MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Broadcast Video Content Segmentation by Supervised Learning

Kevin Wilson, Ajay Divakaran

TR2009-085 July 2009

Abstract

Today's viewers of broadcast content are presented with huge amounts of content from broadcast networks, cable networks, pay-per-view, and more. Streaming video over the internet is beginning to add to this flow. Viewers do not have enough time to watch all of this content, and in many cases, even after selecting a few programs of interest, they many want to speed up their viewing of the chosen content, either by summarizing it or by providing tools to rapidly navigate to the most important parts. New display devices and new viewing environments, for example using a cell phone to watch content while riding the bus, will also increase the need for new video summarization and management tools. Video Summarization tools can vary substantially in their goals. For example, tools may seek to create a set of still-image keyframes, or they may create a condensed video skim [14]. Even after specifying the format of the summary, there can be different semantic objectives for the summary. A summary meant to best convey the plot of a situation comedy could differ substantially from a summary meant to show the funniest few scenes from the show. Most of these processing goals remain unachieved despite over a decade of work on video summarization. The fundamental reason for this difficulty is the existence of the "semantic gap", the large separation between computationally easy-to-extract audio and visual features and semantically meaningful items such as spoken words, visual objects, and elements of narrative structure. Because most video summarization goals are stated in semantic terms ("the most informative summary," "the most exciting plays of the match"), while our computational tools are best at extracting simple features like audio energy and color histograms, we must find some way to bridge these two domains.

Springer Book on Content Analysis

Broadcast Video Content Segmentation by Supervised Learning

Kevin W. Wilson and Ajay Divakaran

1 Introduction

Today's viewers of broadcast content are presented with huge amounts of content from broadcast networks, cable networks, pay-per-view, and more. Streaming video over the internet is beginning to add to this flow. Viewers do not have enough time to watch all of this content, and in many cases, even after selecting a few programs of interest, they may want to speed up their viewing of the chosen content, either by summarizing it or by providing tools to rapidly navigate to the most important parts. New display devices and new viewing environments, for example using a cell phone to watch content while riding the bus, will also increase the need for new video summarization and management tools.

Video summarization tools can vary substantially in their goals. For example, tools may seek to create a set of still-image keyframes, or they may create a condensed video skim [14]. Even after specifying the format of the summary, there can be different semantic objectives for the summary. A summary meant to best convey the plot of a situation comedy could differ substantially from a summary meant to show the funniest few scenes from the show.

Most of these processing goals remain unachieved despite over a decade of work on video summarization. The fundamental reason for this difficulty is the existence of the "semantic gap," the large separation between computationally easy-to-extract audio and visual features and semantically meaningful items such as spoken words, visual objects, and elements of narrative structure. Because most video summarization goals are stated in semantic terms ("the most informative summary," "the most exciting plays of the match"), while our computational tools are best at extracting

Kevin W. Wilson
Mitsubishi Electric Research Laboratory, e-mail: wilson@merl.com

Ajay Divakaran
Sarnoff Corporation e-mail: adivakaran@sarnoff.com

simple features like audio energy and color histograms, we must find some way to bridge these two domains.

This chapter presents our approach to bridging the semantic gap by supervised learning from hand-labeled examples. Our processing goal in this work is to locate all of the scene change locations in the content in a way that will work across a broad range of genres, including news, situation comedies, dramas, how-to shows, and more. We believe that this is a useful and semantically meaningful goal that can serve as a building block in a variety of higher-level video summarization systems.

The remainder of this section will give a high-level motivation for our design choices. Section 2 reviews previous work with an emphasis on scene change detection and on summarization techniques that use supervised learning. Section 3 gives a high-level overview of our approach to scene change detection. Sections 4 and 5 describe the features and classification algorithm used in our approach. Section 6 describes our experimental results, and Sections 7 and 8 conclude.

1.1 Why supervised learning?

Given sufficient training data, the relationship between input features, such as the cepstral audio features and color histogram-based video features that we use, and the output decision (between scene boundaries and non-scene-boundaries) can be determined by algorithms such as the support vector machine (SVM). This approach minimizes the degree to which features and thresholds must be hand-tuned, thus allowing us to quickly and easily add new low-level input features to our system. Domain-specific knowledge can be included in the system through appropriate choice of features.

1.2 Why scene changes?

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 thus allowing users to more quickly and effectively manage their content. Therefore, in this paper, the term “scene change” refers to a semantically meaningful change that will usually, but not always, 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 humans who label 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 obvious similarities. In contrast to scene changes, shot changes, the change from one continuous sequence filmed by a single camera to another such sequence, is a much-studied problem [7] 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.

2 Previous Work

This section reviews previous work on video summarization, emphasizing work on scene change detection and previous uses of supervised learning.

Beyond video summarization, semantic video processing has also been used for video search applications, recent examples of which include "Video Google" [12], MediaMill [17], and CuVid [3]. Much of the computational infrastructure for video search and video summarization overlaps, so some of the previous work we review has been applied to video search as well. However, the generall problem of video search is beyond the scope of this chapter.

2.1 Types of content

Audiovisual content can vary in narrative structure, in degree of editing, and in other ways that have major influences on subsequent processing.

One major axis of variation is between scripted content and unscripted content [18]. Scripted content, such as a situation comedy or news broadcast, tells a story or otherwise presents information in a structured, typically sequential, way. Ideally, one would summarize scripted content by understanding the "story" and retaining only the most important points. In contrast, unscripted content, such as a sporting event or a surveillance video, has little predetermined structure and can often be characterized as a small number of interesting events (e.g. goals in sports content or security breaches in surveillance content) scattered within a background of uninteresting events. Ideally, unscripted content can be summarized by including only interesting events and removing uninteresting events.

Another way in which content can vary is in the degree to which it is edited, produced, and post-processed. Typical broadcast content is captured by professional

camera and microphone operators and will include shot changes, scene changes, visual effects, sound effects, mixtures from several source audio tracks (music, character dialog, laugh tracks, etc.), and audio and visual level adjustments. All of these processing steps combine to yield a consistent and pleasant audiovisual experience. In contrast, unedited content, such as surveillance video or home videos, typically consists of long, uninterupted video shots that may not focus on important elements of the recorded action. One important consequence of the editing process is that it adds useful structure (laughter after a funny moment, ominous music during a scary scene, etc.) that can potentially be exploited by video summarization techniques.

Previous work on video summarization has also varied in its content-specificity. Some work has been specific to specific sports (such as soccer or basketball). Other work has been limited to fairly narrow genres, such as news broadcasts or music videos. Yet other work has been applied across a number genres. More genre-specific work has in general been able to exploit additional structure to achieve more semantically sophisticated processing goals.

2.2 Processing objectives

Once the type or types of content have been specified, the goal of the processing can be specified [14]. Goals can vary from low-level, such as a list of shot change locations, to high-level, such as a semantically meaningful summary of the content. Many high-level systems use lower-level processing techniques as building blocks.

A low- vs. high-level distinction that is important to our work is the difference between shot changes and scene changes. A shot is a consecutive sequence of frames captured from a single camera, while a scene is a semantically coherent video sequence which usually takes place in a single setting but may consist of several video shots. There are numerous challenges to reliably detecting shot changes [7], such as dissolves and wipes as transitions between shots, but clearly scene changes are a higher-level concept that requires more semantic knowledge to find.

Systems may also vary in the final format in which their results will be presented. One possible goal is to have a system that improves the interactive browsing experience by emphasizing important points in the content [11]. Another possible goal is to prepare a non-interactive summary of the content. Non interactive summaries may be presented as short “video skims” or as a collection of still keyframes.

The final consideration is the semantic goal of the output. For sports or surveillance, the goal may be to extract interesting events and ignore typical and uninteresting events. For a situation comedy or a drama, the goal may be to summarize the plot and/or include the most entertaining scenes from the program. For a news program, the summary might include only the most important points from the main news stories.

In all cases, the challenge is to relate the low-level features that can be extracted directly from the content to higher-level semantically meaningful concepts and structure. Achieving high-level semantic goals on a wide variety of content types

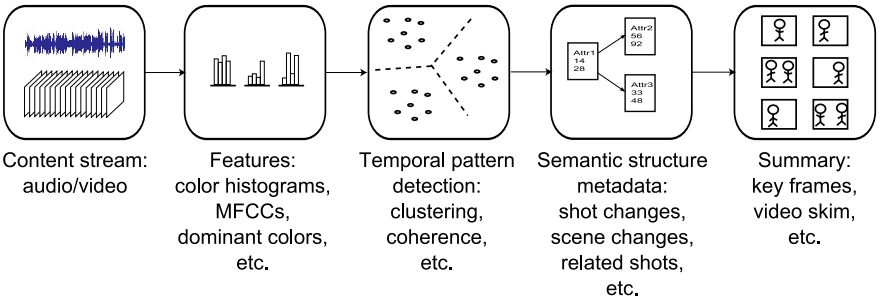


Fig. 1 Typical summarization system framework. Details and emphasis vary, but most previous summarization systems take in audiovisual content and extract low-level features. They then find temporal patterns, typically by clustering on feature similarity or segmenting based on feature coherence. From these patterns, estimates of semantic structure, such as shot or scene change locations, are made. Finally, the results are presented in a user-friendly format, such as a set of informative keyframes or a short video skim.

is an extremely challenging goal that we are not close to achieving. The following section reviews a variety of past work and describes some of the tradeoffs and design decisions that they have made in order to make progress on these problems.

2.3 Existing algorithms

This section presents a roughly chronological review of some of the most relevant video summarization work. Figure 1 shows the high-level schematic framework typical of these systems. All take in (audio)visual streams and all output some estimate of semantic structure, though they differ in their precise goals and in their lower-level design decisions.

Hongjiang Zhang and his collaborators did important early work on video summarization, which they review in [19]. They describe their goal as adding structure to (initially unstructured) video content. They detect shot boundaries using compressed-domain features, and they test this shot detection across a range of content types. Within each shot, they choose one or more keyframes to represent the shot. Although the primary use of these keyframes is as a summary of the content, another use is that visual features, such as color, texture, and shape features, are extracted from the keyframes to enable video search. Camera operations such as panning and zooming and temporal variations in brightness and color are also used to characterize shot content. These features, although intended primarily for video search, are also used to cluster similar shots to improve summarization and browsing.

Aigrain, Zhang, and Petkovic [2] review additional early work on video summarization and elaborate on some of the techniques described in [19]. They describe

several still-image processing techniques for extracting shape, texture, etc. that form the basis for many descriptors used in video summarization. They also make the important point that shot boundary detection is not sufficient for summarizing long-duration content because there can be 500 to 1000 shot changes per hour, a number which is impractical to present as a summary. To deal with long-duration content, they describe two techniques for doing scene change detection instead of shot detection. One genre-specific technique, which they applied to television news, is to detect specific semantically-meaningful shot types, such as shots of the news anchor, and to use these to impose additional structure. The other technique is to use a system of rules based on how video content is edited and perceived to attempt to define scene change boundaries. The former technique had the disadvantage of being very genre-specific, and the latter technique does not appear to have been evaluated thoroughly.

Kender and Yeo [6] approach the problem of scene change detection by defining a continuous-valued "coherence" at each shot change with the intent that scene change points will have low coherence and non-scene change points will have high coherence. They define a pairwise shot similarity based on color histograms and then define "coherence" as a weighted sum of all pairwise similarities between a shot before a potential scene change location and a shot after the potential scene change location. Similarities are weighted by temporal proximity, so dissimilar shots that are close together in time will be strongly indicative of a scene change location, while dissimilar shots that are well separated in time will not have much effect on the overall coherence. Kender and Yeo showed reasonable results on situation comedy and action movie content.

Sundaram and Chang [13] also confront the problem of scene change detection for video summarization. They define a "computational scene" to be a continuous segment of data that exhibits long-term consistency in its underlying low-level features. These features are color- and lighting-based for "video scenes" and based on a characterization of "ambient sound" for "audio scenes." A computational scene boundary is said to occur when a video scene boundary coincides with an audio scene boundary, or in other words when there are coinciding changes (at an appropriate time scale) in the audio features and video features. The hope is that a "computational scene" will correspond to a semantically meaningful scene. The determination of computational scene boundaries requires analysis of feature variation over time, and [13] describes two possible ways of doing this. One way is to perform a time-constrained clustering of shots and then create a transition graph indicating which shot clusters preceded which other shot clusters. The authors found this technique to be quite sensitive to the clustering algorithm parameters. The second proposed way is to define a causal fixed-duration processing window and to use a coherence measure similar to that of [6]. Sundaram and Chang discuss the application of their segmentation techniques to both still keyframe-based summaries and video skims.

Hanjalic, Lagendijk, and Biemond [4] present another approach for using pairwise shot similarity in a segmentation algorithm. (Their motivating application is video search and retrieval, but their segmentation technique could be applied to sum-

marization.) They place a threshold on the maximum dissimilarity within a scene instead of simply calculating weighted sums of coherence values. They apply their technique to two full-length movies.

Truong, Venkatesh, and Dorai [15] build on previous coherence-based scene change detectors by detecting features derived from an understanding of standard video editing and production techniques. These features include "tempo," based on shot length and motion activity and meant to differentiate fast-paced, energetic scenes from slow scenes, and "high impact colors," which finds unusual colors which may be indicative of some recurring theme in the content. Truong et al. provide a thorough evaluation of their techniques on ten full-length movies from a variety of genres.

Wei, Dimitrova, and Chang [16] present another approach to incorporating high-level semantic knowledge into a content analysis algorithm. Unlike other previous work described to this point, Wei et al. are not explicitly trying to segment the video content. Instead, their goal is to classify video segments according to "mood," for example, anger, fear, joy, etc. To do this, they extract color-histogram-based features and shot pace features and use labeled ground truth to train an SVM to classify eight different mood types, achieving roughly 80% accuracy across fifteen full-length films.

Adams, Venkatesh, Bui, and Dorai [1] impose a "three-act" structure on video content in which the three acts correspond to the setup, confrontation, and resolution of the story, and each act has a specified internal structure. They formulate a probabilistic model that relates this structure to lower level tempo features based on shot length, motion activity, and audio magnitude. They then use labeled training data to estimate the parameters of their probabilistic model. Finally, they use their model to make maximum a posteriori (MAP) estimates of the act boundaries. In subsequent, complementary work, Moncrieff and Venkatesh [9] suggest that a scene video content can be characterized as either an action scene, which emphasizes visual information, or a plot development scene, which emphasizes audio information. They detect changes in the scene type by looking for changes in the values of low-level audio features.

The most commercially successful application of video summarization to date is the work of Otsuka et al. [11]. In this work, the goal is to find highlights in sports content. The system uses labeled training data of highlights and non-highlights to learn a Gaussian mixture model (GMM) of the audio features for each of those two classes. These GMMs are used to classify the content's audio, and a graph of the "excitement level" as a function of time is presented to the user in a browsing interface that allows skipping forward and backward directly to exciting parts of the content. This work applies only to sports content, but it has been able to achieve a useful level of accuracy on this task because there is a reasonably straightforward relationship between exciting moments in sports content (which tend to cause cheers from the crowd and excited commentator speech) and low-level audio features.

In summary, researchers have been working on scene change detection for over a decade. Older work largely used changes in the distributions of low-level features as an indication of a scene change location. This is intuitively appealing and

reasonably successful, but it can be difficult to fine-tune the parameters of these systems to maximize performance. Newer work has begun to avoid the problem of hand-tuning by using labeled training data to learn the parameters of classifiers or generative models. It is difficult to directly compare the performance of different approaches because they are tested on different data and have no agreed-upon method of labeling ground truth. The large semantic gap between low-level features and high-level structure means that video segmentation remains a difficult and so far unsolved problem, but we believe that this use of training data is an important step in the right direction.

3 Our Approach

Our approach is to define several low-level audio and video features and to use labeled training data to train an SVM scene boundary classifier with those features as input. Like recent work such as [1, 11, 16], we use a training step so that we can avoid hand-tuning algorithm parameters. Unlike previous work, we use training data that both spans a wide variety of genres and is explicitly labeled with scene change locations. Because the training data implicitly defines what it means to be a scene change, we can learn a classifier that works on scene changes in a drama, which will often correspond to changes in time or filming location, and also on scene changes in news programs, which will often correspond to transitions between news stories. Given enough training data and a rich enough feature set, an SVM classifier should be able to correctly classify either of these types of scene changes. We feel that this is a powerful and conceptually simple path toward bridging the semantic gap.

Early work on scene segmentation emphasized that there should be low "coherence" across scene changes. In our system, we use Bhattacharyya shape and distance (described in detail below) as two of our features. These features measure the difference in lower-level feature distributions across a potential scene change, similar to what previous "coherence" features did, but we do not explicitly require local minima or maxima of these features to occur at scene boundaries. We allow the training process to determine what values will occur. We have chosen this and other features because of their intuitive appeal and because of the use of similar features in previous work, but once we have our training set labeled, we are free to experiment with many different possible features.

Our features are all defined on a local window around the current time point, so our scene change detection can be performed in a single pass by computing features in a sliding window. All of our audio and video features are computationally simple; some of the audio features are already in use on an embedded system as described in [11], and the MPEG-7 Scalable Color feature that we use was designed for computational simplicity. This computational simplicity should make it easy to apply our classifier in real time.

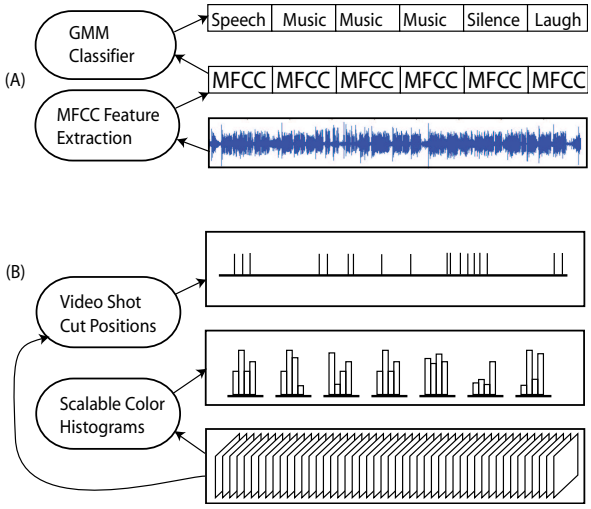


Fig. 2 Schematic overview of audio and video feature streams: (a) MFCC spectral coefficients are computed from the raw waveform. High level semantic labels are computed from the MFCC coefficients. (b) Video shot changes and scalable color histograms are both computed from the raw stream of video frames.

4 Feature Description

We use a discriminative Gausssian-kernel SVM framework [5] 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.

Video and audio feature streams are shown in Fig. 2. 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) [10]. 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. In addition to the semantic audio classes, we also use Bhattacharyya shape and distance parameters (described below) as features.

For Video, we use the MPEG-7 Scalable Color descriptor [8] for each frame. We also extract video frames (at 30 fps) and record the frame number of all shot cuts in the video. We use a basic hard shot cut detector [7].

Using the above audio and video features, we define an SVM input vector X_i for scene(+) and non-scene(-) boundaries as follows: $X_i = \{x_1, x_2, x_3, \ldots x_{13}, x_{14}\}$. In our experiments, our best-performing feature vector contained 14 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 selected) non-scenes. The first 9 components of X_i are histograms of semantic labels as explored in recent work [10], the next two components represent the difference between the audio distribution before and after a particular time t. The next component is based on video shot cut counts, and the final two components represent the difference between the color distribution before and after a particular time t. 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. **Audio 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 con-

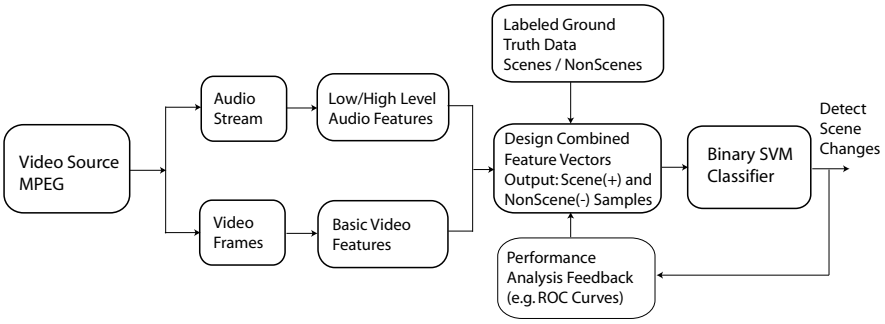


Fig. 3 SVM Classifier Framework.

tained 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 $[t-W_L, t+W_L]$.
6. **Color Bhattacharyya Shape+Distance:** variables x_{13}, x_{14}
   This feature is based on the MPEG-7 Scalable Color Descriptor [8] which is derived from a color histogram defined in the Hue-Saturation-Value color space. It uses a Haar transform encoding thus allowing scalable representation as well as scalable computation for increasing or decreasing the accuracy of the matching and extraction. It is exceptionally compact and easy to compute. We use the coarsest level, 16 parameters per frame, in the interest of computational simplicity. As for the audio Bhattacharyya descriptor, we compute the scalable color descriptor over a window before the point of interest and a window after the point of interest, and after computing diagonal Gaussians, carry out the Bhattacharyya shape and distance comparisons described above. One minor differency is that we have 30 descriptors per second because the video is at 30fps, while we have 92 audio feature vectors per second.

Since we use a kernel-based SVM with a smoothing bandwidth that is equal along all dimensions, we normalize all of the variables in X_i have approximately the same variance. After experimenting with different window sizes, we found that a window length of $W_L = 14$ seconds provided enough data to estimate the Bhattacharyya distances and semantic histograms and yielded good results.

5 SVM Classifier Framework

An SVM [5] 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 bound-

ary for the two classes that should generalize 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 14D vectors as described in Section 4. We expect that an SVM using our 14D feature input vector will be easily implementable on our product platform.

One advantage of the SVM framework is that the data $\mathbf{X}$ can be transformed to a higher dimensional *feature space* via a kernel function. Data may be linearly separable in this space by a hyperplane 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. Fig. 3 shows a block diagram of the overall SVM framework.

6 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 fraction of scene changes correctly detected by our system. The false positive rate is the fraction 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 select candidate locations to test for scene changes.

As shown in the top-most curve of Figure 4(a), using our full-featured 14D input vectors described in Section 4 (with concatenated histograms, Bhattacharyya measures, and shot counts) to describe scene vs. non-scene boundaries, our algorithm scores 70% true positive rate at a false positive rate of 5%. Allowing a higher false positive rate of 20%, the algorithm achieves a 90% detection rate. The remaining curves in Figure 4(a) show performance for various subsets of our full feature set. One notable result is that the shot count feature, which in our experience gives the worst performance of any individual feature, is still performing well above chance. Also notable is that all feature subsets perform worse than the full feature vector. This suggests that 14D is not too high-dimensional for the amount of training data that we use and that we could potentially improve performance further by adding new features. In generating these and all other ROC curves in this chapter, we averaged results from 10 runs, each time using a different set of randomly generated non-scene boundaries.

Figure 4(b) shows results for only within-program scene changes, i.e. excluding transitions from the main program to a commercial and excluding transitions between commercials. Most programs have a consistent style, even across scene changes, so scene changes within a program are typically more subtle than scene changes involving commercial transitions. Not surprisingly, then our overall performance drops substantially to a 55% true positive rate at 5% false positive and an 85% true positive rate at 20% false positive. The most notable pattern to this performance degradation is that our shot counts feature appears to be nearly useless for within-program scene changes. Shot counts by themselves are at close to chance performance, and shot counts in combination with other features perform about as well as those other features without shot counts included.

Figure 5 (a) and (b) show a genre-wise breakup with two different combinations of features. A comparison of Figure 5(a) and Figure 5(b) reveals that while almost all genres benefit from the addition of video features, sitcoms perform almost the same with and without video features. This may be because the laugh track and short, recurring musical themes provide most of the necessary information about scene change location. In general, however, it seems that audio and video are able to complement each other to improve performance in most cases. How-to videos and news videos had the worst performance, which is not surprising given that a scene change in these genres may consist of only a change in topic without any corresponding change in location or on-screen personalities.

7 Future directions

There are a number of possible future directions for this work. At the most concrete level, we would like to experiment with additional feature types, such as motion activity and possibly higher-level audio features. The strength of the supervised learn-

ing approach is that it will do the best it can with whatever input features are used. However, other than by using some expert knowledge in combination with extensive trial and error, it is not usually clear what input features are best. A more principled approach to choosing input features would be a major improvement.

Our supervised learning approach has led to some limitations. It is sometimes the case that to improve the performance on one genre, the SVM decision rule would have to worsen the performance on another genre. In cases like this, the tradeoff between genres will be made based on the amount of training data from each genre in combination with any user-defined weighting of this training data. Given enough training data and rich enough features, we believe that such tradeoffs can be managed successfully, but this issue must be settled empirically. Another approach to this genre tradeoff problem is to automatically determine the content's genre first and then to apply a genre-specific scene change detector. We have avoided this approach so far because we suspect that automatic genre determination will be difficult to do reliably and that the increased complexity of such a system is not worthwhile at present.

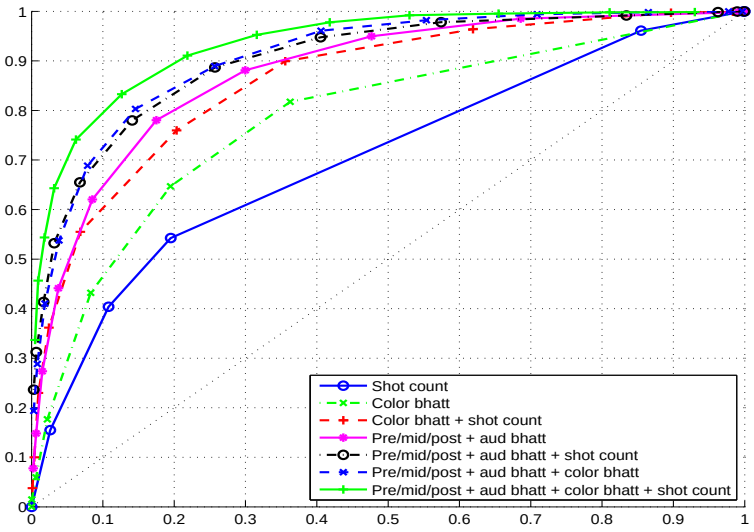
8 Conclusion

In this chapter, we reviewed previous work on video scene segmentation and presented our 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.

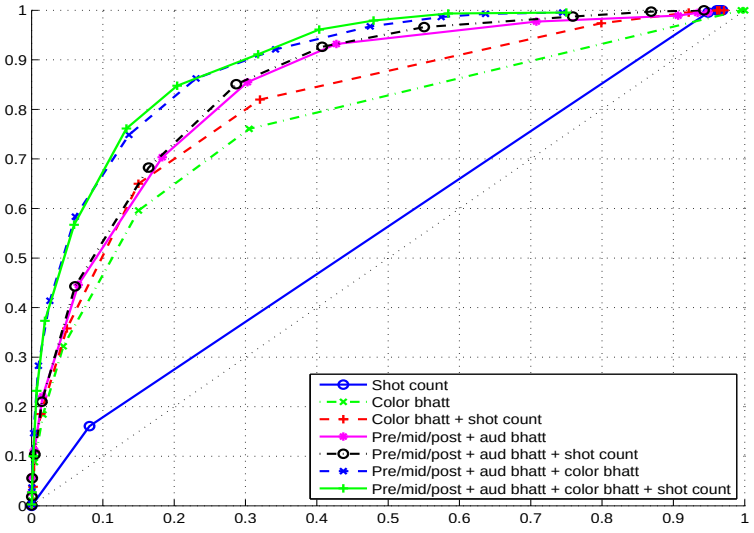
References

1. Adams, B., Venkatesh, S., Bui, H.H., Dorai, C.: A probabilistic framework for extracting narrative act boundaries and semantics in motion pictures. Multimedia Tools and Applications **27**(2) (2005)
2. Aigrain, P., Zhang, H., Petkovic, D.: Content-based representation and retrieval of visual media: A state-of-the-art review. Multimedia Tools and Applications **3**(3) (1996)
3. Chang, S.F., Kennedy, L.S., Zavesky, E.: Columbia university's semantic video search engine. In: ACM Conference on Image and Video Retrieval (2007)
4. Hanjalic, A., Lagendijk, R.L., Biemond, J.: Automated high-level movie segmentation for advanced video-retrieval systems. IEEE Transactions on Circuits and Systems for Video Technology **9**(4) (1999)
5. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer (2001). URL `http://www.amazon.co.uk/exec/obidos/ASIN/0387952845/citeulike%-21`
6. Kender, J., Yeo, B.: Video scene segmentation via continuous video coherence. In: IEEE Conference on Computer Vision and Pattern Recognition (1998)
7. Lienhart, R.W.: Comparison of automatic shot boundary detection algorithms. pp. 290–301. SPIE (1998). URL `http://link.aip.org/link/?PSI/3656/290/1`

8. Manjunath, B.S., Salembier, P., Sikora, T. (eds.): Introduction to MPEG-7 Multimedia Content Description Interface. Wiley (2002)
9. Moncrieff, S., Venkatesh, S.: Narrative structure detection through audio pace. In: IEEE Multimedia Modeling (2006)
10. Niu, F., Goela, N., Divakaran, A., Abdel-Mottaleb, M.: Audio scene segmentation for video with generic content (2007). In submission to ICME
11. Otsuka, I., Radhakrishnan, R., Siracusa, M., Divakaran, A., Mishima, H.: An enhanced video summarization system using audio features for a personal video recorder. IEEE Transactions on Consumer Electronics **52**(1) (2006)
12. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: IEEE International Conference on Computer Vision (2003)
13. Sundaram, H., Chang, S.F.: Video analysis and summarization at structural and semantic levels. In: D. Feng, W.C. Siu, H. Zhang (eds.) Multimedia Information Retrieval and Management: Technological Fundamentals and Applications. Springer Verlag (2003)
14. Truong, B.T., Venkatesh, S.: Video abstraction: A systematic review and classification. ACM Transactions on Multimedia Computing, Communications and Applications **3**(1) (2007)
15. Truong, B.T., Venkatesh, S., Dorai, C.: Scene extraction in motion pictures. IEEE Transactions on Circuits and Systems for Video Technology **15**(1) (2003)
16. Wei, C.Y., Dimitrova, N., Chang, S.F.: Color-mood analysis of films based on syntactic and psychological models. In: IEEE International Conference on Multimedia and Expo (2004)
17. Worring, M., Snoek, C.G.M., de Rooij, O., Ngyen, G.P., Smeulders, A.W.M.: The mediamill semantic video search engine. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (2007)
18. Xiong, Z., Radhakrishnan, R., Divakaran, A., Rui, Y., Huang, T.S.: A Unified Framework for Video Summarization, Browsing, and Retrieval. Elsevier (2006)
19. Zhang, H., Low, C.Y., Smoliar, S.W., Wu, J.H.: Video parsing, retrieval, and browsing: An integrated and content-based solution. In: ACM Multimedia (1995)

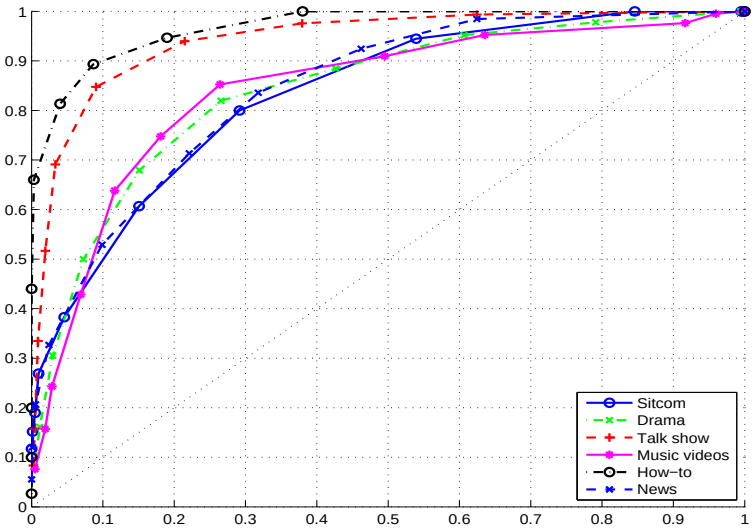


(a) Overall performance, all scene changes including commercials

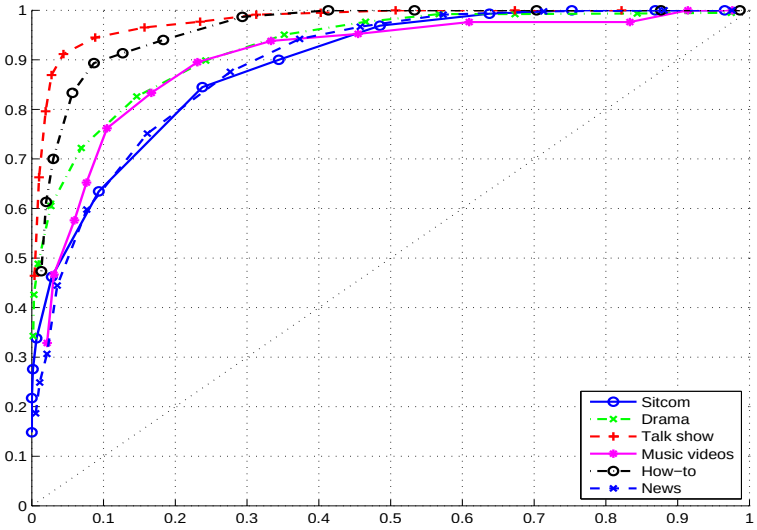


(b) Overall performance, excluding commercials

Fig. 4 Overall ROC results: All curves in each panel are generated with a single classifier, and in both plots the horizontal axis is false positive rate and the vertical axis is true positive rate. (a) and (b) show performance averaged across all genres with and without commercial transitions included, respectively.



(a) Audio features alone by genre



(b) Combined audio and video features by genre

Fig. 5 Genre-specific ROC results: All curves in each panel are generated with a single classifier, and in both plots the horizontal axis is false positive rate and the vertical axis is true positive rate. (a) and (b) show performance by genre for audio features only and for combined audio and video features, respectively.