SCENE BOUNDARY DETECTION VIA VIDEO SELF-SIMILARITY ANALYSIS

Matthew Cooper and Jonathan Foote

FX Palo Alto Laboratory
3400 Hillview Ave. Bldg. 4
Palo Alto, CA 94304
http://www.fxpal.xerox.com
{cooper, foote}@pal.xerox.com

ABSTRACT

In this paper, we present a novel framework for analyzing video using self-similarity. Video scenes are located by analyzing inter-frame similarity matrices. The approach is flexible to the choice of similarity measure and is robust and data-independent because the data is used to model itself. We present the approach and its application to scene boundary detection. This is shown to dramatically outperform a conventional scene-boundary detector that uses a histogram-based measure of frame difference.

1. INTRODUCTION

Video segmentation is an increasingly important problem. Numerous video retrieval and management tasks rely on accurate segmentation of scene boundaries, for example the commercial video-logging system developed by Virage Inc. [1] In this paper, we describe a novel video analysis method based on signal self-similarity. This approach facilitates scene boundary detection and other video characterizations. A particular benefit of this work is that it effectively uses the signal to model itself, making minimal assumptions about the nature or genre of the target video. We present scene detection performance of several videos from different genres, judged against a baseline hand segmentation and a standard segmentation algorithm based on the histogram difference. For each video, the self-similarity algorithm outperforms the standard segmentation technique.

2. SIMILARITY ANALYSIS

We detect scene boundaries by considering the self-similarity of the video across time. For each instant in the video, the self-similarity for past and future regions is computed, as well as the cross-similarity between the past and future. A significantly novel point in the video, i. e. a scene boundary, will have high self-similarity in the past and future and low cross-similarity between them.

[1]www.virage.com

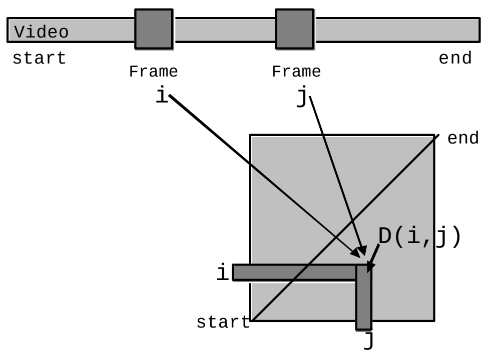


Fig. 1. Diagram of the similarity matrix embedding.

Video frames are parameterized and are then embedded in a 2-dimensional representation [1]. Figure 1 shows how the distance measure is embedded. A measure D of the (dis)similarity between frame parameters $\vec{v}_i$ and $\vec{v}_j$ is calculated for every pair of video frames i and j. The matrix **S** contains the similarity measure calculated for all frame combinations i and j such that the $(i,j)^{th}$ element of **S** is $D(\vec{v}_i, \vec{v}_j)$. Time, or frame index, runs along both axes as well as the diagonal. In general, **S** will have maximum values on the diagonal (because every frame will be maximally similar to itself); furthermore if D is symmetric then **S** will be symmetric as well.

S can be visualized as a square image such that each element (i,j) is given a gray scale value proportional to the similarity measure $D(i,j)$, and scaled such that the maximum value is given the maximum brightness. These visualizations let us clearly identify structure within a video. Regions of high similarity, such as a long sequence of identical frames, appear as bright squares on the diagonal. Repeated sequences are visible as diagonal stripes or checkerboards, offset from the main diagonal by the repetition time. For example, the similarity matrix of panel (a) in Figure 2 is from a golf instruction video, in which shots of golf swings are repeated, separated by fades. Shots are apparent as bright squares on the diagonal; similar shots can be seen in the off-diagonal rectangles. The repetitive structure of the video is apparent in the closeup of panel (b). From the manual segmentation of the golf video, there is a scene break at frame 7317. Subsequent frames show a golf shot followed by a fade. The shot is repeated at frame 7564. In the closeup of the similarity matrix, the shot frames, the fade from frames 7500-7564, and the repeated shot are all visible. Finally, there is a second fade and a new scene which begins at frame 7797. Each of these boundaries are clearly exhibited in the similarity matrix. The similarity between the repeated segments is also evident from the diagonal lines offset from the main diagonal by the repetition time. An advantage of this approach is that the exhibited structure is derived entirely from the current video rather than from predefined models or parameterizations. There are minimal prior assumptions regarding the video content, which is an essential requirement for numerous applications.

3. IMPLEMENTATION

3.1. Computing the Similarity Matrix

Each frame is parameterized by the simple measure of converting each pixel to a greyscale representation by summing RGB values. A 30-bin intensity histogram is then computed, which serves to characterize each frame. These histogram features are compared using the (nonlinear) cosine distance measure

$$D(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{||\vec{v}_i||\,||\vec{v}_j||} \quad . \tag{1}$$

3.2. Scene Segmentation Via Kernel Correlation

Consider a simple 10-frame video consisting of two different "scenes", each with 5 identical frames. When visualized by coloring regions brighter according to similarity, $\mathbf{S}$ will look like a 2×2 checkerboard. White 5×5 squares on the diagonal correspond to the two scenes, which have high self-similarity; black 5×5 squares on the off-diagonal are regions of low cross-similarity. Finding the scene boundary transition is as simple as finding the center of the checkerboard. This can be done using a classic matched filter: correlating $\mathbf{S}$ with a kernel that itself looks like checkerboard [2]. We will call this class checkerboard kernels. Perhaps the simplest is the 2×2 unit kernel

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad . \tag{2}$$

For automatic scene segmentation, we correlate the Gaussian checkerboard kernel (panel (c) of Figure 2) along the diagonal of the similarity matrix $\mathbf{S}$. The result is a one-dimensional function of time (frame index), as shown in the bottom of Figure 3. Intuitively, the correlation emphasizes regions with strong self-similarity while penalizing regions with significant cross-similarity. As illustrated in Figure 3, peaks in the correlation indicate locally novel points in the video, which we label as scene boundaries.

Algorithm 1 *Self-Similarity Based Scene Change Localization*

```
1. Compute cosine similarity matrix
    (a) Transform every tenth frame from RGB to intensity (greyscale) image
    (b) Compute histogram of intensity image
2. Compute correlation along diagonal of similarity matrix with Gaussian checkerboard kernel (Figure 2(c))
3. Locate peaks via analysis of the first and second differences of the output signal
4. Label peaks as scene boundaries
```

3.3. Enhanced DCT Features

As a further enhancement of our algorithm, we investigated advanced features based on low-order discrete cosine transform (DCT) coefficients. Instead of intensity histograms, this implementation processes every 10^{th} frame and transforms the individual RGB frames into the Ohta color space according to:

$$\begin{bmatrix} o_1 \\ o_2 \\ o_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad . \tag{3}$$

In this color space, the three channels are approximately decorrelated [3]. The DCT of each channel is computed and a feature vector is formed by concatenating the resulting low frequency coefficients of the three channels. These feature vectors are compared using the cosine distance measure and Algorithm 1 is used to compute the scene boundaries.

4. EXPERIMENTS

Algorithm 1 was compared against the popular approach based on thresholding the Euclidean distance between inter-frame intensity histograms (e.g. [4]). The histogram-based algorithm is summarized below.

Algorithm 2 *Histogram Based Scene Change Localization*

```
1. Compute Euclidean inter-frame histogram distance
```

   (a) *Transform every tenth frame from RGB to intensity (grayscale) image*

   (b) *Compute histogram of intensity image*

   (c) *Compute Euclidean distance between current histogram and previous (ten frames previous) histogram*

2. *Locate peaks via analysis of the first and second differences of the inter-frame distances.*

3. *Label peaks exceeding threshold as scene boundaries*

To assess performance, two figures of merit are computed to construct receiver operating characteristics (as in [4]):

$$\text{Recall} = \frac{\text{Correct Scene Changes}}{\text{Correct + Missed Scene Changes}} \tag{4}$$

$$\text{Precision} = \frac{\text{Correct Scene Changes}}{\text{Correct + False Scene Changes}} . \tag{5}$$

To control the behavior of Algorithm 1, we vary the size of the kernel used. The kernel's size is proportional to the expected average segment size within the video. To vary the recall and precision, we segment the videos with the size of the kernel ranging from 21 to 71. To explore the high-precision and low-recall performance of Algorithm 1, we used thresholds for the extracted peaks from the correlation with the 71 by 71 kernel. To vary the performance of Algorithm 2, the threshold for labeling scene boundaries is varied.

Automatic scene segmentation experiments were performed using 10000 frames from each of three videos: a golf instructional video, a cartoon video ("Knighty Knight," also analyzed in [4]), and a music variety show ("The Live Music Show" from [5]). Manual segmentations were performed for comparison against the two automatic methods. The manual segmentations produced a total of 181 scene boundaries including both cuts and fades. The performance of the two approaches is summarized by the receiver operating characteristic of Figure 4. Figure 4 shows three curves: the solid curve for Algorithm 1 with DCT features, the dashed curve with crosses for Algorithm 1 with histogram features, and the dashed curve with squares for Algorithm 2. At every level of recall, both self-similarity approaches outperform the histogram-thresholding approach, and the self-similarity approach using the DCT features performs best of the three.

5. RELATED WORK

Space does not permit more than a brief overview of the large number of video segmentation algorithms; for a good review see [4]. The 2-dimensional distance matrix has been used to visualize text [6] and time series [7] but no analysis or segmentation was performed. Similar techniques have been used to analyze and segment audio by one of the authors [2].

6. CONCLUSION

We have presented a novel video segmentation algorithm and demonstrated its performance. Better features based on color and shape further improved the segmentation. Though on first inspection it seems that Algorithm 1 requires $\mathbf{O}(n^2)$ computations, this is not the case in practice. For segmentation, there is no reason to calculate similarity matrix values further from the diagonal than the extent of the kernel, which is typically a small constant. Thus the algorithm can be computed in $\mathbf{O}(n)$ computations, and can be computed on-the-fly as long as a small frame buffer (the size of the kernel) is available. Additionally, because both the similarity matrix and the kernel will typically be symmetric, many computations are redundant: only one half of the matrix and the kernel need be computed and correlated. Thus the algorithm is quite competitive with seemingly simpler approaches such as histogram differences.

7. REFERENCES

[1] J. Foote. Visualizing Music and Audio using Self-Similarity. *Proc. ACM Multimedia 99*, Orlando, FL, 1999.

[2] J. Foote. Automatic Audio Segmentation using a Measure of Audio Novelty. *Proc. of IEEE International Conference on Multimedia and Expo*, vol. I, pp. 452-455, 2000.

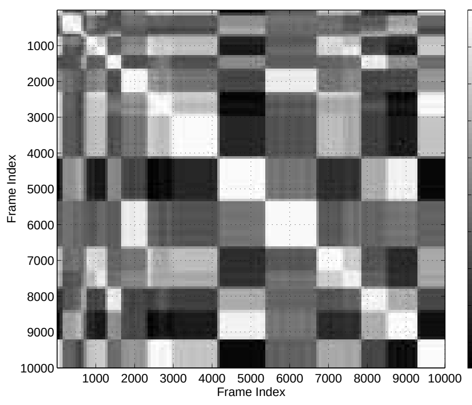
[3] Y-I Ohta, T. Kanade, and T. Sakai. Color Information for Region Segmentation. *Comp. Graphics & Image Processing*, **13**:222-241, 1980.

[4] J. Boreczky and L. Rowe. Comparison of video shot boundary detection techniques. *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1996.

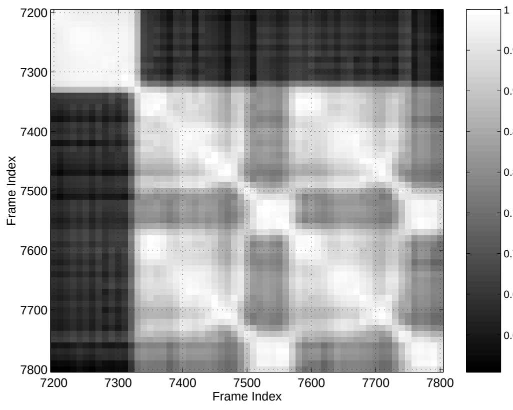
[5] MPEG Requirements Group. Description of MPEG-7 Content Set, Doc. ISO/MPEG N2467, MPEG Atlantic City Meeting, October 1998.

[6] K. Church and J. Helfman. Dotplot: A Program for exploring Self-Similarity in Millions of Lines of Text and Code *J. American Statistical Assoc.*, **2**(2):153–174, 1993.

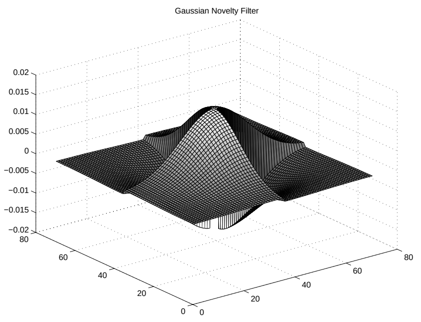
[7] J. Eckman, et al. Recurrence Plots of Dynamical Systems. *Europhys. Lett.* **4**(973), November 1987.



(a)



(b)



(c)

Fig. 2. Panel (a) shows a similarity matrix computed per Algorithm 1. Panel (b) shows a closeup of Panel (b) revealing the structure of repeated scenes within a single video segment. Panel (c) shows the Gaussian checkerboard kernel used to identify scene boundaries.

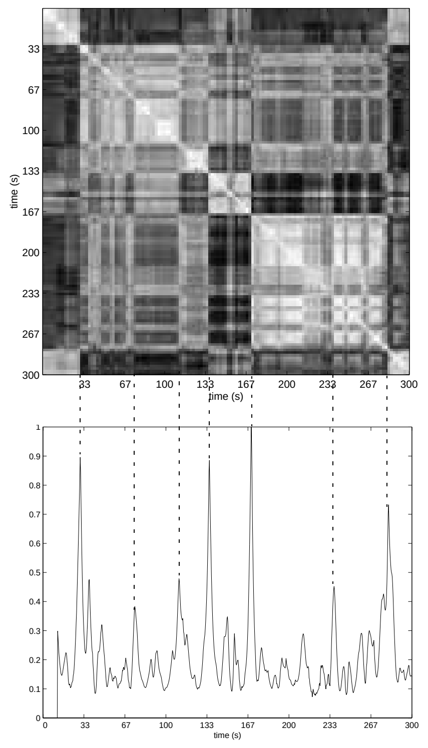


Fig. 3. The top panel shows a similarity matrix computed from the Home Video in [5] using DCT features. The bottom panel shows the kernel correlation (Algorithm 1). The dashed lines indicate correspondences of peaks in the kernel correlation to boundaries in the similarity matrix.

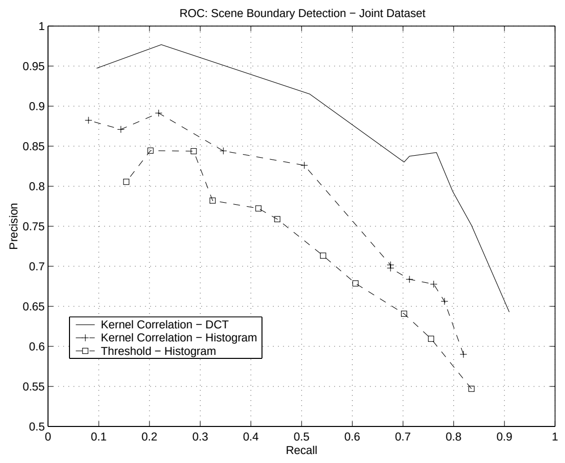


Fig. 4. The receiver operating characteristic curves for the automatic scene segmentation of the three videos. The dashed curves represent results using the histogram features (Algorithms 1, the line with crosses, and 2, the line with squares) and the solid curve represents results for the kernel correlation technique with DCT features(Algorithm 1).