A Robust Video Scene Extraction Approach to Movie Content Abstraction

Ying Li,[1] C.-C. Jay Kuo[2]

[1] IBM T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532

[2] Department of Electrical Engineering, University of Southern California, Los Angeles, California 90089

ABSTRACT: This research addresses the problem of automatically extracting semantic video scenes from feature films based on multimodal information. A three-stage scene detection scheme is proposed. First, we use pure visual information to extract a coarse-level scene structure based on generated shot sinks. Second, audio cue is integrated to refine the scene detection results by considering various kinds of audiovisual scenarios. Finally, we introduce users into this process by allowing them to interactively tune the final results to their own satisfaction. The generated scene structure forms a compact yet meaningful abstraction of the video data, which can help facilitate the content access. Preliminary experiments on integrating multiple media cues for movie scene extraction have yielded encouraging results.

© 2004 Wiley Periodicals, Inc. Int J Imaging Syst Technol, 13, 236–244, 2003; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/ima.10063

Key words: movie content analysis; automatic video scene detection; adaptive silence detection; audio content analysis

1. INTRODUCTION

The amount of multimedia information generated in today's society is growing rapidly, which poses a serious technological challenge on how the information is integrated, processed, organized, summarized, and indexed in a semantically meaningful manner. Of all existing media types, video is the most challenging one because of its rich content. Recently, research on efficient video indexing, browsing, and retrieval has attracted increasing attention. Many applications have emerged in areas such as video-on-demand, distributed multimedia systems, digital video libraries (Flickner et al., 1995; Wactlar et al., 1996), distance education, entertainment, surveillance, and geographical information systems (MPEG-7 Requirement Group, 1999).

When the amount of data is small, users can retrieve the desired content in a linear fashion by simply browsing the data sequentially. However, with a huge amount of data, a linear search is no longer feasible. What is needed is the capability of automatically abstracting the essential video content and forming a compact yet meaningful data representation for effective information access. To accomplish this goal, a hierarchy with multiple layers of abstraction is desired as shown in Figure 1.

Most earlier work on video abstraction was centered around shot-based video representation, where a video program was represented in the form of cascaded shots (Zhang et al., 1993; Yeo and Liu, 1995). However, low-level shot detection usually results in a video segmentation result that is too fine to be useful. Besides, the syntactic shot structure does not correspond to the underlying video semantics in a direct and convenient way. Therefore, most current work focuses on group- or scene-based video segmentation. Along this direction, a video sequence is first segmented into shots and then is semantically related and temporally adjoining shots are further grouped into scenes based on either color similarity, motion continuity, or temporal locality as reported by Yeung et al. (1996), Rui et al. (1996), and Lin et al. (2001). More recently, people have started to integrate audio cues into the system framework because the accompanying audio source also contains important information. For instance, Li and Ming (2001) proposed to integrate the shot-based audio classification results with visual analysis results for a better system performance. Similar ideas were also explored by Jiang et al. (2000), except that they used audio classification results to guide visual content analysis. To extract scenes from TV news, Nam et al. (1997) performed a speaker change detection by tracking voiced phonemes in audio signals. A more sophisticated audio-visual-based scene detection scheme was proposed by Yoshitaka and Miyake (2001), where analyses on cut detection and video editing effects as well as background noise/music detection were combined to make the final decision.

The goal of this research is to develop a robust video scene extraction scheme by integrating audiovisual cues. In particular, we first use pure visual information to generate a coarse-level scene structure based on computed shot sinks. Then, the audio cue is integrated to refine the scene results by considering various kinds of audiovisual scenarios. Finally, we introduce users into this process by allowing them to interactively tune the results to their own satisfaction. The major difference between this and other existing work is that we have considered more complex and thus more practical audiovisual scenarios for generic video contents. Moreover, the proposed system is built on a more sophisticated audio classification scheme.

Movie is our major application target because it has a clear story structure that can be well exploited and exemplified by our ap-

Correspondence to: Ying Li; e-mail: yingli@us.ibm.com
Contract grant sponsor: Integrated Media Systems Center
Contract grant sponsor: National Science Foundation Engineering Research Center; contract grant number: Cooperative Agreement Number EEC-9529152
Contract grant sponsor: Hewlett-Packard Company

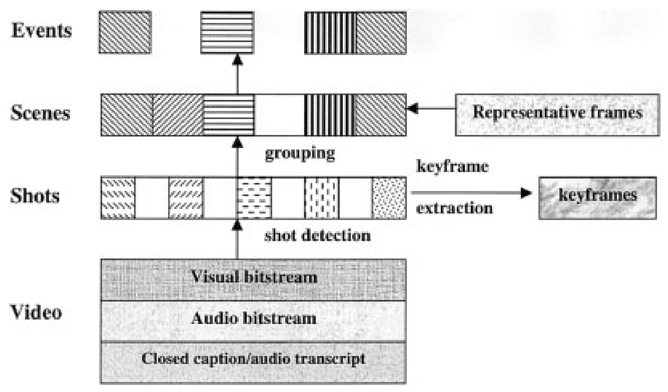


FIGURE 1. A hierarchical representation of video content.

proach. Moreover, movie has many special characteristics, such as the use of complex film-editing techniques. It is possible that different editing styles on content presentation, shot selection, ordering, and timing may result in different stories (Reisz and Millar, 1968). Therefore, it is not only interesting but also challenging to work with movies, because all these special features need to be taken into account for better content understanding and abstraction.

The rest of this article is organized as follows. We briefly review our previous work on low-level audiovisual content analysis in Section 2. In Section 3, we elaborate on the proposed scene detection scheme, which includes the sink-based scene extraction, the audiovisual-based scene refinement, and the user-interaction three parts. Experimental results obtained on five test TV programs and movies are reported in Section 4. Finally, concluding remarks are drawn in Section 5.

2. AUDIO AND VISUAL CONTENT PREANALYSIS

The first step toward visual content analysis is shot detection. In this work, a color histogram-based approach is used to perform this task. Specifically, once a distinct peak is detected in the frame-to-frame histogram difference, we declare it as a shot cut (Li and Kuo, 2000). An average of 92.5% recall and 99% precision rates have been achieved in this work. In the second step, we proceed to extract one or more keyframes from each shot to represent its underlying content. For simplicity, we assign the first and last frames of each shot as its keyframes.

In the audio domain, the following two major tasks are carried out to analyze the embedded audio content:

1. Audio feature extraction. Six types of audio features are considered in this work, which include short-time energy function, short-time average zero-crossing rate (ZCR), short-time fundamental frequency (SFuF), energy band ratio (EBR), silence ratio (SR), and Mel-Frequency Cepstral Coefficients (MFCCs; Zhang and Kuo, 2001; Reynolds and Rose, 1995). Specifically, the short-time energy feature provides a representation of the signal's amplitude variation along time, and the ZCR feature gives a simple measure of the signal's frequency content. The SFuF feature is mainly extracted to determine the signal's harmonic nature, and the energy ratio of low- to high-frequency bands is an important indicator for voiced and unvoiced signal detection. Finally, the SR feature measures the ratio of accumulated silence intervals to the entire signal period, and the MFCC feature gives the signal's frequency spectrum on the mel-frequency scale. In summary, the first five features are primarily extracted for audio classification purpose, whereas the MFCC feature is mainly used for speech similarity comparison because of its successful usage in various speech processing applications.
2. Shot-based audio classification. In this step, each shot is classified into one of the following six classes based on the five features extracted from its audio content: silence, speech, environmental sound, music, speech with music background, and environmental sound with music background. Specifically, silence is detected by thresholding the energy and ZCR features; speech is recognized by exploiting the inter-relationships among energy, ZCR and SFuF, as well as by checking EBR and SR values. Finally, music is separated out by evaluating both ZCR and SFuF features. More detailed discussion on this work can be found in Zhang and Kuo (2001). For the rest of this article, we call the first three classes as the nonmusic class, and the last three, the music class.

3. VIDEO SCENE CONSTRUCTION

Scene is a subjectively defined concept that depicts and conveys a high-level story. Generally speaking, a scene contains a collection of semantically related and temporally adjacent shots that have the following three features.

- Visual similarity. Similar visual contents, such as similar color layout and continuous object activities, could be observed in these shots, especially in movies because of one of the filming techniques called *montage*. As described by Tarkovsky (1986), "One of the binding and immutable conditions of cinema is that actions on the screen have to be developed sequentially, regardless of the fact of being conceived as simultaneous or retrospective.... In order to present two or more processes as simultaneous or parallel you have to show them one after the other, they have to be in sequential montage." This means that, in order to convey conversations, innuendos, or reactions, filmmakers have to repeat important shots to express the content and motion continuity. This type of editing style is actually very popular in daily TV series and movies, where repeating shots of persons or settings are frequently interleaved with other shots.
- Audio similarity. The similarity of audio contents is manifested as similar background noise that exists in these shots. In addition, if the same person is talking in several shots, his speeches in all these shots should present similar acoustic characteristics.
- Time locality. Shots are temporally close to each other if they are within the same scene; otherwise, it is less likely that they belong to the same thematic topic. For instance, given two shots of the same person, if they are juxtaposed together, they are more likely to be in the same scene than the case where they are placed far apart from each other.

By taking the above three features into account, we have developed a robust scene detection scheme that consists of the following three stages: *sink-based scene construction*, *audiovisual-based scene refinement*, and *user interaction*. Each of them is detailed below.

3.1. Sink-based Scene Construction. In this step, we aim at extracting all video segments that are characterized by a repetitive

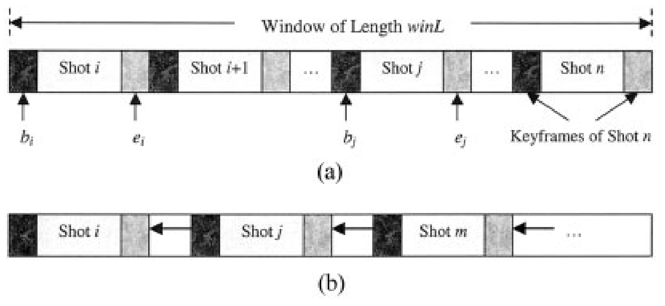


FIGURE 2. (a) Shots contained in a window of length *winL*, and (b) shot *i*'s sink.

visual structure. A new concept called *shot sink* is introduced for this purpose. Particularly, a shot sink contains a pool of shots that are temporally close and visually similar. Shot sinks are generated using the proposed window-based sweep algorithm as described below.

3.1.1. Window-based Sweep Algorithm. Given shot *i*, this algorithm finds all shots that are visually similar to *i* and pushes them into its sink. However, because of the scene's temporal locality feature discussed earlier, this search will be constrained to a certain temporal range, as indicated by the timing window in Figure 2(a), where the current window contains $n - i + 1$ shots and has a length of *winL*. To compare the visual similarity of two shots, in principle we should compare every pair of video frames, with each taken from one shot. However, to reduce the complexity of this process, only keyframes are used for comparison. This is acceptable because keyframes are representatives of the shot.

Denote shots *i* and *j*'s keyframes by b_i, e_i, and b_j, e_j $(i < j)$ as shown in Figure 2(a); we compute the similarity between shots *i* and *j* as

$$Dist_{i,j} = \frac{1}{4}(w_1 \times dist(b_i, b_j) + w_2 \times dist(b_i, e_j) + w_3 \times dist(e_i, b_j) + w_4 \times dist(e_i, e_j)), \quad (1)$$

where $dist(b_i, b_j)$ could be either the Euclidean distance or the histogram intersection between b_i and b_j's color histograms. w_1, w_2, w_3, and w_4 are four weighting coefficients that are computed as

$$w_1 = 1 - \frac{L_i}{winL}, \quad w_2 = 1 - \frac{L_i + L_j}{winL},$$

$$w_3 = 1, \quad w_4 = 1 - \frac{L_j}{winL}, \quad (2)$$

where L_i and L_j are lengths of shots *i* and *j* in terms of frames, respectively. The derivation of these four coefficients is explained as follows. First, due to the *montage* effect, we know that when shots *i* and *j* are within the same thematic topic, they would share certain visual similarity although shot *j* further advances shot *i*'s content. To measure the content similarity between these two shots, we first check the similarity between e_i and b_j since they form the closest frame pair and should have the smallest distance if shot *j* does continue shot *i*'s content. Thus, we set w_3 to 1. In contrast, the similarity between b_i and b_j becomes smaller as shot *i* gets longer, thus we set w_1 to be $1 - (L_i/winL)$, where *winL* is introduced for the normalization purpose. Similarly, we can derive the formulas for w_2 and w_4.

Now, if $Dist_{i,j}$ is less than a predefined threshold *shotT*, we consider shots *i* and *j* to be similar and put shot *j* into shot *i*'s sink. As shown in Figure 2(b), all shots similar to shot *i* are nicely linked together in their temporal order. One thing worth mentioning is that if shot *i*'s sink is not empty, we have to compute distances from the current shot, say, shot *m*, to all other resident shots in the sink (shots *i* and *j* in this case). Shot *m* is only qualified to be in the sink when the average of all distances is less than *shotT*.

Basically we will run this algorithm for every shot. However if one shot has already been included in a sink, we will skip this shot and continue with the next.

Two parameters are used in this algorithm, i.e., the window length *winL* and the threshold *shotT*. Below are some discussions on how to determine them.

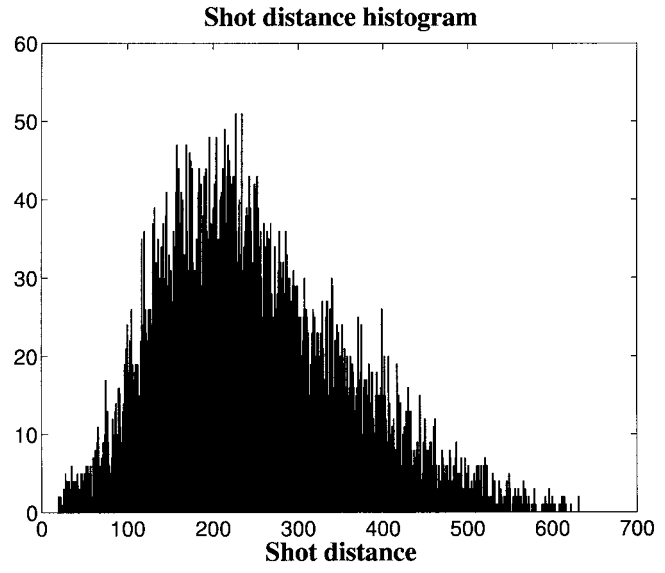
1. Determining window length *winL*. We have tried two ways to choose parameter *winL*, namely, a fixed value and an adaptive value that varies with every incoming movie. In the former case, we empirically set *winL* to be a predefined value that covers the duration of an ordinary movie scene. In the latter case, *winL* is set to be proportional to the average shot length. Hence, the faster the movie tempo, the shorter the window length. On the basis of our experiments, we find that a fixed value usually produces better results, which is perhaps due to the reason that as a semantic unit, scene is somehow independent of the underlying shot structure. *winL* is set to be 2000 (frames) in this work.

2. Determining threshold *shotT*. Parameter *shotT* is used to threshold the similarity measurement between two shots. Because our distance metric uses the color information and because different movies tend to have different primary hue, an empirically set threshold may not always work. Figure 3(a) shows a shot distance histogram for one test movie where each distance is computed from one shot to another within the temporal window. As we can see, a Gaussian density function $N(\mu; \sigma)$ can be used to approximate this distance distribution. Inspired by this finding, we propose to determine the threshold as follows. First, we normalize each computed distance $Dist_{i,j}$ with μ and σ, i.e.,

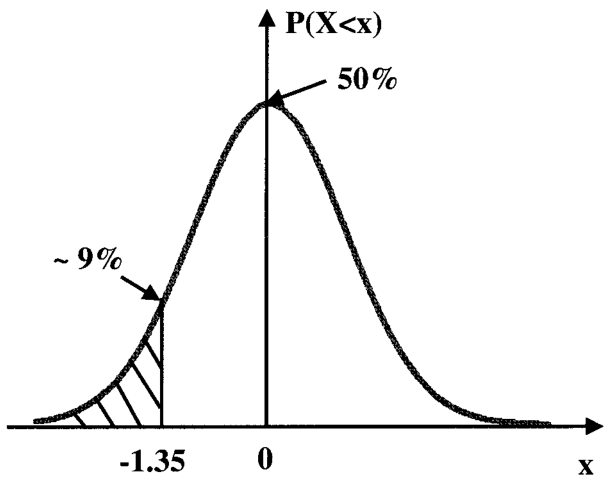
$$Dist_{i,j} = \frac{Dist_{i,j} - \mu}{\sigma}. \quad (3)$$

Then, we compare it with another threshold *shotT*′ which is derived from the Gaussian density function. Parameter *shotT*′ can be easily adjusted to fit all movies because it applies to normalized distances. Empirically, we find that $shotT' = -1.35$ produces a good result for all test data, where about 9% of the shots in the timing window are qualified for the sink because $P(X < x \mid x = shotT' = -1.35) = 0.089$ as shown in Figure 3(b).

After obtaining shot sinks, we proceed to extract the coarse-level scene structure. The basic principle applied here is, whenever there is a temporal overlap between two sinks, we label them to be in one scene. This is due to the fact that no shots that are inter-related will belong to different scenes, because different scenes usually focus on different thematic topics and do not overlap in the time domain. Moreover, the shots that do not belong to these sinks but are covered by their temporal ranges are also included into the same scene. For instance, if shot 1's sink contains shots 1, 3, 5, 7, and 10, and shot 2's sink contains shots 2, 4, 6, 8, and 12, it is natural for us to group shot 1 to shot 12 into one scene, because apparently something is



(a)



(b)

FIGURE 3. (a) The shot distance histogram for a test movie, and (b) the normalized distribution of shot differences.

going on among these two sets of shots. This type of repetition pattern can be frequently found in TV series and movies. When a scene contains only one shot, we label it as an *isolated scene*.

3.2. Audiovisual-based Scene Refinement. Two major types of errors are observed in coarse-level scene detection: *false negatives* (misses) and *false positives* (false alarms). When the temporal window is too narrow, false alarms might occur because the shot sink cannot cover all similar shots, and a similar new scene would be initialized by mistake. Moreover, scenes without any repetition patterns, which are called *progressive scenes*, may result in isolated scenes. On the other hand, when the temporal window is too wide, we might introduce false negatives, where one scene is falsely merged into another scene. This situation usually occurs when the two neighboring scenes have similar background or the same movie characters, but different thematic topics.

Because we can reduce false negatives by shortening the timing window, we will mainly focus on the solution to reduce false positives by using the accompanying audio information. To achieve this goal, we first classify each shot into one of the six audio classes as described in Section 2. Then, neighboring scenes are possibly merged according to a set of heuristically derived rules as detailed below.

3.2.1. Rule-based Scene Mergence. At this step, a set of audiovisual-based rules are developed to possibly merge two neighboring scenes. For convenience, let us take scenes k and $k + 1$ as an example. Figure 4 shows the structure of these two scenes, where shot j is the last shot of scene k and shot $j + 1$ is the first shot of scene $k + 1$. The starting and ending frames of shot j are denoted by b_j and e_j, respectively. For the rest of this subsection, we will mainly work with these two shots because they form the actual connecting interface of the two neighboring scenes.

On the basis of extracted audiovisual features of these two shots, we have considered five different scenarios as described below. Within each scenario, one or two rules are derived to determine if these two scenes should be merged or not.

Scenario I: Both shots belong to the music class:

- Rule 1: If both shots belong to the music class, i.e., they are either pure music, speech with music background, or environmental sound with music background, we merge these two scenes together.

The rationale of this rule is that music is usually mixed by the music director during the postproduction so as to convey the inner feeling of a key story figure or to reflect the atmosphere under a certain circumstance. In other words, music is deliberately added so as to enhance viewers' experience for an unbreakable story unit. Therefore, in most cases, two neighboring scenes of different thematic topics do not share the same background music.

When several temporally consecutive, but visually different shots share the same piece of music, it is the director's implication that all of them are actually semantically related and thus, should be recognized as one unit. This situation is not common, but does exist in feature films, which could be seen as a type of directorial style. For instance, one of our test movies contains a very long scene which describes a girl's daily life in a ranch farm from various view aspects. Consequently, this scene consists of many shots with totally different visual contents. Nevertheless, its continuous music background gives us the clue that all shots contribute to the same story unit.

Finally, it is worthwhile to point out that when we apply Rule 1, we assume that both shots share the same music piece. This assumption may not be true in certain types of movies, e.g., in a musical, where there are many music or songs constantly changing from time to time or from shots to shots. However, because it needs some extra work to distinguish two different music pieces, currently we assume that in our test movies, no two music pieces with different themes are played successively.

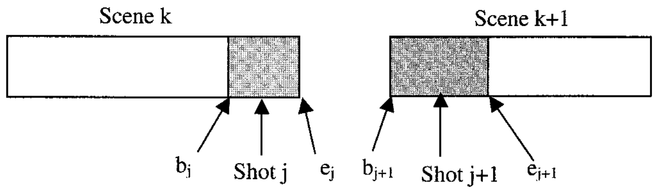


FIGURE 4. An example of two temporally adjoining scenes.

Scenario II: Both shots belong to the nonmusic class:

We will further divide this scenario into the following two subscenarios according to their specific audio types.

Subscenario II (a): Both shots contain speech:

- Rule 2: If both shots contain speech and such continuous speech is detected across the scene boundary, we merge these two scenes together.

The rationale of this rule is rather intuitive. A continuous speech flow across the scene boundary means that there is a continuous story flow over the two scenes. Thus it is natural for us to merge them together. The two steps adopted in this process are given below.

1. For the video segment starting from b_j and ending at e_{j+1}, i.e., the segment containing both shots, we identify and isolate the "speech segments" from the background using the proposed *"adaptive silence detection"* algorithm, which will be detailed below.
2. Check if the scene boundary is contained in one of the detected speech segments. If yes, it means that a continuous speech flow is detected across the scene boundary, and thus we merge the two scenes together. Otherwise, we proceed to subscenario II (b).

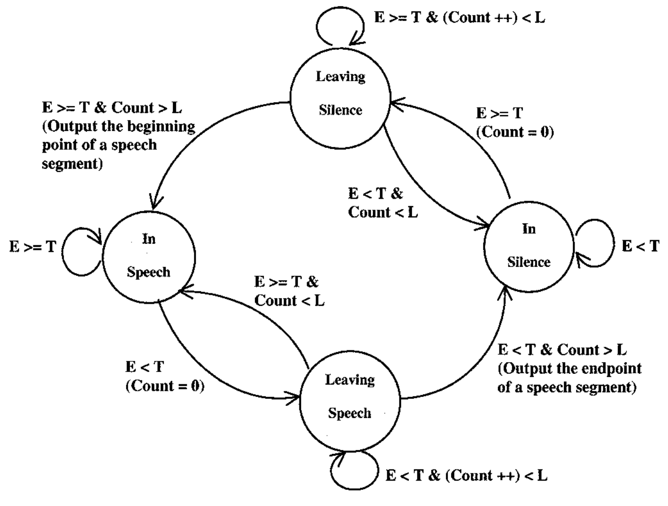
Adaptive Silence Detection. A classical approach to silence detection relies on a global energy thresholding scheme (Zhang and Kuo, 2001; Rabiner and Schafer, 1978). Although this simple scheme works well for the static audio content, it is not suitable for movies where various complex audio types exist. More recent work in this area focuses on the end-of-utterance detection, which mainly targets at real-time ASR (automatic speech recognition) under an adverse environment (Li and Zheng, 2001; Hariharan et al. 2001). These methods are also not applicable to our work because their ultimate goal is to collect every utterance, rather than excluding every silent period as demanded by our case.

In this work, we propose to detect silence by adapting to the underlying dynamic audio content. Particularly, given the audio signal of one speech shot, we first sort all audio frames into an array based on their energies precomputed in the dB scale. Then, for all frames whose energy values are greater than a preset threshold *engyT*, we quantize them into N bins where bin_1 has the lowest and bin_N has the highest average energy. Because we already know that both silence and speech signals are present, obviously bin_1 gives the lower boundary of the silence energy and bin_N has the upper limit of the speech energy. Thus, the threshold T separating speech and silence should be a value between these two extremes. Specifically, we calculate T as

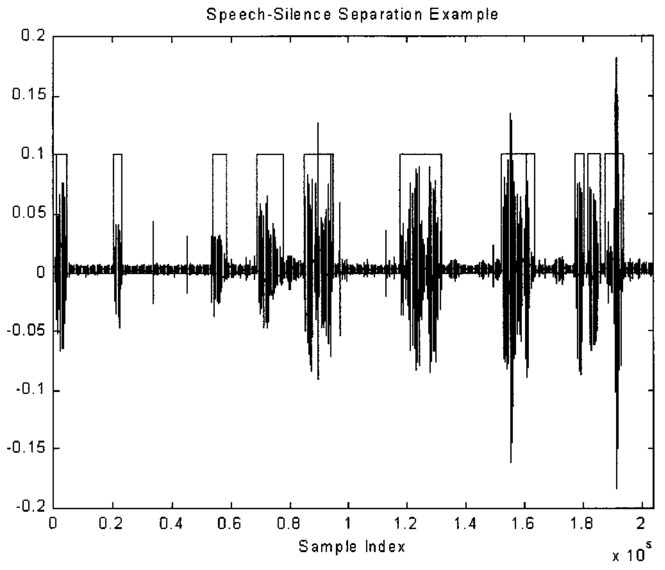
$$T = ENGY_{sl} + \alpha \times (ENGY_{sp} - ENGY_{sl}), \tag{4}$$

where $ENGY_{sl}$ and $ENGY_{sp}$ are the average energies in the first and last three bins, respectively, and α is a weighting coefficient that is set to 0.4 in this work. Also, we set *engyT* to 30.0 and N to 10. Obviously, with this approach, no matter how the background noise changes, the threshold could always be adaptively adjusted.

Next, we use a four-state transition diagram (Li and Zheng, 2001) to separate speech segments from the background using the obtained threshold T. As shown in Figure 5(a), the diagram has the following four states: *in-silence*, *in-speech*, *leaving-silence* and *leaving-speech*. Either in-silence or in-speech can be the starting state, and any state can be a final state. The input of this state machine is a sequence of frame energies, and the output is the beginning and ending frame indices of detected speech segments. The transition conditions between two states are labeled on each edge, and the corresponding actions are described in parentheses. In particular, *Count* is a frame counter, E denotes the frame energy, and L indicates the minimum length of a silence or speech segment, which is set to be 300 ms in the current work. As we can see, this state machine basically groups blocks of continuous silence/speech frames as silence/speech segments while removing impulsive noise at the same time.



(a)



(b)

FIGURE 5. (a) A state transition diagram for speech-silence segmentation where T stands for the derived adaptive threshold, E denotes the frame energy, *count* is a frame counter, and L indicates the minimum speech/silence segment length, and (b) a speech-silence segmentation example where detected speech segments are bounded by the passbands of the superimposed pulse curve.

This algorithm works best when it is performed within a shot because we can assume a quasi-stationary background in this case. For the rest of this subsection, the two terms *silence* and *background noise* will be used interchangeably because they mean the same thing in this work.

Figure 5(b) gives a speech-silence segmentation example on an audio signal recorded from a speech shot. A pulse curve is used to illustrate the results where detected speech segments are bounded by the passbands. As we can see, all speech fragments are successfully isolated from the impulse noise. The loud background sounds are also removed.

Subscenario II (b): For all other cases:

- Rule 3: For all other cases, if these two shots have similar background noise, we merge these two scenes together.

The philosophy behind this rule is that, because a scene usually progresses under a certain circumstance; if two shots share similar background noise, it is highly possible that they belong to the same scene. To measure the background similarity, we use the following two processing steps.

1. Detect all speech and silence segments for both shots using the proposed adaptive silence detector. Then, we compute short-time energy for every detected background segment. Finally, for all background segments within each shot, we compute the mean μ and standard deviation σ in terms of their energy values, and denote the parameters by $E(\mu_j, \sigma_j)$, and $E(\mu_{j+1}, \sigma_{j+1})$ for shots j and $j+1$, respectively.
2. Compute the *Kullback Leibler 2* (KL2) distance between the above two parameter pairs. If it is less than threshold *spbgT*, it means that similar background noise is detected in these two scenes, and we thus merge them together. Currently, the threshold *spbgT* is empirically determined and is set to be 7.0 in this work.

The adopted KL2 distance is briefly reviewed below.

Kullback Leibler Distance. The Kullback Leibler distance (or the relative cross-entropy) between two random variables A and B is an information theoretic measure that equals the additional bit rate accrued by encoding random variable B with a code that was designed for optimal encoding of A (Cover and Thomas, 1991). The larger this value, the greater the distance between the probability density functions (PDFs) of these two variables. Its formulation is given as

$$KL(A; B) = E_A(\log(P_A) - \log(P_B)), \tag{5}$$

where P_A and P_B are A and B's PDFs, respectively. $E_A(\cdot)$ is the expectation operation performed with respect to A's PDF.

Because this expression is asymmetric, it is not strictly a distance metric. Thus in practice, the KL2 distance, which is defined as (Siegler et al., 1997)

$$KL2(A; B) = KL(A; B) + KL(B; A), \tag{6}$$

is often used as an alternative.

When both A and B have Gaussian distributions as $A(\mu_A, \sigma_A)$ and $B(\mu_B, \sigma_B)$, the KL2 distance becomes

$$KL2(A; B) = \frac{\sigma_A^2}{\sigma_B^2} + \frac{\sigma_B^2}{\sigma_A^2} + (\mu_A - \mu_B)^2 \times \left(\frac{1}{\sigma_A^2} + \frac{1}{\sigma_B^2}\right) - 2. \tag{7}$$

Again, the larger the value, the greater the distance between the two variables. When A and B have the same PDF, this distance will reduce to 0.

The Kullback Leibler distance has been successfully used in speech and speaker recognition areas because of its effectiveness (Beigi et al., 1998) and is thus used in this work.

One thing worth pointing out is that although we can also use pure visual information to detect the background change, for instance, we can first segment the foreground objects from the background and then compare the background. Yet intelligent object segmentation still remains to be a difficult problem today; thus the use of audio information in this task appears to be a better solution.

Scenario III: One of the shots is from the music class, whereas the other is from the nonmusic class:

In most cases, this scenario will naturally result in two different scenes according to our discussion in Scenario I. However, we found in experiments that sometimes music will suddenly start or get stronger enough to be detected in the middle of a scene, thus causing two consecutive shots to have different audio background. To handle such a scenario, we use the following rule:

- Rule 4: In Scenario III, when both shots contain speech signals, i.e., one of them contains pure speech and the other, speech with music background, the two scenes will be merged if any of their speech segments are acoustically similar.

The rationale of this rule is, when acoustically similar speeches are detected in these two shots, it is likely that the same speaker is present or talking in both shots, which implies that they may belong to the same scene. The following three-step process is performed to measure the speech similarity:

1. For each shot, find all of its speech segments using the adaptive silence detector. For simplicity, let us assume that shot j has N_1 and shot $j+1$ has N_2 speech segments.
2. Compute a 14-dimensional MFCC feature vector for every speech frame and subsequently calculate the mean vector $\vec{E}$ and covariance matrix C for every speech segment. To simplify the computation, we assume C to be a diagonal covariance matrix. As a result, we have totally N_1 pairs of $(\vec{E}, C)$ for shot j, and N_2 pairs for shot $j+1$.
3. Compute the KL2 distance between any pair of speech segments, with each taken from one shot, respectively. Finally, we calculate the minimum distance *mDist* as

$$mDist = \min_{1\le i\le N_1, 1\le j\le N_2} [KL2((\vec{E}_i, C_i), (\vec{E}_j, C_j))], \tag{8}$$

where

$$KL2((\vec{E}_i, C_i), (\vec{E}_j, C_j)) = \frac{1}{2}\operatorname{tr}[(C_i - C_j)(C_j^{-1} - C_i^{-1})] + \frac{1}{2}\operatorname{tr}[(C_i^{-1} + C_j^{-1})(\vec{E}_i - \vec{E}_j)(\vec{E}_i - \vec{E}_j)^T], \tag{9}$$

which is the vector version of Equation 7. When *mDist* is less than the threshold *spbgT*, it is very possible that the corresponding two speech segments are from the same person, we thus merge scenes k and $k + 1$ together.

Here is one example scene of this particular scenario taken from a test movie. Particularly, it has two shot. The first shot shows a landscape of snow-capped mountains, while the second one is about a woman walking in a forest with a horse. Clearly, these two shots cannot be merged if only the visual cue is considered. Moreover, as the background music suddenly starts in the second shot, we should not merge them either based on the rules derived before. However, by noticing that there is a continuous speech flow of an off-screen voice throughout these two shots, we know that they are actually semantically related, or at least they are purposefully correlated by the director, thus they should belong to the same scene.

Scenario IV: A special transition effect exists across a shot boundary:

- Rule 5: If a special effect is detected across a shot boundary, we declare a scene break.

Special effects such as dissolve, wipe, fade-in, and fade-out are usually used to convey passages of time, changes of the place, or any other scenes of transition (Reisz and Millar, 1968). Thus, a scene break is declared in this case.

Scenario V: One or both scenes are isolated scenes:

- Rule 6: If the two scenes are visually similar, we merge them together.

Because a scene normally contains more than one shot, the purpose of this rule is to correlate isolated scenes to their neighbors. Specifically, assume scene $k + 1$ is an isolated scene, the similarity between scenes k and $k + 1$ is measured as follows.

1. Assume scene k contains shot 1 to shot j, and scene $k + 1$ contains shot $j + 1$, we compute the minimum distance between these two scenes as

$$Sdist_{k,k+1} = \min_{1 \le i \le j}[Dist_{i,j+1}], \tag{10}$$

where the distance $Dist_{i,j+1}$ between shots i and $j + 1$ is computed using Equation 1 with the 4 weighting coefficients set to be equal.

2. Distance $Sdist_{k,k+1}$ is then normalized as described in Section 3.1.1 and compared with threshold $shotT'$. If it is smaller, it means that parts of these two scenes are visually similar, thus we could merge them together.

Table I. Scene extraction results obtained with the pure visual cue.

Video	Duration (min)	Hits	Misses	False Alarms	Precision (%)	Recall (%)
Sitcom 1	20	12	0	2	86	100
Sitcom 2	28	13	0	3	81	100
Movie 1	58	34	0	37	48	100
Movie 2	61	38	1	68	36	97
Movie 3	60	40	1	50	44	98

Table II. Scene extraction results obtained with both audio and visual cues.

Video	Duration (min)	Hits	Misses	False Alarms	Precision (%)	Recall (%)
Sitcom 1	20	12	0	0	100	100
Sitcom 2	28	13	0	1	93	100
Movie 1	58	34	0	1	97	100
Movie 2	61	38	1	3	93	97
Movie 3	60	40	1	6	87	98

Note that the above process demands a lot of computations, thus it could be omitted if the speed is a concern.

3.3. User Interaction.

After obtaining the refined scene results via mergence of scenes detected in the coarse level, we could go one step further, i.e., to let the user tune the results to their own satisfaction. This is a highly desirable feature because "scene" is a subjectively defined concept, and different people may have different opinions of what comprises a scene. For example, in one of our experiments, we find that there are scenarios where the speech flow continues over two consecutive shots while the visual background has changed. In this case, on one hand, because the background has changed, a scene boundary should be declared. But on the other hand, because a continuous speech flow is detected, the two shots should be merged together according to Rule 2 or 4. Apparently, the information collected from the audio and visual cues has resulted in conflicting conclusions. One solution to this problem is that we shall only stick to one cue, i.e., we either follow the visual hint and ignore the audio cue, or vice versa. The second solution is to let the user make the decision by interacting with the system. Note that if a completely automatic scene detector is preferred, we may choose the first solution.

4. EXPERIMENTAL RESULTS

The proposed scene detection scheme has been studied using five test sequences with two TV sitcoms and three feature films. All of them are stored in MPEG-1 format with a frame rate of 29.97 frames/sec. To validate the effectiveness of the proposed algorithms, representatives of various movie genres are selected. Specifically, Movie 1 is a thrilling, tragic romance, Movie 2 is a comedic drama, and Movie 3 is an action movie. Totally there are 2025 shots in these test programs.

Tables I and II give the scene extraction results for the above five test sequences. Precision and recall rates are computed to evaluate the system performance, where

$$\text{precision} = \text{hits}/(\text{hits} + \text{false alarms}),$$
$$\text{recall} = \text{hits}/(\text{hits} + \text{misses}). \tag{11}$$

Particularly, results in Table I are the coarse-level results obtained using pure visual information, whereas results in Table II are refined ones achieved by integrating the audio and visual cues. The ground truth is manually collected by the first author of this article, and in case there are conflicting conclusions from audio and visual cues, a higher priority is given to the audio cue because human involvement cannot be quantitatively experimented. In fact, we found in experiments that the audio cue can generally bring more meaningful results than the visual cue. Below we give some discussions on the obtained results.

- Overall, very satisfactory precision and recall rates have been achieved when both audio and visual cues were utilized. Moreover, by comparing the two tables, we see that the average precision rate has been increased by almost 35% when we applied the scene refinement process, which is very impressive.
- We see from Table II that the precision rate is slightly lower than the recall rate because some false alarms still cannot be completely removed. For instance, in Movie 2 there were three false alarms, with each corresponding to an isolated football scene. In particular, the first scene is a close-up view of the ball, the second contains cheering audiences, and the last gives a snapshot of football players. Apparently, none of them share similar visual or audio content. As a result, no previous rules could be applied to merge these scenes. In fact, they could only be merged based on a real semantic understanding, *e.g.* we humans know that these are all typical scenes related to a football game, thus they should be merged together. However, with our current technology, the real semantic understanding still remains to be a very difficult problem. Therefore, to let the user interact with the system may be the only feasible solution.
- Our system performs slightly better in "slow" movies than in "fast" movies. There are two reasons for this observation. First, some speech or music shots cannot be correctly detected because of the loud background sounds in action movies. This undoubtedly brings negative effects to our detection scheme. Second, in "fast" movies, visual contents are normally more complex and thus more difficult to capture. In fact, we even had different scene segmentation results from different people who were invited to watch the movie and gave their respective scene understanding.
- The missed detections were mainly caused by the errors introduced during the shot sink generation, where visually similar shots, which actually belong to different scenes, were mistakenly linked together. Moreover, despite the high recall rate we obtained, some scene boundaries were not precisely located, which was also caused by the sink error.
- Compared to the 2025 shots in the original test sequences, only 137 scenes are detected from them now, which gives us an almost 14-orders of magnitude reduction. Apparently, the video content can be represented more compactly and more meaningfully in the scene structure than in shot structure.

As for the system execution time, currently it took our system, which runs on a Dell-PC with Pentium II processor, approximately 8 minutes to generate the final results for a 1-hour-long movie. However, if we also consider the time used for audio and visual content preanalysis, including the shot detection and audio classification, it may take about 40 minutes, which is still faster than real time.

A system with a friendly GUI has been developed for the proposed scene extraction scheme using MFC and DirectShow technology. When a movie clip is opened, all of its detected scenes will be displayed, together with their respective keyframes. Besides, all shots contained in each selected scene are also displayed simultaneously. The purpose of showing keyframes is to give users a glimpse of the scene content. Nevertheless, when more detailed information is needed, an active movie player can be initialized to playback the corresponding video segment. Finally, when keyword extraction results are available, they can also be displayed so as to better assist users in the content understanding.

5. CONCLUSION

The scene structure forms a compact yet meaningful abstraction of the video data, and thus it can be used to facilitate efficient video indexing and browsing. This work presented a content-based video scene extraction scheme, which aims at detecting semantically meaningful scenes from feature films. Multiple media modalities including audio and visual cues were exploited, and special film editing techniques were also investigated to achieve a better system performance. Although currently movie is our major focus, the methodology presented in this article could be easily extended to other types of generic video. As for our future work, we plan to incorporate other image/video processing techniques, such as human face detection/recognition and object tracking, into the system.

REFERENCES

H.M. Beigi, S.H. Maes, and J.S. Sorensen, A distance measure between collections of distributions and its application to speaker recognition, ICASSP'98, Seattle, WA, May 1998.

T.M. Cover and J.A. Thomas, Elements of information theory, John Wiley and Sons: New York, 1991.

M. Flickner, H. Sawhney, and W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, Query by image and video content: The QBIC system, IEEE Computer Magazine 28 (1995), 23–32.

R. Hariharan, J. Hakkinen, and K. Laurila, Robust end-of-utterance detection for real-time speech recognition applications, ICASSP'01, 2001.

H. Jiang, T. Lin, and H.J. Zhang. 2000. Video segmentation with the assistance of audio content analysis, ICME'00, New York.

Q. Li, J. Zheng, Q. Zhou, and C. Lee, A robust, real-time endpoint detector with energy normalization for ASR in adverse environments, ICASSP'01, May 2001.

Y. Li and C.-C.J.Kuo, Real-time video segmentation and annotation of MPEG video based on multimodal content analysis I and II, Technical Report, University of Southern California, July 2000.

Y. Li, W. Ming, and C.-C.J. Kuo, Semantic video content abstraction based on multiple cues, ICME'01, Tokyo, August 2001.

T. Lin, H.J. Zhang, and Q.Y. Shi, Video scene extraction by force competition. ICME'01, Tokyo, August 2001.

MPEG Requirements Group, MPEG-7 applications document v.8. Document of ISO/MPEG N2860, MPEG Vancouver Meeting, July 1999.

J. Nam, A. Cetin, and A.H. Tewfik, Speaker identification and video analysis for hierarchical video shot classification, ICIP'97, October 1997.

L. Rabiner and R. Schafer, Digital processing of speech signal, Prentice-Hall Inc.: Upper Saddle River, NJ, 1978.

K. Reisz and G. Millar, The technique of film editing, Hastings House: New York, 1968.

D.A. Reynolds and R.C. Rose, Robust text-independent speaker identification using Gaussian Mixture speaker models, IEEE Trans Speech Audio Process 3 (1995), 72–83.

Y. Rui, T.S. Huang, and S. Mehrotra, Constructing table-of-content for video, ACM J Multimedia Syst 7 (1998), 359–368.

M.A. Siegler, U. Jain, B. Raj, and R.M. Stern, Automatic segmentation, classification, and clustering of broadcast news, Proceedings of Speech Recognition Workshop, Chantilly, February 1997.

A. Tarkovsky, Sculpting in time—reflections on the cinema, University of Texas Press: Austin: 1986.

W.D. Wactlar, T. Kanade, M.A. Smith, and S.M. Stevens, Intelligent access to digital video: Informedia project, IEEE Computer Magazine 29 (1996), 46–52.

B.L. Yeo and B. Liu, Rapid scene analysis on compressed video, IEEE Trans Circuits Systems Video Technol 5 (1995), 533–544.

M. Yeung, B.L. Yeo, and B. Liu, Extracting story units from long programs for video browsing and navigation, IEEE Proc Multimedia (1996), 296–305.

A. Yoshitaka and M. Miyake, Scene detection by audio-visual features, ICME'01, Tokyo, August 2001.

H.J. Zhang, A. Kankanhalli, and S.W. Smoliar, Automatic partitioning of full-motion video, Multimedia Systems 1 (1993), 10–28.

H.J. Zhang, C.Y. Low, S.W. Smoliar, and J.H. Wu, Video parsing, retrieval and browsing: An integrated and content-based solution, ACM Multimedia'95, November 1995.

T. Zhang and C.-C.J. Kuo, Audio content analysis for on-line audiovisual data segmentation, IEEE Trans Speech Audio Processing 9 (2001), 441–457.