An SVM Framework for Genre-Independent Scene Change Detection

Naveen Goela, Kevin Wilson, Feng Niu, Ajay Divakaran

TR2007-033 August 2008

Abstract

We present a novel genre-independent SVM framework for detecting scene changes in broadcast video. Our framework works on content from a diverse range of genres by allowing sets of features, extracted from both audio and video streams, to be combined and compared automatically without the use of explicit thresholds. For ground truth, we use hand-labeled video scene boundaries from a wide variety of broadcast genres to generate positive and negative samples for the SVM. Our experiments include high-and low-level audio features such as semantic histograms and distances between Gaussian models, as well as video features such as shot cut positions. We evaluate the importance of these measures in a structured framework, with performance comparisons oriented via ROC curves. We achieve over 70% detection rate for 10% false positive rate on our corpus of over 7.5 hours of data collected from news, talk shows, sitcoms, dramas, music videos, and how-to-shows.

IEEE International Conference on Multimedia and Expo (ICME)

AN SVM FRAMEWORK FOR GENRE-INDEPENDENT SCENE CHANGE DETECTION

Naveen Goela, Kevin Wilson, Feng Niu, Ajay Divakaran

Mitsubishi Electric Research Laboratories
Cambridge, MA 02139

Isao Otsuka

Mitsubishi Electric Corporation
Kyoto, Japan

ABSTRACT

We present a novel genre-independent SVM framework for detecting scene changes in broadcast video. Our framework works on content from a diverse range of genres by allowing sets of features, extracted from both audio and video streams, to be combined and compared automatically without the use of explicit thresholds. For ground truth, we use hand-labeled video scene boundaries from a wide variety of broadcast genres to generate positive and negative samples for the SVM. Our experiments include high- and low-level audio features such as semantic histograms and distances between Gaussian models, as well as video features such as shot cut positions. We evaluate the importance of these measures in a structured framework, with performance comparisons obtained via ROC curves. We achieve over 70% detection rate for 10% false positive rate on our corpus of over 7.5 hours of data collected from news, talk shows, sitcoms, dramas, music videos, and how-to shows.

1. INTRODUCTION

In broadcast video content, scene changes provide structure that can be useful for understanding, organizing, and browsing the content. Our primary motivation for studying scene change detection is to improve the video-browsing capabilities of consumer electronics devices to allow users to more quickly and effectively manage their content. Thus, in this paper, the term “scene change” refers to a semantically meaningful change that may or may not have an obvious manifestation in the video and/or audio. Furthermore, we choose a definition of “scene change” that results in an average of one scene change every few minutes, which we believe is a useful granularity for content browsing.

Our work depends on hand-labeled ground truth, so the operational definition of a scene change depends on the opinion of the human who located scene changes in our video corpus. In sitcoms and dramas, scene changes typically correspond to changes in filming location or to the entrance of a significant new character. For news, scene changes correspond to boundaries between news stories. For talk shows, scene changes correspond to changes from one guest or skit to another. Similar judgements are made for other genres. In all genres, the transitions between program content and commercials and the transitions from one commercial to the next are also considered scene changes.

Detecting these scene changes using simple audio and video features is challenging because scene changes for different genres, and even scene changes within one genre, do not necessarily have any obvious similarities. Shot changes, the change from one continuous sequence filmed by a single camera to another such sequence, is a much-studied problem [1] that can largely be solved using simple, low-level video features. We will use such a shot-change detector as a component in our scene change detector, but it is important to note that our semantic scene change detection task is a distinct and more challenging problem.

Detecting video scene changes over scripted and unscripted content has been explored in the past using image differences and visual motion vectors, as well as differences in audio distributions [2, 3, 4, 5]. Usually, after a feature extraction step, a comparison with a set threshold is required. In other cases, research has focused on developing audio and visual models but without a framework to compare the effectiveness of features easily. Our work provides a more thorough performance analysis, and in addition, our testing and training are done on a much more diverse range of content than any of [2, 3, 4, 5]. ([2] used only news content. [3] used sitcoms and short scenes from a few movies. [4, 5] used an hour-long segment of a single movie.)

There has also been work on the shot change (as opposed to scene change) problem that parallels our work. [6] proposes an HMM model for genre-independent shot change detection using audio and video features that does not rely on hand-tuned thresholds; however, they differ from our approach because they use a generative, as opposed to discriminative, model. In addition, our paper investigates the relative performance of several different features, both individually and in combination. [7] uses an SVM framework for shot change detection. It focuses more on video features, and it trains and tests on a much smaller corpus than that of our current work.

In summary, detecting semantic scene changes is challenging due to factors including (1) the lack of training data; (2) difficulty in defining scene changes across diverse genres; (3) absence of a systematic method to characterize and compare performance of different features; (4) difficulty in determining thresholds in hand-tuned systems. We address issues (1) and (2) by hand-labeling several hours of data comprising content from several genres. We address issues (3) and (4) by using an SVM framework which automatically determines decision boundaries during training and which can accommodate a variety of audio- or video-derived features.

2. FEATURE DESCRIPTION

We use a discriminative Gausssian-kernel SVM framework [8] for detecting video scene changes. During the training phase, the classifier requires input vectors for scene changes as well as non-scene changes, and constructs the optimal (possibly non-linear) decision boundary separating the vectors in the input space. Our goal is to find good features for distinguishing scene boundaries from non-scene boundaries in diverse video content. Because of our finite amount of training and test data, we also require that our input vectors to the SVM be relatively low-dimensional. Finally, we base our choice of features on the fact that certain feature streams are readily available, computationally efficient, and amenable to our product platform.

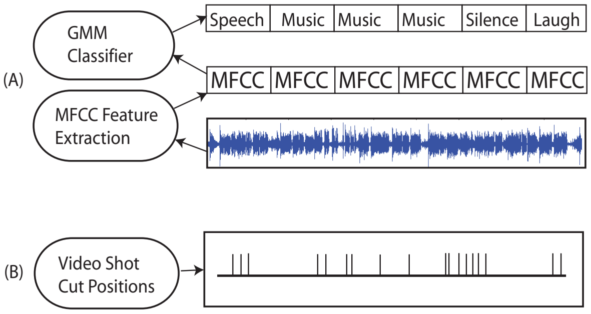

Fig. 1. Feature streams: (A) Low level MFCC spectral coefficients, high level semantic labels. (B) Video shot cut frame positions.

Video and audio feature streams are shown in Fig. 1. We rely mostly on audio because visual features are computationally more expensive. For audio, we start with an MPEG video source and extract a single-channel audio stream at 44.1 KHz. We compute 12 Mel-frequency cepstral coefficients (MFCCs) over 20 ms frames. Based on the low-level MFCC features, we classify each second of audio into one of four semantic classes: $\{$music, speech, laughter, silence$\}$ using maximum likelihood estimation over Gaussian Mixture Models (GMMs) [9]. The mixture models for each semantic class were estimated from separate data. These semantic labels help us to detect, for example, the brief snippets of music that accompany scene changes in some content or the laughter that often comes at the end of a scene in a sitcom.

To show that our framework supports diverse feature types, we also extract video frames (at 29.97 fps) and record the frame number of all shot cuts in the video. We use a basic hard shot cut detector [1]. Prior work on video scene changes has used motion vectors and image differences at the pixel level, which we may incorporate into our work in the future.

Using the above audio and video features, we define an SVM input vector X_i for scene(+) and non-scene(-) boundaries as follows: $X_i = \{x_1, x_2, x_3, \ldots x_{11}, x_{12}\}$. In our experiments, our best-performing feature vector contained 12 dimensions, but we experimented with various features and subsets of varying dimensionality.

The input vectors X_i describe the local information about a particular time position t (in seconds) within the video. We compute an X_i at the hand-labeled time positions for scenes and (randomly generated) non-scenes. The first 9 components of X_i are histograms of semantic labels as explored in recent work [9], the next two components represent the difference between the audio distribution before and after a particular time t, and the final component is based on video shot cut counts. The components are defined as follows:

1. **Pre-histogram:** variables x_1, x_2, x_3
   The pre-histogram tallies the number of semantic labels in the set $\{$music, speech, laughter, silence$\}$ within a window of $[t - W_L, t]$, where W_L is a chosen window size. The histogram is normalized to sum to 1. We discard one dimension from the 4D histogram because it is fully determined by the remaining three histogram values.
2. **Mid-histogram:** variables x_4, x_5, x_6
   The mid-histogram is similar to the pre-histogram and tallies semantic labels within $[t - \frac{W_L}{2}, t + \frac{W_L}{2}]$.
3. **Post-histogram:** variables x_7, x_8, x_9
   The post-histogram tallies labels within $[t, t + W_L]$.
4. **Bhattacharyya Shape+Distance:** variables x_{10}, x_{11}
   We calculate the Bhattacharyya shape and Mahalanobis distance between single Gaussian models estimated from the low level MFCC coefficients for region $[t - W_L, t]$ and region $[t, t + W_L]$.

$$D_{shape} = \frac{1}{2} \ln \frac{|\frac{C_i + C_j}{2}|}{|C_i|^{\frac{1}{2}} |C_j|^{\frac{1}{2}}} \tag{1}$$

$$D_{mahal} = \frac{1}{8} (\mu_i - \mu_j)^T (\frac{C_i + C_j}{2})^{-1} (\mu_i - \mu_j) \tag{2}$$

   The covariance matrices C_i and C_j and the means μ_i and μ_j represent the (diagonal) covariance and mean of the MFCC vectors before and after a time position t.

   Bhattacharyya shape and Mahalanobis distance are sensitive to changes in the distributions of the MFCCs, so these features provide much lower-level cues about changes. For example, a scene change accompanied by a change from a male speaker to a female speaker would generate a large MFCC Mahalanobis distance even though the semantic histograms would show that both scenes contained primarily speech. (Our speech class is trained on both male and female speech.)
5. **Average Shot Count:** variable x_{12}
   The final component is twice the average number of shot cuts present in the video within a window $[t - W_L, t + W_L]$.

Since we use a kernel-based SVM with a smoothing bandwidth that is equal along all dimensions, we ensure that all of the variables in X_i have approximately the same variance. For histograms, we also experimented with replacing the semantic histograms with a related feature equal to the difference between the pre- and post-histograms. This histogram difference is lower-dimensional and can potentially provide better performance, but we achieved slightly better performance with the full histogram features. For average shot counts, we originally used the number of shot changes present within a few seconds of a time t, but due to noise and misalignment, this did not work as well. Instead, we averaged over a larger window length. After experimenting with different window sizes, we found that an optimal window length of $W_L = 14$ seconds provided enough data to estimate the Bhattacharyya distances and semantic histograms.

3. SVM CLASSIFIER FRAMEWORK

A support vector machine (SVM) [8] is a supervised learning algorithm that attempts to find the maximum margin hyperplane separating two classes of data. Given data points $\{X_0, X_1, \ldots X_N\}$ and class labels $\{y_0, y_1 \ldots y_N\}, y_i \in \{-1, 1\}$, the SVM constructs a decision boundary for the two classes that generalizes well to future data. For this reason, the SVM has been used as a robust tool for classification in complex, noisy domains. In our case, the two classes are scene(+) versus non-scene(-) boundaries. The data points X_i are up to 12D vectors as described in Section 2. We expect that an SVM using our 12D feature input vector will be easily implementable on our product platform.

One advantage of the SVM framework is that the data **X** can be transformed to a higher dimensional *feature space* via a kernel function. Data may be linearly separable in this space by a hyperplane

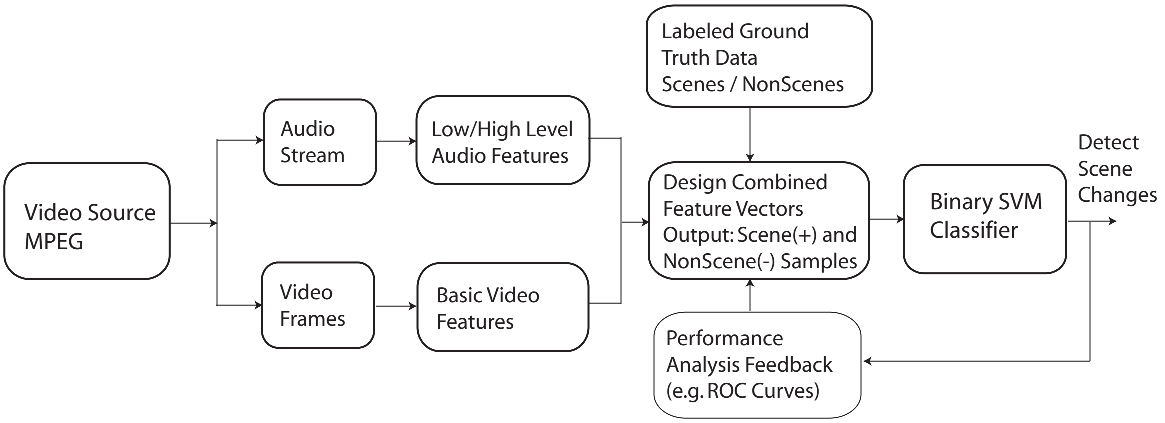
Labeled Ground Truth Data Scenes / NonScenes
Audio Stream
Low/High Level Audio Features
Video Source MPEG
Design Combined Feature Vectors Output: Scene(+) and NonScene(-) Samples
Binary SVM Classifier
Detect Scene Changes
Video Frames
Basic Video Features
Performance Analysis Feedback (e.g. ROC Curves)

Fig. 2. SVM Classifier Framework.

that is actually a non-linear boundary in the original input space. In our implementation, we found a radial basis kernel worked well:

$$K(X_i, X_j) = e^{-\gamma D^2(X_i, X_j)} \tag{3}$$

We use L_2 distance although various distance functions are possible. We fixed the value of the kernel bandwidth $\gamma = 2.0$, but could adjust this value for less smoothing if more training data were available. With limited training samples, we would like a smooth boundary to account for noise. Noise is introduced in various ways such as inaccuracies in the audio or video feature streams (misclassified semantic labels, missed/false shot cuts, alignment of streams), and in incorrect hand-labeled boundaries.

We used over 7.5 hours of diverse content to generate training and test samples for the classifier. This amounted to 530 scene(+) sample points. For non-scene(-) samples, we automatically generated twice as many random non-scene boundaries chosen at time positions outside a specific W_L of scene(+) positions. Due to the difficulty in collecting a large amount of scene(+) boundaries, most previous research has not focused on supervised learning for scene separation. However, the advantage of casting the scene change detection problem as a classification problem is that we eliminate the need for explicit thresholds for variables since the decision boundaries are tuned by the SVM. Furthermore, we are able to compare various combinations of features quickly, based on their performance against ground truth. The SVM provides a unifying framework for jointly modeling separate features. This allows us to add features as necessary to accommodate diverse video content. Even if we do not use a supervised approach in the end, a supervised learning algorithm can indicate which features work well for our application.

Fig. 2 shows a block diagram of the overall SVM framework. We first split an MPEG video source into audio and video streams, and extract low/high level audio features and video features. Using ground truth labeled data, we design input vectors for scene(+) and non-scene(-) samples using a combination of features. The final step is to detect scene changes using the binary SVM and characterize performance via ROC curves. The performance results can be used as feedback to design better input vectors based on available feature streams.

4. EXPERIMENTS

In our experiments, we tested (1) the ability of our framework to compare different sets of features in terms of ROC performance; and (2) the ability of our framework to detect scene changes over a wide variety of broadcast genres. We used the OSU SVM Toolbox (`http://sourceforge.net/projects/svm/`), and results are based on 5-fold cross-validation.

In order to generate ROC curves, we varied the SVM cost penalty for misclassifying a scene(+) boundary versus misclassifying a non-scene(-) boundary. Based on the cost ratio, the SVM produces a different separating hyperplane, yielding a performance result with different true and false positive rates. The true positive rate is the percentage of scene changes correctly detected by our system. The false positive rate is the percentage of non-scene boundaries that were classified incorrectly as scene boundaries. Ideally, we wish to achieve high true positive rates and low false positive rates. In classifying a new video piece, it may be necessary to achieve a false positive rate of 5% and as high a true positive rate as possible. In other cases, we can lower the false positive rate by other means such as pre-processing, only choosing candidate locations to test for scene changes.

Using our 12D input vectors described in Section 2 (with concatenated histograms, Bhattacharyya measures, and shot counts) to describe scene vs. non-scene boundaries, our algorithm scores 62% for a true positive percentage, corresponding to a false positive percentage of 5%. Allowing a higher false positive percentage of 20%, the algorithm achieves an 83% detection rate. The best result is shown on Panel A of Fig. 3. Since we used primarily audio features in finding video scene changes, we believe this is a strong result for a genre-independent system. We also used a broad, semantically meaningful definition for scene change. In generating the ROC curves, we averaged results from 10 runs, each time using a different set of randomly generated non-scene boundaries.

We also compared different sets of features (Panels A-C in Fig. 3). In Panel A, we show that the inclusion of shot counts improves scene change detection significantly. Since shot cut count is a basic video feature, we believe that including computationally expensive video motion vectors or image differences would result in higher performance. In Panel B, we show that the Bhattacharyya measures and pre/mid/post histograms perform much better together than they

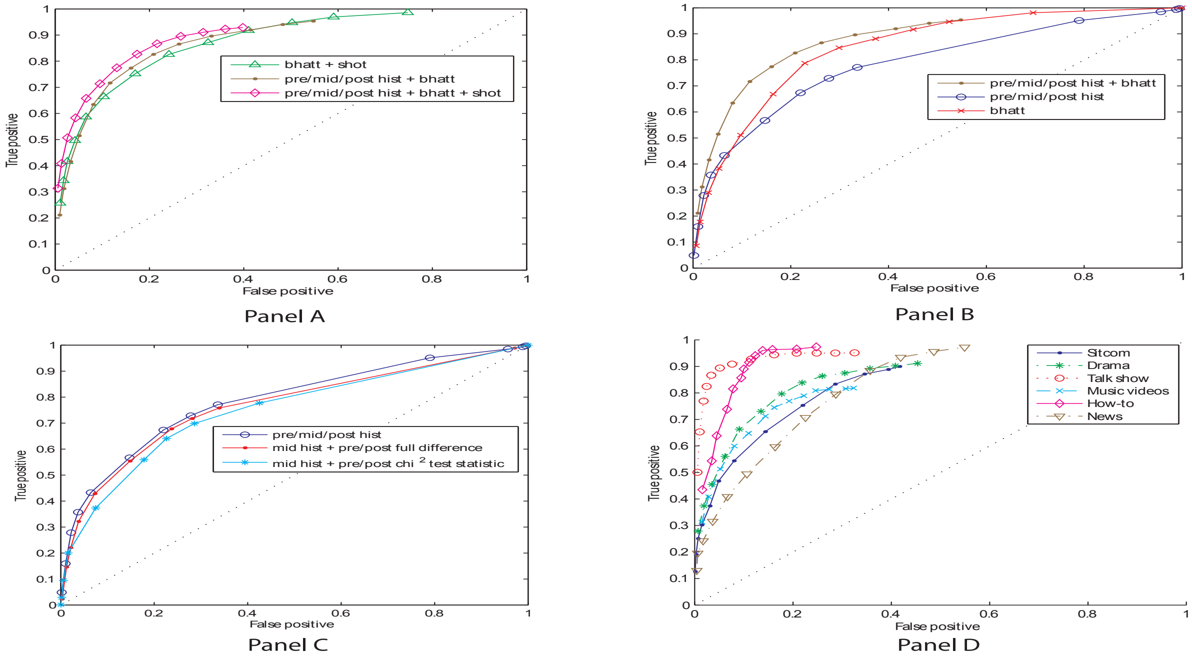

Fig. 3. ROC performance results: Panels A-C show combined performance over all content genres. Panel A shows the improvement achieved using average video shot cut counts. Panel B shows the improvement in combining bhattacharya measures with pre/mid/post histograms. Panel C shows a slight improvement in using concatenated histograms as opposed to histogram differences or the chi-square histogram statistic. Finally, Panel D shows performance results for a wide variety of video content.

do individually. In Panel C, we show that using concatenated histograms is superior to taking the absolute difference between histograms, or calculating the χ^2 statistic between histograms (a scalar quantity). This highlights the fact that the SVM is able to find a decision boundary automatically, and we do not require thresholding the difference between histograms.

In Fig. 3, Panel D, we show a comparison across a broad class of video genres such as sitcoms, dramas, talk shows, music videos, how-to's, and news. The optimal 12D input vectors used for this experiment were described in Section 2. From the results we see that the SVM framework scores best on talk shows, and performs the worst on news shows. When hand segmenting news shows, we noted that when a scene changed, it was more of a context change in the story, but the indicating factors we use such as audio or video shot counts did not reflect this change as strongly. In other content, it was also easier to note a scene boundary as a new guest appeared or a short music clip was inserted.

5. CONCLUSION

In this paper, we presented an SVM kernel-based classifier framework that is useful for comparing sets of features for scene change detection. The framework works over a wide class of broadcast content such as sitcoms, news, dramas, how-to's, music videos, and talk shows. In future work, we plan to experiment with additional video features to improve performance over diverse genres.

6. REFERENCES

[1] Rainer W. Lienhart, "Comparison of automatic shot boundary detection algorithms," 1998, vol. 3656, pp. 290–301, SPIE.

[2] H. Jiang, T. Lin, and H. Zhang, "Video segmentation with the support of audio segmentation and classification," in *Proc. IEEE ICME*, 2000.

[3] S. Lu, I. King, and M.R. Lyu., "Video summarization by video structure analysis and graph optimization," in *Proc. IEEE ICME*, 2004.

[4] H. Sundaram and S.F. Chang, "Video scene segmentation using video and audio features," in *Proc. IEEE ICME*, 2000.

[5] H. Sundaram and S.F. Chang, "Audio scene segmentation using multiple models, features and time scales," in *IEEE ICASSP*, 2000.

[6] John S. Boreczky and Lynn D. Wilcox, "A hidden markov model framework for video segmentation using audio and image features," in *Proc. IEEE ICASSP*, 1998.

[7] G. Camara Chavez, M. Cord, F. Precioso, S. Philipp-Foliguet, and A. de A. Araujo, "Video segmentation by supervised learning," in *Computer Graphics and Image Processing, 2006. SIBGRAPI '06*, 2006.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, August 2001.

[9] Feng Niu, Naveen Goela, Ajay Divakaran, and Mohamed Abdel-Mottaleb, "Audio scene segmentation for video with generic content," In submission to ICME, January 2007.