" television or 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.

5.1. Dissolve Detection

A thirteen minute set of video was used as a "training" set, on which parameter values were selected. The training set's 23700 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 human viewer might not notice the transition at first glance. 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) and $T_{dist} = 83000$.

With these values, 57 out of the 59 dissolves were properly detected, with 9 false alarms (a rate of one per 2633 frames). Different thresholds can be selected to yield different detection probability / false alarm tradeoffs.

When run on a larger, 29 minute (52700 frame) test set, the parameter values listed above yielded a detection rate of 132 out of 156, with 43 false alarms. As this was not the training set for our algorithm, these values are not optimal; adjusting the parameters can give higher detection rates while maintaining this number of false alarms. Further tweaking of the parameter values can also yield much higher detection rates at the expense of greater false alarm numbers; for instance, a 148/156 detection rate is possible if we allow 49 false alarms. If necessary, further processing can be done to cut the number of false alarms (for example, requiring the frame space vectors have at least a certain L2 length during a dissolve, or utilizing other statistical properties dissolve transitions must have).

Including the overhead due to DC frame extraction, our algorithm processed video at about 170 frames per second. In fact, nearly 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.

5.2. Wipe Detection

In testing the wipe algorithm, a broadcast video set of news and sports 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, 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 42100 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). 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.

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.

6. CONCLUSIONS

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 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). Each could be further improved by additional pruning of false alarms, as well as by more sophisticated adaptive threshold techniques. In many applications, differentiating one type of wipe from another can yield semantic information; such automated classification of wipe types would be a useful future addition.

7. REFERENCES

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

[2] M. Sugano, Y. Nakajima, H. Yanagihara, and A. Yoneyama, "A fast scene change detection on MPEG coding parameter domain," in *Proc. IEEE ICIP*, 1998, vol. 1, pp. 888–892.

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

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

[5] Y. Nakajima, K. Ujihara, and A. Yoneyama, "Universal scene change detection on MPEG-coded data domain," in *Visual Comm. and Image Proc*. Proc. SPIE, 1997, vol. 3024, pp. 992–1003.

[6] K. Shen and E. J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," in *Proc. IEEE ICIP*, 1995, vol. 2, pp. 252–255.

[7] W. Fernando, C. Canagarajah, and D. R. Bull, "Fade and dissolve detection in uncompressed and compressed video sequences," in *Proc. IEEE ICIP*, 1999.

[8] 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.

[9] H.-H. Yu and W. Wolf, "A multi-resolution video segmentation scheme for wipe transition identification," in *Proc. IEEE ICASSP*, 1998, vol. 5, pp. 2965–2968.

[10] M. Wu, W. Wolf, and B. Liu, "An algorithm for wipe detection," in *Proc. IEEE ICIP*, 1998, vol. 1, pp. 893–897.

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

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

[13] W. Fernando, C. Canagarajah, and D. R. Bull, "Wipe scene change detection in video sequences," in *Proc. IEEE ICIP*, 1999.