Video Shot Detection Combining Multiple Visual Features

Di Zhong and Shih-Fu Chang

Department of Electrical Engineering, Columbia University

December 27, 2000

1 Overview

Shot based indexing techniques have been widely used to organize video data. Scene change detection is the most commonly used method to segment image sequences into coherent units for video indexing. A shot is a sequence of contiguous frames that are recorded from a camera. There is usually one continuous action within a shot, with no major change of scene content. However, there are still many different changes in a video (e.g. object motion, lighting change and camera motion), it is a nontrivial task to accurately detect scene changes. Furthermore, the cinematic techniques used between scenes, such as dissolves, fades and wipes, produce gradual scene changes that are harder to detect.

Scene cut detection algorithms have been studied since the early 90's. The basic method is to measure the pixel difference frame-to-frame in terms of intensity or color [8]. In [8], the number of changed pixels is counted and if the number exceeds a certain percentage, a scene cut is detected. This method is not robust due to the camera and object motions that can cause large pixel value differences.

Color histograms have been used to overcome the problem, as color distributions in successive frames are not significantly affected by camera or object motions. Assume H_i is an N-bin color

histogram extracted from frame *i*, the frame difference is defined as :

$$D_i = \sum_{j=1}^{n} \left| H_i(j) - H_{i+1}(j) \right| \tag{1}$$

If D_i is larger than a given threshold, a scene cut is detected at the frame *i+1*. A more efficient distance measure, χ^2-test, is proposed in [5], and shown to have better performance in experiments compared to other measures. In the χ^2-test, the distance between two color histograms H_i and H_{i+1} is defined as:

$$\chi_i^2 = \sum_{j=1}^{n} \begin{cases} \dfrac{(H_i(j) - H_{i+1}(j))^2}{\max(H_i(j), H_{i+1}(j))} & \text{if } H_i(j) \neq 0 \text{ or } H_{i+1}(j) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Although direct comparison of frame-to-frame color difference is good for direct scene changes, gradual transitions such as fade-in, fade-out, dissolve and wipe cannot be accurately detected in the same way. As shown in **Figure 1**, the frame-to-frame color differences within a gradual scene change are much smaller than that of a direct scene cut. In the meanwhile, gradual transitions last much longer (more than 1 second) compared to direct scene changes.

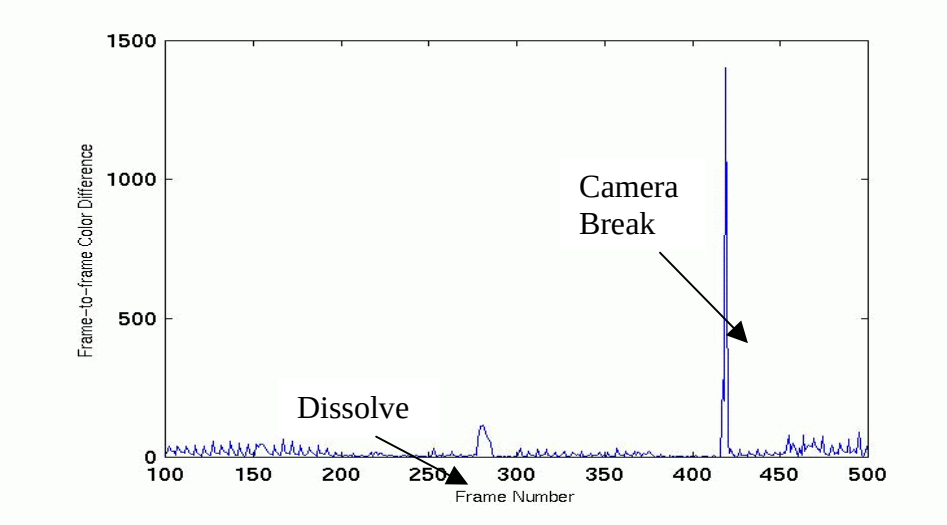


Figure 1 Comparison of direct scene cut and gradual

Because of the low difference values of gradual scene changes, a single threshold cannot distinguish them from camera or object motions. To resolve this problem, a twin-comparison algorithm is developed in [8]. This method requires two cutoff thresholds, one higher threshold for direct changes and a lower one for gradual transitions. The higher threshold is applied first. If there is no direct scene cut, the lower threshold is then used to detect potential transitions. Once a candidate transition is detected, frame-to-frame differences are accumulated for successive frames. If the accumulated distance is larger than the higher threshold, a gradual transition is declared. Note that this is based on the assumption that transitions last over a certain period, and frame-to-frame differences within the transition period do not drop below the lower threshold. In [7], an edge based approach is proposed to detect direct scene cuts and gradual transitions at the same time. It computes the percentages of edges that enter and exit between two frames. Shot boundaries are detected when the percentage is over a given value. Dissolves and fades are detected by comparing the enter and exit percentages.

Scene cut detection often includes feature extraction and comparison for suucessive pairs of successive frames. It is a time consuming process that cannot be done in real time on a regular PC or workstation. As most digital videos are compressed in MPEG, detecting scene changes directly in the compressed domain has been studied to accomplish real time performance. In [3], statistics of motion vectors are used to detect scene cut. For a P-frame, the ratio of the number of intra-coded blocks and the number of inter-coded blocks is computed. For a B-frame, the ratio of the number of backward motion vectors and the number of forward vector is computed. High ratio values indicate shot changes at P- or B-frames. On the other hand, a low ratio on a B frame indicates that there is a scene cut at its preceding I-frame (in transmission order).

In this chapter, we present a new scene cut detection scheme which combines multiple visual

features in both the compressed and uncompressed domains. The main contribution of our work includes:

- An effective scheme to combine motion and color features in both the compressed and uncompressed domains.
- New algorithms to detect gradual transitions, as well as flash lights, lighting changes and camera motions.
- A multi-level scene change detection for browsing and correction.
- Demonstrated real time performance with high accuracy
- Extensive experiments on large amount of various types of videos

2 A New Scheme Combining Multiple Visual Features

Although abrupt shot boundaries have been well studied in existing works, robust detection of gradual transitions is still a challenging issue. In [1], a comparison of many existing scene cut detection algorithms is conducted. Around 90 percent accuracy is reported for direct cut detection. For gradual scene changes, the accuracy is in the range of 70 to 80 percent. While building a real-time video parsing and analysis framework that can be applied to live videos, we also met some other challenging issues. First, the real-time requirement usually conflicts with high detection accuracy. More complicated visual features, which are used to obtain accurate scene cuts, can hardly be computed in real time. Furthermore, lighting changes, e.g., flashlights that occur often in home videos, raise another problem that causes false detection results.

To address these issues and further improve scene detection accuracy, we developed a multi-statge scene cut detection scheme that combines motion, color and edge information. Although various visual features and comparison methods have been studied, it is agreed that no single

feature works better in all situations [1]. How to effectively combine visual features to obtain robust scene cuts is an open issue. We use machine-learning techniques (i.e. decision tree) to find combined measure metrics and thresholds. Compressed-domain features are applied before those in the un-compressed domain in order to accomplish real-time performance. We also developed special modules to detect flashlights and lighting changes.

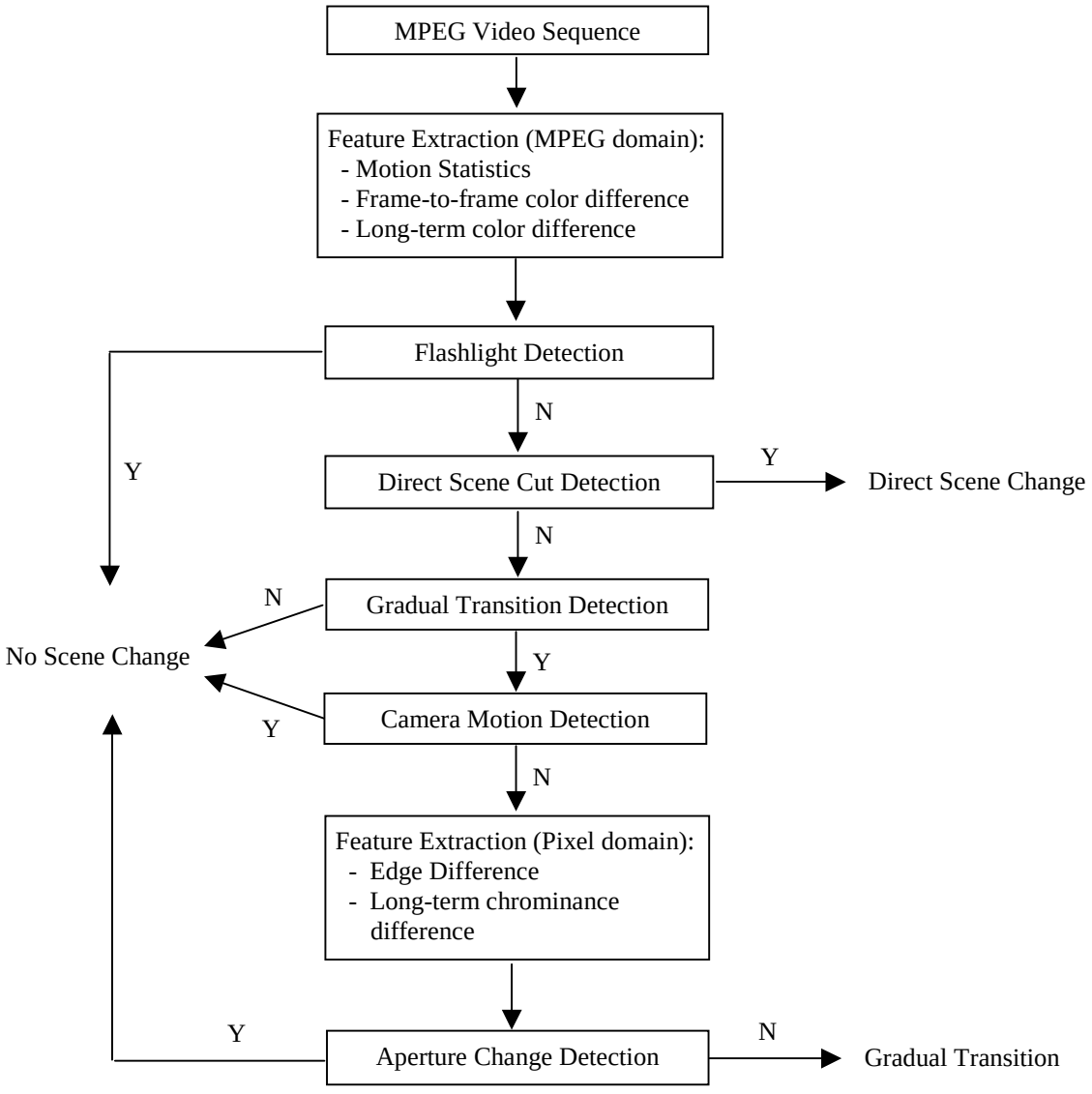


Figure 2 The scene cut detection schema that combines multiple visual features

The diagram of the schema is shown in **Figure 2**. An overview of the scheme is given below.

Detailed algorithms will be described in the following sections.

Compress domain features are first extracted. Motion statistics are computed from different types of motion vectors in P or B frames. Color differences are extracted from the DC images of I- or P- frames. It only requires partial decoding without inverse DCT to extract these features, and thus the process is faster than real-time even on a regular Pentium II 300 PC. The next module detects possible flashlights. It is based on the comparison of the frame-to-frame color difference and the long term color difference. If a flashlight is detected at a frame, no scene cut detection will be performed on this frame. Otherwise, scene cut detection algorithms are applied.

Because direct scene changes can be detected accurately, we first check if there is a direct cut at the frame. If no cut is detected, we then check if there is a gradual transition by detecting starting and ending edges of transitions. As gradual scene transitions are opted to be confused with camera motions or aperture changes, when a potential gradual transition is found, we first check camera motions based on motion vectors in the MPEG compressed domain. If there is no camera motion, we extract edge and chrominance features from de-compressed frames, and compute the differences to examine whether the difference is caused by aperture changes. In case of aperture changes, scene structure and chrominance are not affected as much as brightness. A gradual transition is declared only if image edges or chrominance are significantly changed.

In the following sections, we will give a detailed explaination of the above modules and algorithms in the order that they are applied.

3 Feature Extraction in the MPEG Compressed Domain

As digital video sequences are usually compressed for storage and transmission, it has been proposed in some works to detect scene cuts directly in the compressed domain without fully

decoding the bitstream. Here we focus on the most popular MPEG-1 or MPEG-2 compressed videos. We adopt the motion statistic features proposed in [3], and define two more color statistic features that are also extracted in the DCT-based compressed domain. As there are three different types of compressed frames [2], different feature extraction methods are used.

MPEG video sequences are composed of three types of frames, i.e., I, P and B. An I-frame is completely intra-coded without motion prediction. A P-frame is inter-coded based on motion prediction errors from its past I- or P- frame. A B-frame is coded based on bi-directional motion prediction from its past and later I- or P- frames. I- and P- frames are also referred as anchor frames.

For an I-type frame, the frame-to-frame and long-term color differences are computed. The color difference between two frames *i* and *j* is computed in the YUV space, and is defined as follows.

$$D(i,j) = \left|\overline{Y}_i - \overline{Y}_j\right| + \left|\sigma_Y^i - \sigma_Y^j\right| + w*(\left|\overline{U}_i - \overline{U}_j\right| + \left|\sigma_U^i - \sigma_U^j\right| + \left|\overline{V}_i - \overline{V}_j\right| + \left|\sigma_V^i - \sigma_V^j\right|) \quad (3)$$

where $\overline{Y},\overline{U},\overline{V}$ are the average Y, U and V values computed from the DC images of the frame *i* and *j*; $\sigma_Y,\sigma_U,\sigma_V$ are the corresponding standard deviations of the Y, U and V channels; *w* is the weight on chrominance channels U and V. The frame-to-frame color difference is computed between the I-frame and its previous P-frame (Eq 4). Note that the DC image of a P-frame is interpolated from its previous I- or P- frame based on the forward motion vectors.

$$D_{fram-to-frame}(i) = D(i, i-M-1) \quad (4)$$

where M is the number of B-frames between a pair of successive anchor frames.

The long-term color difference is computed between the I-frame and its *k*th previous P or I frame, which is :

$$D_{long-term}(i) = D(i, i-(M+1)*k) \quad (5)$$

where $k>1$ and is usually set to the range from 5 to 10, which corresponding to a 0.2 second to 0.4 second time interval for typical MPEG videos.

For a P-type frame, the computation of frame-to-frame and long-term color differences is the same as an I-type frame. Color statistics are extracted from interpolated DC images. In addition, the motion measure *Rp* is computed. *Rp* is the ratio of intra coded blocks to forward motion vectors in the P-frame (detail can be found in [3]). Here forward motion vectors in a P-frame refer to motion estimation from its former I or P frame.

For a B-type frame, we only compute two motion-based measures, *Rf* and *Rb*. *Rf* is the ratio between forward and backward motion vectors in the B-frame. *Rb* is the ratio between backward and forward motion vectors in the B-frame (again, detail can be found in [3]). Note that backward motion vectors in a B-frame refer to motion estimation from its next anchor frame, while forward motion vectors refer to motion estimation from its former anchor frame, all in the display order.

4 Flashlights Detection

Flashlights occur frequently in home videos (e.g. ceremonies) and news programs (e.g. news conferences). They cause abrupt brightness changes of a scene and will be detected as false scene changes if not handled properly. We apply a flash detection module before the scene change detection process. If a flashlight is detected, the scene cut detection is skipped for the flashing period. As we will demonstrate, when a scene cut happens at the same time of a flashlight, our algorithm will not detect the flashlight and can still detect the scene cut correctly.

Flashlights usually last less than 0.02 second. Thus for normal videos with 25 to 30 frames per second, one flashlight will affect at most one frame. An example of flashlights is show in **Figure 3**. As we can see, the affected frame has a very high brightness.

a. the frame before a flashlight

b. the frame of a flashlight

Figure 3 The effect of flashlights on a scene (video shot by Prof. Shih-Fu Chang)

Based on our observation of home videos, a flashlight causes the following changes in a recorded video sequence. First, it may generate a bright frame. Note that because the frame interval is longer than the time of flashlights, a flashlight does not always generate the bright frame. Secondly, a flashlight often causes the aperture change of a video camera, and generates a few dark frames in the sequence right after the flashlight. The average intensities over the flashlight period in the above example are shown in **Figure 4**.

As shown in **Figure 4**, the intensity jumps to a high level at the frame where the flashlight occurs. The intensity goes back to normal after a few frames (e.g., 4 to 8 frames) due to aperture change of video cameras. On the contrary, for a real scene cut, the intensity (or color) distribution will not go back to the original level. Based on this feature, we use the ratio of the frame-to-frame color difference and the long-term color differences, to detect the flashes. The ratio is defined as follows.

$$Fr(i) = D(i, i-1) / D(i+\delta, i-1) \tag{6}$$

where i is the current frame, and δ is the average length of aperture change of a video camera (e.g.

5). If the ratio $Fr(i)$ is higher than a given threshold (e.g. 2), a flashlight is detected at the frame *i*. Obviously, if we use the long term color difference at frame $i+\delta$ to detect flashlight at frame *i*, this will become a non-causal system. In actual implementation, we need to introduce a latency not less than δ in the detection process. Also, in order to determine the threshold value, we use a local window centered at the frame being examined to adaptively set thresholds.

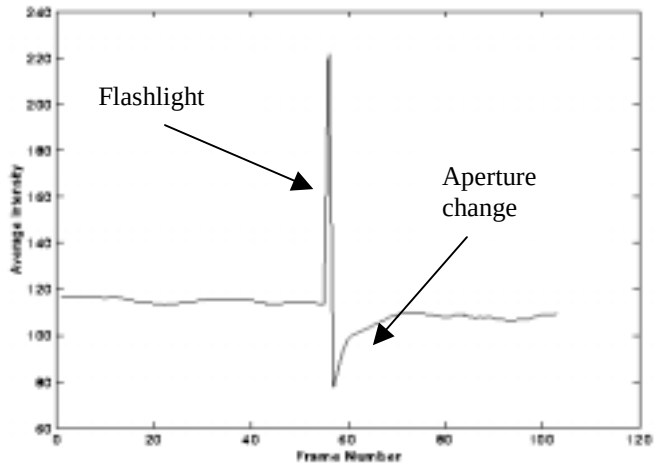


Figure 4 Typical intensity changes in a video sequence due to a flashlight

Note that the above flash detection algorithm only applies to I- and P- frames, as we do not extract color features at B frames. However, a flashlight occurring at a B-type frame (i.e. bi-direction projected frame) does not cause any problem in the scene cut detection algorithm we adopted and modified in [3]. This is because a flashed frame is almost equally different from its former and successive frames, and thus forward and backward motion vectors are equally affected [2].

In occasions where a scene cut happens at or right after a flashlight, the flashlight will not be detected because the long-term color difference is also large due to the scene cut. As our goal is to detect actual scene cuts, misses of flashlights are acceptable. Furthermore, as we will discuss in the next section, our algorithms are able to pick the right scene cut position under this situation.

5 Scene Cut Detection

Given the color and motion measures of the frame-to-frame differences, the scene cuts can be detected by identifying peak values of these measures. As scene changes in videos from different sources usually have different characteristics, it is hard to set a global threshold that can detect peak values in different videos. Even within the same video (e.g., a news program), different parts may have different levels of peak values. To solve the problem, we use a local window to detect peak values. The size of the window is usually 30 to 60 frames, and is centered at the frame that is being examined for scene cuts.

Assume the size of window is $2*\delta+1$, feature values for each frame are divided by their corresponding average values over the window $[i-\delta, i+\delta]$. These new peak-to-average ratios (*PA*) are defined as follows.

$$PA_T(i) = \frac{T(i)}{\sum_{k=i-\delta}^{i+\delta} T(k) \Big/ (2*\delta+1)}, \quad \text{where } T \in (D_{frame-to-frame}, R_p, R_b, R_f) \tag{7}$$

Note that because different frame types have different features, the sum in the above equation is conducted only over the frames where the corresponding feature is available. For an I frame, *PA* of $D_{frame-to-frame}$ is computed. For a P frame, *PA's* of $D_{frame-to-frame}$ and R_p are computed. For a B frame, *PA's* of R_b and R_f are computed.

Given all these *PA* ratios within a local window, it is not trivial to combine them in a single decision process that detects scene cut on a given frame. One approach is to try different combinations manually, and then compare their performances to find the most appropriate model. Considering there are many thresholds involved in various combinations, such manual selection

process are complex and time-consuming. In this work, we adopt a decision tree based learning process to find proper decision models and approximated thresholds.

5.1 Combining Color and Motion Features Using Decision Tree

Decision tree is a popular, simple machine learning technique. It involves a tree in which a non-leaf node is labeled with a feature. The branches at the non-leaf node correspond to the possible values or ranges of the feature. As an example in **Figure 5**, the feature at the top level is frame-to-frame color difference, the branches below the node are the possible value ranges of the feature, e.g., more than 100 and less or equal 100. Leaf nodes are labeled with a class, i.e. scene cut or no scene cut. Decision trees are used for classifying instances - one starts at the root of the tree, then, taking appropriate branches according to the feature at each branch node, and eventually comes to a leaf node. The label on that leaf node is the class for that instance.

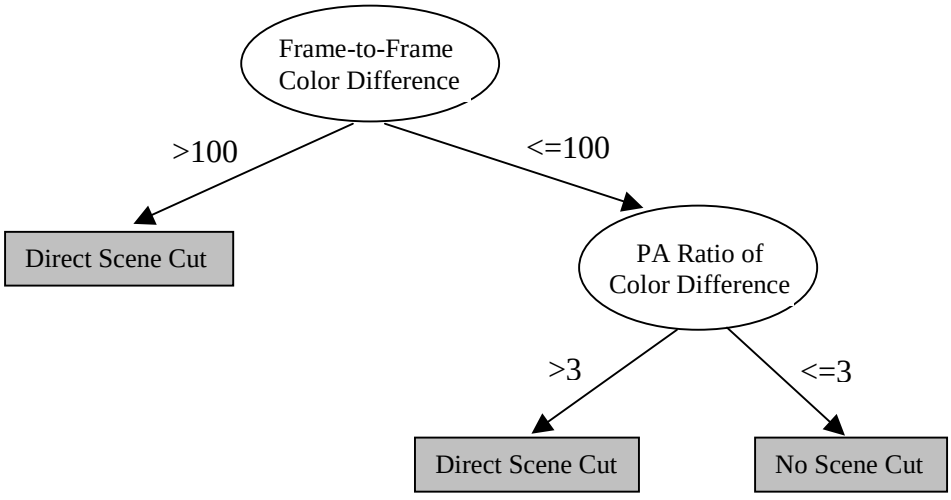


Figure 5 An example of decision tree

Similar to other machine learning techniques, feature selection is the key to success in developing decision tree techniques. Simply putting all measures together without identifying

proper features will result in ineffective solutions. Based on the characteristics of compressed data, we choose to build scene cut decision trees for I, P and B frames separately. This is because different types of frames have different characteristics as well as features. We expect that decision rules are different for different types of frames. Furthermore, direct scene cuts and gradual transitions are also handled separately because their detection models are different. Gradual transitions last longer than direct scene cuts, and will be detected only based on color differences. Thus we need to build and train four different decision trees: three for direct scene cut detection and one for gradual transition detection.

Our training videos include baseball, news, sitcom and home videos. We manually labeled each scene cut and its corresponding frame type. We use the public domain induction tool OC1 (Oblique Classifier 1 [4]) to build our scene cut classifier. Oblique decision tree methods are tuned especially for domains in which attributes are numeric. After the training, we manually prune and merge deep level nodes in the output trees to obtain simplified final decision models. These models are discussed in the following sections, i.e., 5.2 to 5.4.

5.2 Direct Scene Cut Detection

Direct scene cuts are detected at all three types of frames. As we mentioned before, detection of direct scene cuts is relatively easy. If we detect a peak difference within a local window, it indicates a scene cut. Here we examine the PA ratios to find peak values. The decision tree is trained to learn the order of the color and motion features to be compared in the detection process.

For the *k*th frame, if it is an I frame, we use the following PA ratios : $PA_{D_{frame-to-frame}}(k)$, $PA_{D_{frame-to-frame}}(k+M+1)$ and $PA_{Rb}(k+j)$ where *j=1…M*. Here *M* is the number of B frames between the I frame and its next P frame. Note here the frame numbers are in transmission order.

An example of *M=2* is shown in **Figure 6**. Note here frame *k+1* and *k+2* are B frames displaying before the I frame (*k*), but being transmitted afterwards due to the frame prediction procedure used in MPEG.

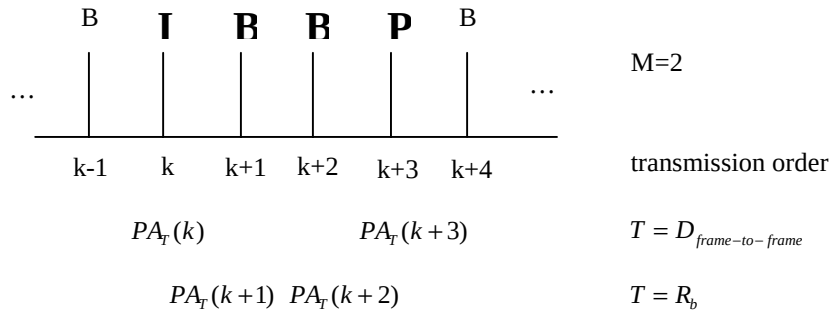


Figure 6 PA ratios used in I frame scene cut detection

In addition to $PA_{D_{frame-to-frame}}(k)$ that is used to check if there is a peak at frame *k*, $PA_{D_{frame-to-frame}}(k+M+1)$ is checked to make sure that there is no peak value at frame *k+M+1* (as we can safely assume there is no two scene cuts with a few frames). This is mainly to handle very fast camera motions that result in large frame differences in consecutive frames (instead of only in one frame for regular direct scene cut). $PA_{Rb}(k+j)$ is checked to see if the scene cut occurs at the frame *k+j* [3], which is displayed before the frame *k*. This is because the frame *k* is compared with its former P frame to obtain the frame-to-frame color difference, and will also have a peak difference when a scene cut actually occurs at one of the B frames before it (in the display order). The decision tree derived is shown in **Figure 7**.

The threshold *TH* to detect peak PA ratios of the frame-to-frame color difference is about 5 to 6, which are obtained from the training process. The *TH_Rb* to detect peak PA ratios of R_b is around 2 to 3. As discussed in [3], a large R_b indicates that there is a direct scene cut at the B frame. The

optimal thresholds are slightly different for different type of videos (e.g., baseball and home video). As we will discussed in the section 5.4, in practice, a general multi-level schema can be used to enable users to easily correct false alarms and misses.

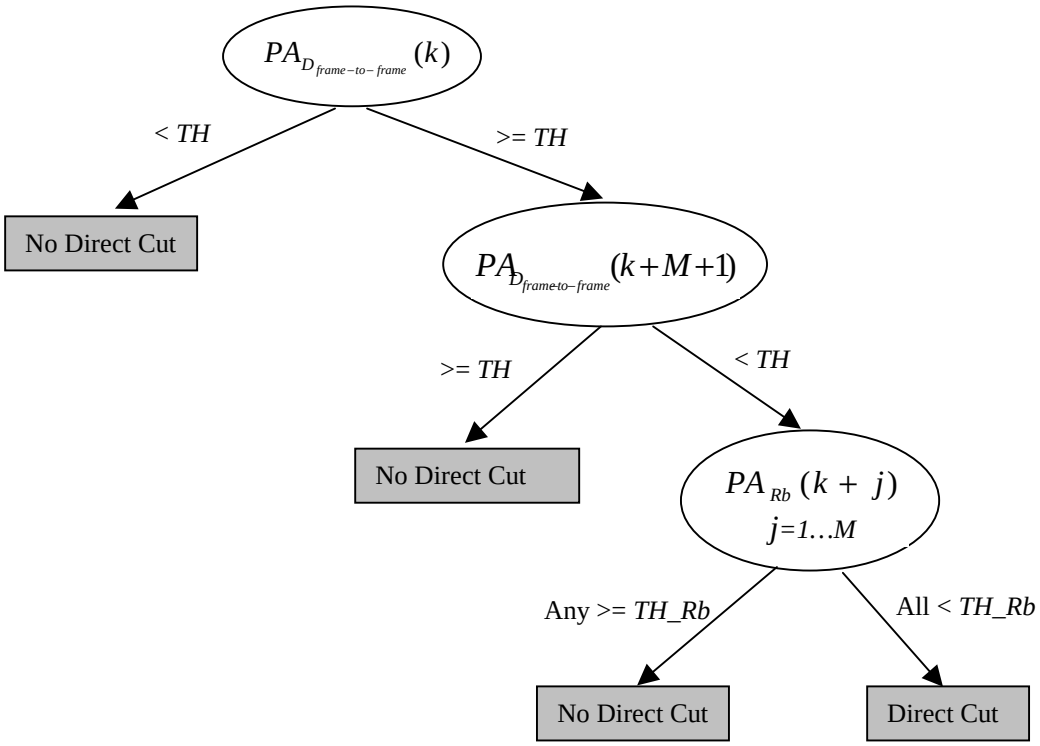


Figure 7 Direct scene cut detection at I-type frames

If the *k*th frame is a P frame, $PA_{Rp}(k)$ is checked in addition to the features that are checked for an I frame, including $PA_{D_{frame-to-frame}}(k)$, $PA_{D_{frame-to-frame}}(k+M+1)$ and $PA_{Rb}(k+j)$ where $j=1...M$. Its decision rules are summarized in **Figure 8**.

Similarly, a peak *Rp* or $D_{frame-to-frame}$ ratio indicates a potential direct scene cut. The thresholds are *TH_Rp* and *TH* for motion and color respectively. In addition, if the two relatively lower thresholds, *TH_Rp1* and *TH1* are both satisfied, a possible scene cut is also declared. This is because the DC image of a P frame is interpolated from its leading I or P frame, and its computed frame-to-frame color difference may be smaller than the actual difference. The same validation

process is followed to make sure there is no peak at the frame *k+M+1*, and there is no scene cuts at its following B frames. Here *TH_Rp* is in the range of 15 to 25.

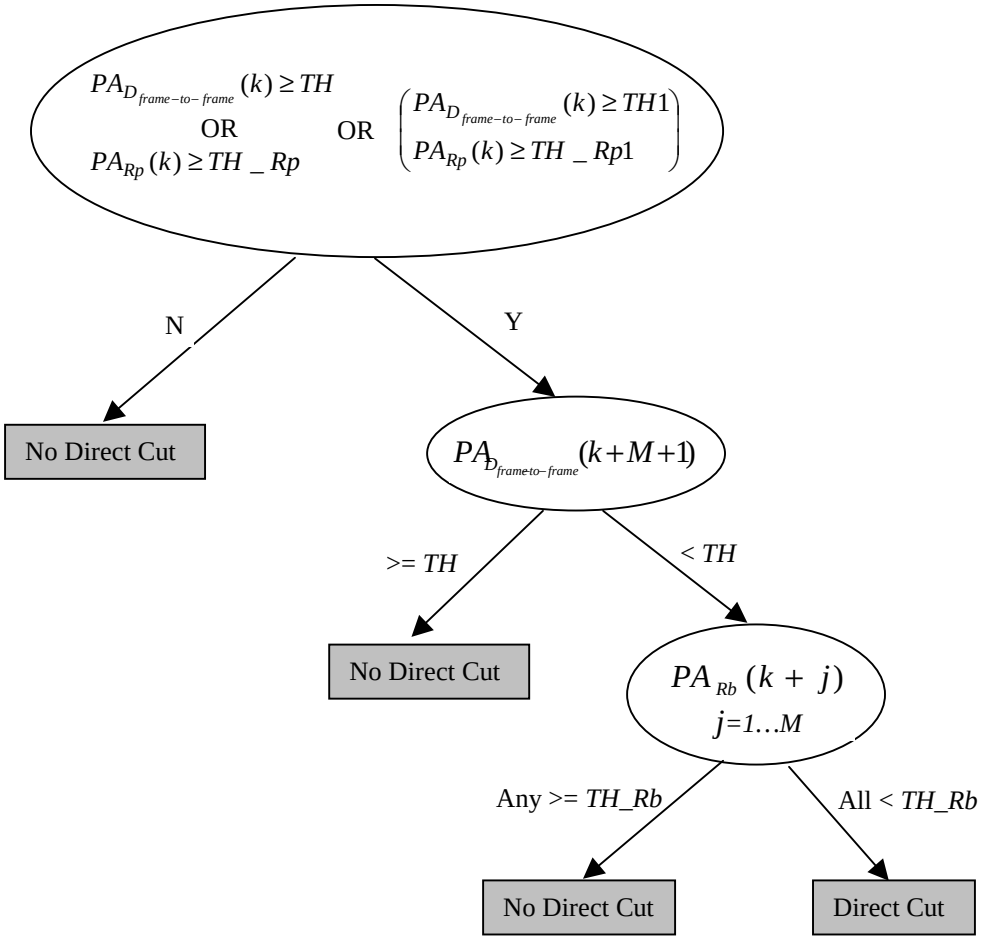


Figure 8 Direct scene cut detection at P-type frames

For a B-frame, we check PA_{Rb} and PA_{Rf} ratios at the frame *k* and its successive B frames. As shown in **Figure 9**, assume *L* is the frame number of the last B frame before the next anchor frame, the *k*th frame is a direct scene cut point if PA_{Rb} value of the frame k and all its following B-frames are larger that the given threshold *TH_Rb* . If the PA_{Rf} values of the frame *k* and all its following B-frames are larger than the threshold *TH_Rf*, its past I or P frame (in transmission order) is

detected as a scene cut. This approach is an enhanced version of what has been proposed in [3]. Instead of only examining one B-frame, the new approach checks more B-frames to improve accuracy and robustness.

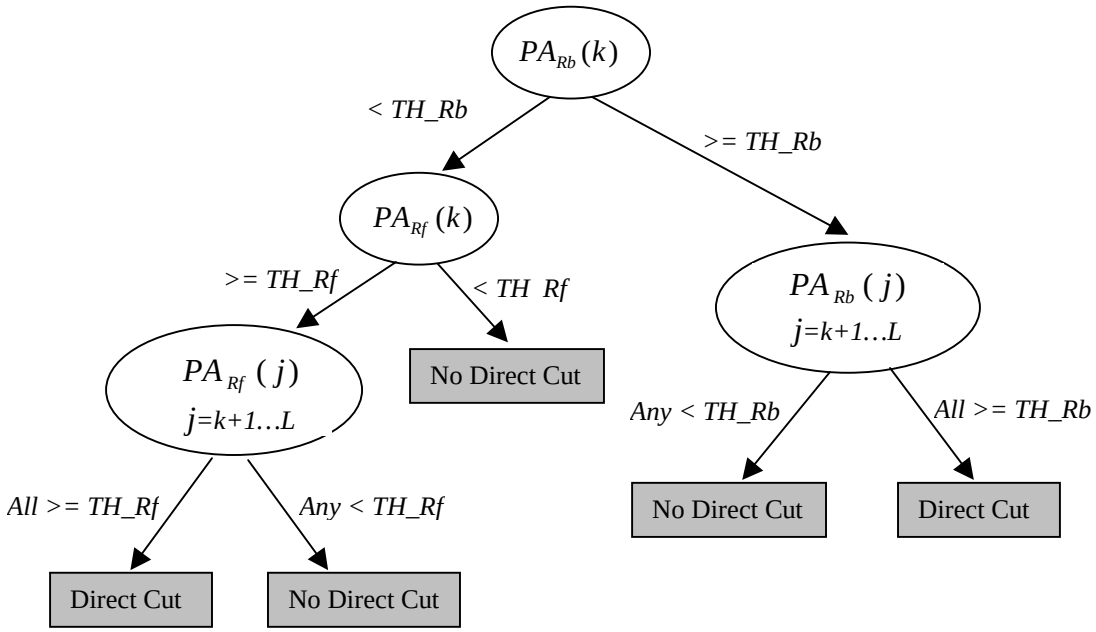


Figure 9 Direct scene cut detection at B-type frames

5.3 Gradual Transition Detection

If no direct scene change is detected, the algorithm checks for gradual transitions that do not show high peak values in above modules. The widely used twin-comparison algorithm is designed to track a transition assuming that frame-to-frame differences do not drop below a threshold within the whole period. However, for long transitions in some videos (e.g. sports or sitcom), differences may drop to a very low level for a few frames within transitions. This will cause misses and/or falses in the twin-comparison method. Other researches have developed algorithms to compute the

intensity variance and then detect parabolic curves to find dissolve or fade related transitions. Due to noise and motion, it is hard to find desired parabolic shapes without introducing many false alarms.

Using the learning method of decision tree, we find it is rather simple and robust to detect the beginning and ending edges of transitions, which have the shape of up and down steps respectively. An example transition is shown in **Figure 10**, b1-b6 and e1-e6 are PA ratios of frame-to-frame color differences computed at anchor frames (Eq 2.4).

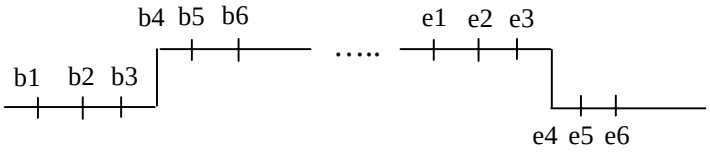


Figure 10 The beginning and ending edges of a gradual scene change

Based on induction results from the decision tree learning, we use color differences at six successive anchor frames to detect the beginning and ending steps (note that we do not compute color measures at B-frames). The final decision rules are summarized as follows.

- If b1, b2 and b3 are smaller than *TH_G1*, and b4, b5 and b6 are larger than *TH_G2*, a beginning edge is detected at the frame of b4
- If e1, e2 and e3 are larger than *TH_G2*, and e4, e5 and e6 are smaller than *TH_G1*, an ending edge is detected at the frame of e4

The above rules are inducted from MPEG-1 sequences with 30 frames per second and two B-frames between each pair of anchor frames. Thus, in general, we need to examine a window of about half second to properly detect steps. From our experiments, the threshold *TH_G1* is around 1.1 to 1.3; the threshold *TH_G2* is from 0.7 to 0.8. As frame-to-frame variances always exist at the boundaries of a transition (which may not be true within a transition), and are usually noticeable (i.e. large), our detection method can pick beginning and ending edges with a high recall rate.

The remaining problem, which also exists in other scene cut detection algorithms, is that fast camera or object motions may produce similar up and down steps, which will cause false positives and therefore reduce precision. To alleviate the problem, we check the distance between a pair of beginning and ending steps. In detail, when a ending step is detected, the distance with the last beginning step, *L*, is computed, and

- If *L>L1 and L<L2*, then there is a transition. Otherwise there is no transition.

Here *L1* is the minimum length of a transition (e.g. 10 frames or 0.3 second). *L2* is the maximum length of a transition (e.g. 60 frames or 2 seconds).

The length constraint removes most false alarms since up and down steps caused by motions typically do not come up in pairs within short periods. One most likely false situation is when there is a sudden camera motion and the motion also stops suddenly in a short time. Slow camera motions usually do not create step-like changes. This is acceptable in many applications because the above camera action quickly changes the recording view and may be considered as a "true' scene cut.

5.4 Multi-Level Scene Cut Detection

A scene cut detection with 100 percent accuracy is not realistic, even though we have used various methods to help us choose effective decision models and threshold values and consider many important issues. In practice, we have noticed that given a good browsing interface (such as the one we will show in Chapter 5), it is easy for users to identify and correct false alarms (assume there are a limit amount of errors). On the other hand, it is hard to correct a missed scene cut without playing and watching the whole video again. Adjusting some decision thresholds to lower values can minimize misses, but it usually causes many false alarms.

To solve this problem, we adopt a multi-level scheme for exploring this tradeoff situation. In this scheme, multiple sets of thresholds are used instead of just the optimized threshold values. Scene cuts are detected at different levels. The multi-level schema is shown in **Figure 11**.

For each frame, the detection process goes from a higher level to a lower level when a scene cut is not detected at the higher level. The process stops whenever a scene cut is detected. Because obviously, scene cuts at one level are also scene cuts at the lower levels, the output of a scene cut includes the frame number as well as its detecting level.

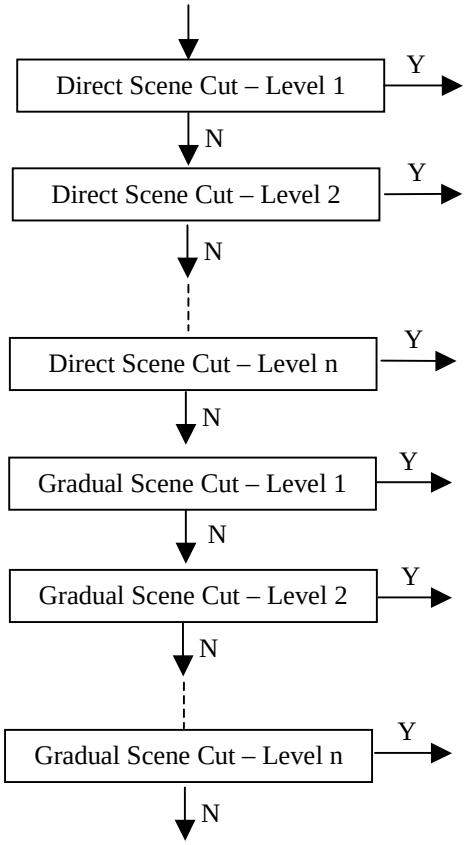


Figure 11 The multi-level scene cut detection scheme

In practice, because the direct scene cut detection is usually accurate, we only apply more than one levels in the gradual scene cut detection stage. Gradual scene changes, such as dissolve and fade, are likely to be confused with fast camera panning/zooming, motion of large objects and light

variance. A high threshold will miss scene transitions, while a low threshold may produce too many false alarms. Our multi-level approach generates a hierarchy of scene cuts. Users can quickly go through the hierarchy to see positive and negative errors at different levels, and correct them.

6 Verification Using Complex Features

As we discussed before, the main problem in the scene cut detection is to distinguish between real scene breaks (especially gradual transitions) and normal changes, such as camera or object motions and lighting changes. Many features and methods have been studied and are proven to improve the detection accuracy. However, most of these approaches require more complicated feature extraction algorithms, such as motion estimation and edge detection. Typically, these algorithms are computation intensive and cannot be performed in real-time on a regular workstation or PC.

Here, we develop a verification mothod to increase detection accuracy without losing the real-time performance. In this scheme, extraction and comparison of more detailed visual features are applied only when a potential gradual transition is detected. Note that this process can also be included in our multi-level scene cut detection scheme discussed in the previous section. Complex models and features are applied only to the candidate frames.

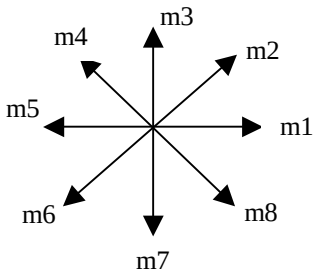
In the following sub-sections, we will present two methods for final verification - camera motion detection and aperture change detection. Typically, these methods require higher resolution and accuracy of extracted features.

6.1 Camera Motion Detection

In this work, we focus on the detection of camera panning operations. Compared to object motion and camera zooming, panning operations usually produce much larger visual content changes (e.g.

color), and thus cause more false alarms. On the other hand, panning can also be detected more reliably compared with other motion activities. The method we utilized here looks for dominant direction of motion vectors as an indication of the camera panning.

We first compute the histogram of eight motion directions for every P-frame based on motion vectors that are available in the MPEG compressed domain (**Figure 12**). This feature extraction process does not require much computation as the number of motion vectors (i.e. macro blocks) is small (330 for a CIF size frame). The detection of panning at P-frames is applied only when a potential gradual scene change is found.



m9 : number of intra coded macro-blocks
m0 : number of macro-blocks with zero motion

Figure 12 Panning detection based on histogram of motion vectors

The algorithm to detect panning in a P-frame is given as follows.

```
Let i be the direction with maximum number of motion vectors (1 to 8)
    j be the direction with second most motion vectors (1 to 8)

if (mi>m9) and (mi>m0) and (mi/mj>2)  then
    panning
else
  no panning
```

We find the dominant motion directions. If there are more motion vectors in the dominant direction than the intra coded blocks as well as the blocks with zero motion, we compare the dominant motion direction with the second dominant one. If the former contains a lot more vectors than the later one does (e.g. twice, as we used in our implementation), a panning is detected.

At the ending edge of a potential gradual transition, the above panning detection method is applied to all the P-frames within the transition period. If pannings are detected in P-frames more than a certain percentage (e.g. 50%), we treat this change as a panning instead of a gradual transition.

Note that there are special edit operations in some movies or sports that result in effect similar to the camera panning. One example is the wipe with a new scene coming in and an old scene going out. In such cases, camera motion detection can be used as an optional module when needed in specific domain.

6.2 Aperture Change Detection

Aperture changes of a video camera happen occasionally in movies, news programs, and especially home videos. Aperture changes are usually caused by changes of lighting condition. For example, when a camera is panned from a bright scene to a dark scene, the aperture will gradually open wider to receive more light. This process causes image intensity changes over a short period in the recorded video (**Figure 13**).

a. panning to dark scene **b**. statrt of aperture change **c**. the changed aperture

Figure 13 An example of aperture change that generates two potential transitions (videos shot by Prof Shih-Fu Chang)

The example in **Figure 13** causes two falsely gradual transitions, one from bright to dark and the

other from dark to bright. To solve this problem, we compare the chrominance and edge features of an I-frame right after a transition (potential) with features of an I-frame before the transition. If the differences are smaller than a threshold, the transition is ignored.

Chrominance histograms and edge direction histograms are computed from the two decoded I-frames. The reason to use I-frames is because an I-frame requires less decoding computation and it does not depend on other frames. The chrominance histogram is calculated in the HSV space using only H (hue) and S (saturation) values. The luminance values are not compared as they are sensitive to lighting changes. We use a 36-bin histogram that has 12 levels of H and 3 levels of S. The color difference is computed using the L1 distance measure.

$$D_{color} = \sum_{k=1}^{36} \left| H_1(k) - H_2(k) \right| \tag{8}$$

Edge histograms are counted at 16 equally divided directions by calculating the gradient of each edge pixel. Here edge pixels are detected using Sober operator. For an edge pixel (i,j), let dx=f(i)-f(i-1) and dy=f(j)-f(j-1), its gradient is arctan(dy/dx). The distance between two edge histograms E1 and E2 is defined as follows.

$$D_{edge} = \sum_{k=1}^{16} \left| E_1(k) - E_2(k) \right| \tag{9}$$

Given D_{color} and D_{edge} between two I-frames at the begin and end of a potential transition, if they are both under given thresholds (e.g. around 0.5 in our implementation), an aperture change is declared and the transition is ignored. This is based on the consideration that edge and chrominance features are less affected by aperture changes. For a real transition, these differences are expected to be larger. The other illumination invariant features can also be used [6].

7 Results and Discussions

We developed a real-time scene cut detection system using the scheme described in previous sections on a Pentium-III 600 PC. The PC has a FutureTel MPEG compression card that can capture and compression live video feeds from VCR or cable. This system allows us to test our scene change detection algorithms on a large amount and wide variety of video sources.

In our experiments, we have used a total of around 5.5 hours videos from different sources. As listed in Table 1, there are two half-hour baseball videos from ESPN and Fox respectively. The two tennis videos are from games at two different places. Home videos are obtained from two different personal owners. There are also half hour videos from sitcom, news, cartoon, movie and trailer. Most of these videos contain commercials. They are recorded to VHS tapes from TV broadcasting channels, and then digitized and compressed to MPEG-1 streams at 30 frames per second and CIF resolution (i.e., 352x240) using the FutureTel MPEG encoding card. The movie file is obtained from a VCD and its original format is MPEG-1. The bit rates of these MPEG videos are from 1.2Mbps to 1.5 Mbps.

home video	2 (two different people)	30x2 minutes

Video Type	**Number of Sequences**	**Length**
Baseball	2 (ESPN and Fox)	30x2 minutes
Tennis	2 (two different games)	30x2 minutes
Sitcom	1 (Senfield)	30 minutes
News	2 (CNN)	30 minutes
Cartoon	1 (animals)	30 minutes
Movie	1 (comedy)	30 minutes
trailer	1 (Hot Shots)	30 minutes

Table 1. Description of the 6 hours of videos in the experiment dataset

The baseball and tennis videos contain fast object and camera motions. In the meanwhile, different scenes in a game have similar backgrounds as one game is played at the same stadium or court. Home videos are usually very jerky. There are non-smooth camera motions and unexpected aperture changes. Sitcom videos include many back and forth camera angle switches. It is challenging to pure color based approaches because many angle changes only slightly change the color distribution of a scene. News programs have been widely studied. A news video often contains various scenes in different stories. Cartoons are a special type of videos that are usually consisted of computer-generated graphics. Here a scene cut means significant changes of graphical objects within a scene. Commercials exist in all types of videos except the movie and trailer. The half hour movie is extracted from a comedy that is originally recorded on a VCD in MPEG-1 format. The trailer is a promotion video provided by Hot Shots & Cool Cuts Inc., a stock footage company. It contains a lot of short unrelated shorts that are sampled from the company's large video collections.

We manually identify the ground truth by using a MPEG player with frame accuracy. In our experiments, the scene cut detection results are compared with the ground truth in terms of precision and recall. Assume N is the ground truth number of scene cuts, M is the number of missed cuts and F is the number of false alarms, the recall and precision are defined as follows :

$$\text{Recall} = \frac{N - M}{N} \tag{10}$$

$$\text{Precision} = \frac{N - M}{N - M + F} \tag{11}$$

These two measures are both important. We certainly do not want to miss any critical scene changes. On the other hand, too many false alarms will compromise the efficiency of indexing and summarization. In practice, as we mentioned before, it is relatively easy for users to manually identify false alarms and improve the precision. Thus in the training stage when the thresholds are selected, we prefer lower threshold values, which tend to give more scene cuts, if recalls are not significantly affected (e.g. reduced less than 5%).

We use two sets of thresholds in the experiments. One is for home videos (i.e. amateur). The other one is for the rest of videos (i.e. professional). Amateur and professional videos have very different characteristics. The former have jerky camera motions compared to smooth camera motions in the professional one. On the other hand, gradual transitions or editions in home videos are simpler and are not as many as those in professional videos. The two sets of thresholds are chosen based on the consideration of these characteristics. At the training stage, we use a small amount of video data (around 1 hour) digitized from sources (including news and home videos) that are different from the above testing videos. Note in most of our detection modules, we do not compare absolute value of a specific measure (e.g., frame to frame difference or motion vector percentage) against a fixed threshold. Instead, we normalized these measures with the local window average and use machine learning tools to automatically choose optimal thresholds. By doing these, we were able to generalize the algorithms and make them applicable to different type of videos.

The detailed experiment results are shown in the following tables (Table 1 to 6). For the baseball videos, we have good recall and precision results. Considering the total length of baseball videos is about 60 minutes, there are about 1.2 false alarms per minute (i.e. per 1800- frames), and about 1 miss every two minutes. Many false alarms (~50%) come from close-up scenes when a

person suddenly appears in front of the camera, which totally changes a scene. The rest of false alarms are mainly from panning and zooming when cameras are following player in the field. Most misses occurred at gradual transitions that are inserted between games and replays. Similar precision and recall results are obtained for tennis. Shots in tennis videos are longer (~ 1.5 times) than those in baseball videos. Camera and object motions in long shots causes most falses, as the large changes are more likely to happen in a long panning or zooming shot. Again, the wiping of company logos as transitions between game and replays accounts for many misses.

Video	# of scene cuts (ground truth)	# of false	# of missed	recall	precision
baseball	891	74	37	96%	92%
tennis	622	50	39	94%	92%
sitcom	461	35	23	95%	93%
news	276	32	7	97%	89%
cartoon	313	42	5	98%	88%
movie	225	18	31	86%	92%
trailer	655	22	18	97%	97%
home video	184	65	9	95%	73%
TOTAL	3627	338	169	95%	91%

Table 2 Detection results of all scene cuts for 8 different types of videos

# of direct cuts (ground truth)	# of missed	Recall
3263	107	97%

Table 3 Total missing number of direct scene cuts

Table 4 Total missing number of gradual transitions

# of transitions (ground truth)	# of missed	recall
364	62	83%

Table 5 Two different reasons that cause false alarms

Total false # in test videos	Camera or object motion	Lighting or Aperture Change
338	266 (79%)	72 (21%)

Table 6 Flashlights detection results of home videos

# of flashlights (ground truth)	# of missed	recall
64	18	72%

Our algorithms perform well on sitcom videos. Many direct scene changes in sitcoms do not have large color differences when two successive shots are taken from the same scene. Motion features are helpful in detecting such changes without bringing in too many false alarms. The precision rate is relatively low for news videos (about 89%). Most false positives are caused by camera motions during the field reporting segments. Although news stories are taken by professionals, the recording situations in the field are often not good, and thus we see some jerky camera motions. The recall rate is very high for cartoon videos, while many computer-simulated camera and object motions tend to be confused with scene changes. The detection results for the trailer video are very good due to the fact that most scene changes are direct scene cuts between un-related video shots. The movie sequence has a lower recall than precision. Dark scenes taken at night account for many missing scene cuts. Also there are more advanced special effects and edits being used in the movie.

Note all the above results are obtained by using the same set of threshold values.

Higher motion thresholds (*TH-Rb, TH-Rf and TH-Rp*) are used for home videos (about 25% higher) in order to tolerate jerky camera motions. As expected, we have a high precision recall, but a low precision rate (72%). Our testing home videos have many long shots (e.g., a few minutes long), in which the camera often changes from one view to another several times. If we compute the number of false alarms over time, it is about 2 falses every minute. Typical camera operations that produce false scene cuts include 1) fast moving from one angle to another; 2) zooming in too close to an object; and 3) following a fast moving object.

As mentioned, direct scene cuts and gradual scene cuts have completely different characteristics. It is appropriate to evaluate their performances separately. When we count the number of missing scene cuts, we also label each change as either direct or gradual. The final results are shown in Table 3 and 4 respectively. For direct scene cuts, we achieved a very high 97% recall rate on average. The recall rate is 83% for gradual scene changes. Our experiments show that to improve the later recall rate to 90% introduces too many false alarms.

For false scene cuts, we classify them into two classes: motion-related and lighting-related. The motion-related refers to both camera and object motions, which often occur at the same time. The lighting-related includes real lighting changes as well as aperture changes. As shown in Table 5, 80% false alarms are motion-related.

Detection of flashlights in one of our home videos is shown in Table 6. The recall rate is 72%. Most misses, which result in false scene cuts, are caused by subsequent aperture changes after flashlights (**Figure 4**). When these changes are longer than the size our detection buffer window (about half second), the ratio Fr in Eq. 6 falls below our threshold and causes miss detection of a flashlight. We can increase the length of local window to improve flashlight detection accuracy.

In summary, we performed extensive tests on our scene cut detection scheme. It achieves the best results for sports and sitcom videos. News and cartoons are slightly less accurate, and the most challenging ones are home videos. This ranking is consistent with the degree of irregularity of camera motions in different types of videos. It indicates that motion is the main issue, and we need advanced methods that can capture motions more accurately. Utilizing better motion estimation techniques and exploiting more complicate features (e.g., region level features) are promising directions, although at the cost of computational complexity.

References

1. Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer, “Performance Characterization of Video-Shot-Change Detection Methods”, IEEE Transactions on Circuits and System for Video Technology,Vol 10, No. 1 Feb 2000
2. ISO/IEC 13818 –2 Committee Draft (MPEG-2)
3. J. Meng, Y. Juan and S.-F. Chang, "Scene Change Detection in a MPEG Compressed Video Sequence, " SPIE Symposium on Electronic Imaging: Science & Technology- Digital Video Compression: Algorithms and Technologies, SPIE vol. 2419, San Jose, Feb. 1995.
4. S.K. Murthy, S. Kasif, and S. Salzberg, “A System for Induction of Oblique Decision Trees”, Journal of Artificial Intelligence Research 2:1 (1994), 1-32.
5. Akio Nagasaka and Yuzuru Tanaka,"Automatic Video Indexing and Full-Video Search for Object Appearances", Visual Database Systems II, Elsevier Science Publishers B.V.1992
6. Daniel Toth, Til Aach, and Volker Metzler, “Illumination-Invariant Change Detection”, Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation, Austin Texas, Apr 2000
7. Ramin Zabih, Justin Miller, Kevin Mai, ”A Feature-Based Algorithm for Detecting and Classifying Scene Breaks”, ACM Multimedia, 1995. pp189-200.
8. H.J.Zhang, C.Y.Low, S.W.Smoliar and J.H.Wu "Video Parsing, Retrieval and Browsing: An Intergrated and Content-Based Solution", Proc.ACM Multimedia'95, San Francisco, Nov 5-9, 1995.