A Cost-Effective and Robust Shot Change Detection and Classification Technique based on Camera Tracking

JungHwan Oh
Kien A. Hua *

Computer Science Program
School of Electrical Engineering & Computer Science
University of Central Florida
Orlando, FL 32816-2362
U. S. A.

E-mail: {oh, kienhua}@cs.ucf.edu

July 24, 2000

Abstract

Shot is considered a good unit for indexing and retrieval in video databases. Existing automatic shot boundary detection techniques require many threshold parameters which are difficult to select. To address this drawback, we present in this paper a content-based approach based on camera tracking. This is accomplished by tracking the background areas of the video. This strategy can detect and identify both abrupt and gradual changes. Our experimental results, based on six categories of 26 videos, indicate that the proposed method is more reliable than existing schemes although it processes 400% less data.

KEYWORDS: Video content analysis, shot boundary detection, shot change classification, video database management systems.

1 Introduction

With the advances in electronic imaging, storage, telecommunications, and data compression techniques, the amount of digital video data has grown enormously in recent years. This brings about the need for video data management systems. Organizing and managing video data, however, is much more complex than managing conventional text data due to the huge volumes of video files and their complicated contents.
A video obtained from various sources is called a video clip. It can last from a few seconds to several hours. For most video applications, video clips are convenient units for data entry. However, since an entire video stream is too coarse as a level of abstraction, it is generally more beneficial to store video as a sequence of shots to facilitate information retrieval. This requirement calls for techniques to segment videos into shots which are defined as a collection of frames recorded from a single camera operation. This process is referred to as shot boundary detection (SBD).

Shot boundary detection has been an area of active research. Many techniques have been developed to automatically detect transitions from one shot to the next. These schemes differ mainly in the way that the inter-frame difference is computed. The difference can be determined by comparing the corresponding pixels of two images [1, 2, 3]. Color or grayscale histograms can also be used [4, 5, 6, 7]. Alternatively, a technique based on changes in edges has also been developed [8]. Other schemes use domain knowledge [9, 10, 11, 12, 13] such as predefined models, objects, regions, etc. Hybrids of the above techniques have also been investigated [14, 15, 16, 17, 18, 19, 20]. Although the aforementioned methods are effective, they have the following weaknesses:

- Existing shot boundary detection techniques require many input parameters which are hard to determine but have a significant influence on the quality of the result [21]. Very simple techniques using color histograms need at least three threshold values and the correctness of shot detection results varies from 20 to 80% depending on those values. At least six different threshold values are necessary for the technique using edge change ratio. These values must be chosen properly to get satisfactory results. Choosing these thresholds is a great challenge since their good values vary from video to video.

- The incorrect detection rate of current techniques is very high. Around 20% of incorrect detection rates have been reported [21] for abrupt shot change detection when color histogram and edge change ratio approaches were used. For gradual shot change (i.e., fade or dissolve) detection, the incorrect detection rates have been observed up to several hundred% in worst cases.

- Although the computation cost of current shot detection techniques can be \( O(m) \), where \( m \) is the number of pixels in a frame, the amount of data needed to be processed is still huge. As an example, let us consider a video clip with a frame size of \( 320 \times 240 \), and a frame rate of 30 frames/sec. If a pixel occupies three bytes, these techniques need to deal with more than two Gigabytes for the duration of only five minutes. These approaches are too costly for most video applications.
To address these issues, we propose to detect shot boundaries in a more direct way by tracking the camera motion through the background areas in the video. In our scheme, a common area of the video frames is defined as the background, and inter-frame differences are computed based on this area. We are tracking the camera motion by using background tracking to handle various camera motions. Since this approach relies on the content (background) of the video, it represents a departure from existing pixel-based techniques. Our experiments based on various videos indicate that this new method has many advantages as follows:

- It can detect and classify both abrupt and gradual shot changes in video sequences.
- It needs to process substantially less data since it uses only the background instead of the whole image.
- It is much less sensitive to the threshold values and is very effective in reducing the number of incorrect detection.

The remainder of this paper is organized as follows. In Section 2, we discuss some related techniques in more detail to make the paper self-contained. The proposed scheme and its advantages are presented in Section 3. In Section 4, we discuss our experimental results. Finally, we give our concluding remarks in Section 5.

2 Related Works

Video content analysis is known to be very difficult. It is well known that no single technique is flawless. In this section, we briefly describe some recent schemes, and discuss their weaknesses. Our objective is to address these drawbacks as much as possible in our new technique.

Pixel matching techniques [1, 2, 3] compare the corresponding pixels of every two consecutive frames to determine their similarity. This approach is sensitive to noise and object movement since it is strictly tied to pixel locations. This drawback leads to false detections. To alleviate this problem, one can compare pixels also with neighboring pixels. The sizes of these matching neighborhoods must be selected carefully to obtain good results. Typically, a larger matching neighborhood is preferred for faster-moving objects. However, neighborhoods too large can result in false matches between pixels. This is aggravated by the fact that objects in a video can move at different speeds even within the same shot. These facts make choosing a good value for this threshold a great challenge.
To address the above problems, color histograms can be used [5, 6, 7]. In this approach, the color space (i.e., RGB) of an image is reduced using, say the four most significant bits of each pixel value. One bin [22] is used to count the number of pixels that carry a particular color. The difference between two images can be defined as the sum of the differences of the corresponding bins of the two images. A disadvantage of this approach is that it fails to distinguish images with very different structures but similar color distributions.

Another technique compares only the edge pixels to look for entering and exiting edges [8]. In this scheme, Canny algorithm [23] is used to detect edges in a video frame. This scheme then counts the number of pixels on these edges. Similarly, it computes the number of entering and exiting edge pixels by comparing the current frame to the last one. The edge change ratio (ECR) of a video frame is defined as:

\[
ECR = \frac{\text{number of entering and exiting edge pixels}}{\text{total number of edge pixels}}
\]

If ECR is above a certain threshold, a cut is detected. In order to make the measure robust against object motions, edge pixels in one image, which have edge pixels nearby in the other image (i.e., within \( n \)-pixel distance), are not regarded as entering or exiting edge pixels. Moreover, the Hausdorff distance is performed before the ECR calculation to compensate for global motion. This approach has been shown to perform better than using color histograms. Furthermore, it can detect gradual shot changes such as fade and dissolve. This method, however, requires the user to provide the appropriate values for as many as six parameters. Getting good results demands a lot of trial and error [21].

A different approach [10] which uses specific domain knowledge and model-based parsing was proposed for news broadcasts. This scheme assumes that the same presentation style is used by all news agencies. This may not be true in practice. With the advances in authoring tools, the presentation styles are expected to vary widely among different programs. A limitation of this approach is that we can only use it for pre-formatted programs.

3 Proposed Technique

In this section, we first discuss the motivation for the background tracking approach. We then define a generalized background area and introduce our background tracking technique. Finally, we describe the proposed shot change detection and classification method.
3.1 Motivation

As discussed in Section 1, since a shot is generated from one camera operation, tracking the camera motion is the most direct way to identify shot boundaries. This can be achieved by tracking the background areas in the video frames as follows. A camera operation can consist of various combinations of camera motions such as zooming in or out, moving the camera itself or changing its angle in any direction (vertically, horizontally, or diagonally). As the camera is moving within a shot, the background areas change accordingly. These changes happen slowly since camera motions are generally slow and the background areas are far away from the camera. As a result, if two consecutive frames are in the same shot, they normally share a large portion of their background areas. By comparing these areas between every two consecutive frames, we can determine the shot boundaries. This procedure is referred to as background tracking (BGT) in this paper. We will present an efficient technique for background tracking shortly.

Compared to existing methods, BGT is more reliable due to the following reasons:

- It is difficult to select good values for many thresholds under existing techniques [21]. In general, the content of a video changes significantly over time; and there are no threshold values good for the entire video. This is particularly true for long videos. It is not clear how these thresholds can be adjusted dynamically to adapt to the current content of the video. BGT does not have this problem since it is based on background tracking (i.e., searching for common background areas). We still need a threshold to determine if two consecutive frames share enough background to be considered as in the same shot. However, since background change slowly, we have a lot of leeway to select a good value for this threshold. In other words, the threshold used in BGT is much more robust to handle diverse situations. In fact, we used the same threshold values for all 26 videos in our experiments, whereas a good value must be selected carefully for each video under the techniques using color histograms and edge-change ratios.

- Existing methods compare both the background and the objects in the foreground. These objects can move very quickly causing false detections. There are rare situations in which the background actually changes faster than the objects in the foreground. For instance, the camera can follow a couple of Formula-1 racing cars with the crowd in the background. Even under this circumstance, the changes in the background are still slow enough to guarantee common areas between consecutive frames. On the contrary, existing methods might not be able to cope with these fast changes.
In Section 4, we will give experimental results to demonstrate the above advantages of the proposed technique.

3.2 Definition of Background

![Diagram showing Definition of Background](image)

By definition, areas outside the primary object(s) are considered as part of the background. Thus, a background can have any shape and size. Extracting such a background for each video frame is obviously not feasible for video database applications. Instead, we use in our technique a generalized background for all frames as illustrated in Figure 1 (a). The background is shown as the shaded area referred to as *fixed background area* (FBA) in this paper. We will show how to determine the width \( w \), the height \( h \) and the length \( L = c + 2h \) shortly. This area is about 20% of the whole image. This fact substantially reduces the amount
of data we need to process compared to other techniques.

The rationale for the \( \square \) shape of the FBA is as follows:

- The bottom part of a frame is usually part of some object(s).
- The top bar covers any horizontal camera motion.
- The two columns cover any vertical camera motion.
- The combination of the top bar and the left column can track any camera motion in one diagonal direction. The other diagonal direction is covered by the combination of the top bar and the right column. These two properties are illustrated in Figure 1 (b).

The above properties suggest that we can detect a shot boundary by determining if two consecutive frames share any part of their FBAs. This requires comparing each part of one FBA against every part of the other FBA. We will give experimental results later to demonstrate that the proposed FBA is general enough to accommodate all kinds of background. The performance studies indicate that our technique is very effective for a wide range of videos.

### 3.3 Background Tracking

The main idea of our BGT is that two consecutive frames belong to one shot if they share any parts of the background. This check can be done in three steps as follows:

**Step 1 (Shape Transformation):** The \( \square \)-shape FBA of each frame is transformed into a bar-shape transformed background area (TBA) as shown in Figure 2. It shows that the two vertical columns are rotated outward to form the corresponding TBA. We will see shortly that this bar-shape TBA makes BGT easier.

**Step 2 (Signature Computation):** We apply a modified version of the image reduction technique, called Gaussian Pyramid [24], to reduce each 2-dimensional TBA into a single line of pixels. This 1-dimensional array is referred to as the signature of the TBA. The idea of Gaussian Pyramid was originally introduced for reducing an image to a smaller size. Let us consider a simple one-dimensional case as shown in Figure 3 (a). In this figure, a black dot at level 0 of the tree structure denotes an actual pixel
Figure 2: Shape Transformation of FBA
(a) One-dimensional graphic representation of Gaussian Pyramid

(b) Signature & Sign from Sample TBA

Figure 3: Computation of 'Signature' and 'Sign'
in the original image. At each subsequent level, a pixel is computed from five pixels in the previous level using the following equation:

\[ G_l(x) = \sum_{j=-2}^{j=2} f(j)G_{l-1}(2x + j), \]

where \( G_l(x) \) denotes the value of the pixel at position \( x \) in level \( l \), and \( f(j) \) is the one-dimensional mask generated using three constraints as follows:

1. The mask should be symmetric, that is \( f(j) = f(-j) \) for \( j = 0, 1, 2 \)

2. The sum of the mask should be 1. Let \( f(0) = A \), \( f(1) = f(-1) = B \), and \( f(2) = f(-2) = C \).
   Then \( A + 2B + 2C = 1 \).

3. All nodes at a given level must contribute the same total weight to nodes at the next higher level, \( A + 2C = 2B \).

Thus, 13 pixels in level 0 are first reduced to five pixels in level 1, which are then reduced to one pixel in level 2 as illustrated in Figure 3 (a). In this paper, we use \( A = 0.2 \), \( B = 0.25 \), and \( C = 0.15 \). Let us assume that we have a TBA whose size is \( 13 \times 5 \) as shown in Figure 3 (b). The five pixels in each column are reduced to one pixel using Equation (1). This gives us a single line of 13 pixels, which is used as the 'signature' for the image. This signature captures not only the color, but also the spatial information in the background area. It will be clear later that the linear format makes it efficient to compare images. This signature can be reduced eventually to one pixel as seen in Figure 3 (b). We refer to this one pixel as the 'sign' of the image. We note that the 'sign' used here is not a mathematical term. It is an abstracted form of the signature.

**Step 3 (Background Tracking):** We compare the two signatures of every two consecutive frames as follows.

We shift the two signatures toward each other as illustrated in Figure 4. At each shifting step, we compare the overlapped pixels in the two signatures. Given the shifting direction, we first compare the last pixel of \( \text{signature}_i \) with the first pixel of \( \text{signature}_{i+1} \), then the last two pixels of \( \text{signature}_i \) with the first two pixels of \( \text{signature}_{i+1} \), and so forth until the first pixel of \( \text{signature}_i \) is compared with the last pixel of \( \text{signature}_{i+1} \). In each matching step, only the pixels in the overlapped region (see Figure 4) are compared to determine the maximum continuous match. A continuous match is defined as follows. If \( S \) consecutive pixels in the first signature match \( S \) consecutive pixels in the second signature, the continuous matching score is \( S \). We note that the size of the overlapped region is expanding from one pixel to the length of the signature, and shrinking gradually back to one pixel again.
Figure 4: Background Tracking
as illustrated in Figure 4. During the entire matching process, we maintain a running maximum for the continuous matching scores. At the end, this measure indicates how much the two images share the common background. Although a dynamic programming-based substring matching algorithm [25] can be used to compare the signatures more efficiently, the simple matching procedure, presented above, is sufficient to offer excellent performance.

3.4 Dimensions of the FBA and TBA

Now, let us look at how to determine the width ($w$), the height ($h$) and the length ($L$) of TBA. The parameters we use in this discussion (e.g., $c$, $w$, etc.) are illustrated in Figure 1 (a). We first estimate these parameters as $w'$ and $h'$, respectively. We choose $w'$ to be 10% of the width of video frame as follows:

$$w' = \left\lfloor \frac{c}{10} \right\rfloor \quad (2)$$

This value was determined empirically using our video clips. They show that this value of $w'$ results in TBAs which cover the background areas very well. Using this $w'$, we can compute the other estimate as follows:

$$h' = r - w' \quad (3)$$

In order to use Equation (1) properly, the dimensions of TBA should be in the size set $\{1, 5, 13, 29, 61, 125, \ldots\}$. This is due to the fact that computing one pixel needs five pixels, and computing five pixels needs 13 pixels, and so on as shown in Figure 3 (a) and discussed in Step 2 of the BGT procedure. In general, the $j$th element ($s_j$) in this size set is computed as follows:

$$s_j = 1 + \sum_{i=2}^{j} 2^i \quad \text{for } j = 1, 2, 3, \ldots \quad (4)$$

However, $w'$ computed using Equation (2) cannot always be in the size set. In which case, we need to find a nearest value in the set as summarized in Table 1.

For example, $w' = 16$ if $c = 160$. Since the number 16 is not in the size set, we can use the nearest value 13 instead. Actually, the value 13 comes from Equation (4) when $j = 3$. Thus, we need to find this proper $j$ value in order to select the appropriate value from the size set. We can compute $j$ from $w'$ using the following equation:

$$j = 2 + \left\lfloor \log_2 \left( \frac{w' + 3}{6} \right) \right\rfloor \quad (5)$$
Table 1: Approximate the dimensions using the nearest value from the size set

where \( w' \) is computed using Equation (2). For our example, if we substitute 16 for \( w' \) in Equation (5), we obtain the value 3 for \( j \). Feeding this \( j \) value into Equation (4) gives us the value 13 which is the appropriate \( w \) for this TBA.

After we have determined \( w \), we can compute the length \( L \) as follows. First we compute the approximate length \( L' \) using the following equation:

\[
L' = c + 2h'
\]  
(6)

If this value \( L' \) is in the size set, it can be used as the length of TBA. Otherwise, we follow the same steps used to compute \( w \) as follows. We substitute \( L' \) for \( w' \) in Equation (5) to compute the value \( j \). We then plug this \( j \) value into Equation (4) to get the \( L \) value which is the proper length for the TBA. The height \( h \) of FBA can be computed using the size of column \( (c) \) and the length \( (L) \) as:

\[
h = \frac{(L - c)}{2}
\]  
(7)

### 3.5 Shot Change Detection & Classification

Our shot change detection and classification technique consists of four stages as shown in Figure 5. The first two stages are quick-and-dirty tests used to quickly eliminate the easy cases. Only when these two tests fail, we need to track the background in Stage 3. When a cut has been identified, we perform one more test in Stage 4 to classify the cut as either an abrupt or gradual change. We discuss these steps in more details below:

**Stage 1:** This step is illustrated in Stage (1) of Figure 5. We apply pixel matching to every two consecutive TBAs. To make this matching procedure less sensitive to noise and object movements, we compare a
Figure 5: Shot Change Detection & Classification Procedure
pixel in one TBA with the corresponding pixel in the other TBA, plus the pixels in the neighboring area as illustrated in Figure 5. A match with any one of the pixels in the neighboring area is considered as a match. The final mismatch score is computed as follows:

\[
D_p = \left( \frac{\text{Number of Mismatches}}{\text{Number of Pixels in a TBA}} \right) \times 100 \text{ (\%)}
\]  

(8)

To be conservative, we subject a case to further tests if \( D_p \geq 10\% \). That is, we eliminate a case as a cut, in this early stage, only if it is very obvious (i.e., \( D_p < 10\% \)). We note that the number of neighboring pixels \( (N_n) \), used in each comparison, can be computed based on our experience as follows:

\[
N_n = \left( \frac{\min\{c, r\}}{25} \right)^2
\]

(9)

The optimality of the above equation depends on the content of the video. However, the optimal value is not critical because we use a very low threshold (i.e., \( D_p < 10\% \)) to discard only obvious cases. The reader might be curious why we do not use techniques based on a color histogram or edge change ratio instead. The rationale is that pixel matching is very sensitive. If the similarity is greater than 90\%, we can be certain that the two frames are in the same shot.

Stage 2: This step is illustrated in Stage (2) of Figure 5. We measure the inter-frame difference between two frames by computing the difference of their signs. A sign is a pixel, and has three numerical values, in RGB space, for red, green and blue colors, each ranges from 0 to 255 in our case. Therefore, we represent a sign for a frame \( i \) as \((R_i, G_i, B_i)\). The difference between two signs \( (D_s) \) can be computed as follows:

\[
D_s = \left( \frac{\max(|R_i - R_{i+1}|, |G_i - G_{i+1}|, |B_i - B_{i+1}|)}{256} \right) \times 100 \text{ (\%)}
\]

(10)

The numerator \( \max(|R_i - R_{i+1}|, |G_i - G_{i+1}|, |B_i - B_{i+1}|) \) of the above equation computes the maximum value among RGB differences of two sign values. If this sign would be regarded a vector value or a point in three dimensional space, a difference between two signs can be considered as a distance between them, and computed usually as \( \sqrt{(R_i - R_{i+1})^2 + (G_i - G_{i+1})^2 + (B_i - B_{i+1})^2} \). However, the difference between two signs computed by this distance calculation is usually larger than that computed by the above equation (10) since the distance calculation considers the addition of all differences of three values. This make some similar two signs very different. In equation (10), one maximum among three instead of addition of all is computed. From our experiences, we have figured out that this is more effective to find similar and different signs, consequently more suitable for our technique.
If \( D_p < 10\% \), the two frames are considered as belonging to the same shot; otherwise, we need to proceed to the next test. Again, we conservatively use a very small threshold here to avoid any false dismissal of a cut.

**Stage 3:** This step is illustrated in Stage (3) of Figure 5. We apply our background tracking algorithm described in Step 3 of Section 3.3. If the resulting matching score \( (S) \) is less than \( w \) (the width of TBA), the two frames are considered to be in two different shots.

**Stage 4:** This step is illustrated in Stage (4) of Figure 5. It is performed only when a cut is detected to determine whether it is abrupt or gradual. This procedure is very simple compared to existing techniques. Due to the way we compute the sign for each frame, the sign values should increase or decrease monotonically when a gradual change occurs. Any other patterns, e.g., increases followed by decreases, indicate an abrupt change. To check this pattern, we only need to examine approximately two seconds of video frames before and after the cut. In our experiments, since frames are extracted at a 3-frames/second rate, we test the last six and the next six frames from the frame detected as a cut. As mentioned in Stage 2, we represent a sign for a frame \( i \) as \( (R_i, G_i, B_i) \). Assuming that a cut is detected at frame \( i \), we need to compute three different sequences of substractions as follows (where \( k = 6 \) in our experiments).

\[
(R_{i-k} - R_{i-(k-1)}), ..., (R_{i-2} - R_{i-1}), (R_{i-1} - R_{i}), (R_i - R_{i+1}), (R_{i+1} - R_{i+2}), ..., (R_{i+(k-1)} - R_{i+k}),
\]
\[
(G_{i-k} - G_{i-(k-1)}), ..., (G_{i-2} - G_{i-1}), (G_{i-1} - G_i), (G_i - G_{i+1}), (G_{i+1} - G_{i+2}), ..., (G_{i+(k-1)} - G_{i+k}),
\]
\[
(B_{i-k} - B_{i-(k-1)}), ..., (B_{i-2} - B_{i-1}), (B_{i-1} - B_i), (B_i - B_{i+1}), (B_{i+1} - B_{i+2}), ..., (B_{i+(k-1)} - B_{i+k}).
\]

Each of the above substractions results in a positive number, a negative number, or zero. Let us denote a positive result as `'+'` and a negative result as `'-'`. If we retain only the positive and negative cases to capture only the nature of the changes (`'+'` for increasing and `'-'` for decreasing), then each of the above numerical sequences becomes a sequence of `'+'` and `'-'`, e.g., `'+++--++...'`. To determine if a cut is an abrupt change, we need only look in the three sign sequences for any one of the following two patterns: `'++-', '+'+'`. If none of these patterns are found, the cut is a gradual change.

We note that two thresholds are used in the first two tests. However, they are very conservative values (i.e., \( D_p < 10\%, D_s < 10\% \)) to avoid any possible false dismissal. They are essentially good for all kinds of videos. Another threshold is used in Stage 3 (i.e., \( S \geq w \)). It says that two consecutive frames are considered
as in the same shot if the maximum number of continuous matching pixels between their signatures is at least the width of a TBA. As we have discussed in Section 3.1, since we are performing string matching here, not pixel comparisons, we have a lot of leeway to select a good value for this threshold. We will show in the next section that the same threshold values can be used for all 26 videos of very different characteristics. The same cannot be said about existing techniques.

Let us examine the complexity of our shot change detection and classification procedure. Stage 1 can be done in \( N_n \times p \). Since \( N_n \) is constant, it can be done in \( O(p) \), where \( p \) is the number of pixels in a given TBA. In Stage 2, we need to compute \( \text{sign} \) first. This computation takes about \( O(2^{\log(p+1)}) \), which is actually \( O(p) \). Stage 3 uses ‘signatures’ which are already computed in Stage 2, so it can be completed in \( O((\log(p))^7) \), which is less than \( O(p) \). The last stage can be done in constant time whenever a cut is detected. Therefore, the entire procedure can be completed in \( O(p) \) in any case.

4 Experimental Results

To assess the performance of our technique, we compare it with schemes using color histograms and edge change ratio. For convenience, we denote the proposed techniques as BGT (Background Tracking), the color histogram approach as CHD (Color Histogram Difference), and the edge change ratio method as ECR. They are evaluated in terms of their sensitiveness to the threshold values, and their effectiveness in detecting shot changes.

4.1 Robustness

The video clip used for this study is from the TV drama ‘Chicago Hope.’ The first 200 frames were extracted at the rate of 3 frames/second. We computed the inter-frame difference using the three techniques as follows. Under BGT, we used Equation (10) to compute the difference of the sign values of the frames. The plot is given in Figure 6 (a). Under CHD, we computed the inter-frame differences using the color histogram of each frame. The results are plotted in Figure 6 (b). The edge change ratio of each video frame was used to compute the inter-frame differences under ECR. They are plotted in Figure 6 (c). Both CHD and ECR were applied on the whole image with the appropriate values selected for three and six different parameters, respectively. Comparing these three plots, we observe that the cuts are most evident under BGT. To determine cuts, we compared the inter-frame differences to a user-selected threshold. The number of cuts detected by
Figure 6: Comparison of Sensitivity to the Thresholds
each scheme under various threshold values are plotted in Figure 6 (d). It indicates that BGT is much more robust; the other two schemes are very sensitive to the threshold values.

### 4.2 Effectiveness

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Duration (min:sec)</th>
<th>Abrupt Changes</th>
<th>Gradual Changes</th>
<th>Total Changes</th>
<th>Threshold for CHD</th>
<th>Threshold for ECR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TV Programs</strong></td>
<td>Silk Stalkings (Drama)</td>
<td>10 : 24</td>
<td>91</td>
<td>4</td>
<td>95</td>
<td>0.85</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Scooby Doo Show (Cartoon)</td>
<td>11 : 38</td>
<td>101</td>
<td>5</td>
<td>106</td>
<td>0.85</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Friends (Sitcom)</td>
<td>10 : 22</td>
<td>113</td>
<td>3</td>
<td>116</td>
<td>0.80</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Chicago Hope (Drama)</td>
<td>9 : 47</td>
<td>150</td>
<td>6</td>
<td>156</td>
<td>0.85</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Star Trek Deep Space Nine</td>
<td>12 : 27</td>
<td>109</td>
<td>2</td>
<td>111</td>
<td>0.85</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>All My Children (Soap Opera)</td>
<td>5 : 44</td>
<td>48</td>
<td>2</td>
<td>50</td>
<td>0.75</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>General Hospital (Soap Opera)</td>
<td>11 : 20</td>
<td>109</td>
<td>1</td>
<td>110</td>
<td>0.70</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>Flintstones (Cartoon)</td>
<td>6 : 09</td>
<td>40</td>
<td>8</td>
<td>48</td>
<td>0.65</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Jerry Springer (Talk Show)</td>
<td>4 : 58</td>
<td>106</td>
<td>1</td>
<td>107</td>
<td>0.65</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>Sally (Talk Show)</td>
<td>11 : 19</td>
<td>88</td>
<td>2</td>
<td>90</td>
<td>0.70</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Divorce Court (Talk Show)</td>
<td>11 : 11</td>
<td>159</td>
<td>1</td>
<td>160</td>
<td>0.65</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>TV Commercials</td>
<td>31 : 25</td>
<td>911</td>
<td>56</td>
<td>967</td>
<td>0.70</td>
<td>0.45</td>
</tr>
<tr>
<td><strong>News</strong></td>
<td>National (NBC)</td>
<td>14 : 45</td>
<td>181</td>
<td>21</td>
<td>202</td>
<td>0.70</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Local (ABC)</td>
<td>30 : 27</td>
<td>164</td>
<td>12</td>
<td>176</td>
<td>0.65</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>News Conference</td>
<td>9 : 19</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>0.70</td>
<td>0.35</td>
</tr>
<tr>
<td><strong>Movies</strong></td>
<td>Brave Heart</td>
<td>10 : 03</td>
<td>245</td>
<td>1</td>
<td>246</td>
<td>0.65</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>ATF</td>
<td>11 : 52</td>
<td>220</td>
<td>4</td>
<td>224</td>
<td>0.70</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td>Simon Birch</td>
<td>11 : 08</td>
<td>163</td>
<td>1</td>
<td>164</td>
<td>0.60</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Wag the Dog</td>
<td>11 : 01</td>
<td>102</td>
<td>1</td>
<td>103</td>
<td>0.60</td>
<td>0.40</td>
</tr>
<tr>
<td><strong>Sports Events</strong></td>
<td>Tennis (1999 U.S. Open)</td>
<td>14 : 20</td>
<td>113</td>
<td>1</td>
<td>114</td>
<td>0.60</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td>Mountain Bike Race</td>
<td>15 : 12</td>
<td>82</td>
<td>61</td>
<td>143</td>
<td>0.70</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Football</td>
<td>21 : 26</td>
<td>143</td>
<td>20</td>
<td>163</td>
<td>0.65</td>
<td>0.40</td>
</tr>
<tr>
<td><strong>Documentaries</strong></td>
<td>Today's Vietnam</td>
<td>10 : 29</td>
<td>76</td>
<td>17</td>
<td>93</td>
<td>0.60</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>For All Mankind</td>
<td>16 : 50</td>
<td>126</td>
<td>1</td>
<td>127</td>
<td>0.65</td>
<td>0.40</td>
</tr>
<tr>
<td><strong>Music Videos</strong></td>
<td>Kobe Bryant</td>
<td>3 : 53</td>
<td>50</td>
<td>3</td>
<td>53</td>
<td>0.70</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>Alabama Song</td>
<td>4 : 24</td>
<td>63</td>
<td>2</td>
<td>65</td>
<td>0.60</td>
<td>0.45</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>321 : 53</td>
<td>3762</td>
<td>237</td>
<td>3999</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Test Video Clips

In this subsection, we compare the overall performance of the three techniques. There are many different types of shot changes. However, in practice 99% of all changes are abrupt changes (hard cuts), fades or
dissolves [21]. In this study, we distinguish only two types of changes: abrupt and all other gradual changes (e.g., fades, dissolves, etc.) We do not bother to distinguish the different types of gradual changes because they usually represent less than 5% of shot changes in most video clips [21]. Two parameters ‘recall’ and ‘precision’ are commonly used to evaluate the effectiveness of IR (Information Retrieval) techniques [26]. We also use these metrics in our study as follows:

- **Recall** ($H_r$) is the ratio of the number of shot changes detected correctly over the actual number of shot changes in a given video clip.

- **Precision** ($H_p$) is the ratio of the number of shot changes detected correctly over the total number of shot changes detected (correctly or incorrectly).

These rates are influenced by the setting of several parameters under CHD and ECR. The main parameter ($\theta$, threshold for cut detection) for each approach is listed in Table 2.

To select videos for this study, we investigated the performance studies in [9, 27, 8, 28, 29, 30, 31, 15, 32, 33, 34, 21]. Our test set consists of six categories with 26 different clips, and lasts about 5 hours and 20 minutes. It is more complete than any test sets used in [9, 27, 8, 28, 29, 30, 31, 15, 32, 33, 34, 21]. All the video clips in our test set are in AVI format digitized at 30 frames/second. To reduce computation and to show that our technique performs according to the semantics of the videos, not other factors such as resolution or frame rate, we extract 160 × 120 frames from each video clip at the rate of 3 frames/second. The details of these videos are summarized in Table 2.

The results for gradual shot change detection are given in Table 3. CHD is not included in this study because it cannot detect gradual changes. We observe that BGT reduces the incorrect detection rate almost in half. This can be attributed to the fact that BGT only verifies a gradual change after a cut has been detected. We observe that the incorrect detection rate of ‘Scooby Doo Show’ is much less than that of other videos because it is an animation. In animations, edges and background are much more clear. The results for abrupt shot change detection are presented in Table 3. Again, we observe that BGT is much less likely to make incorrect detections.
<table>
<thead>
<tr>
<th>Video</th>
<th>Gradual Shot Changes</th>
<th>Abrupt Shot Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H_r (Recall)</td>
<td>H_p (Precision)</td>
</tr>
<tr>
<td>Silk Stalkings (Drama)</td>
<td>0.50</td>
<td>0.60</td>
</tr>
<tr>
<td>Scooby Doo Show (Cartoon)</td>
<td>0.40</td>
<td>0.60</td>
</tr>
<tr>
<td>Friends (Sitcom)</td>
<td>0.67</td>
<td>0.33</td>
</tr>
<tr>
<td>Chicago Hope (Drama)</td>
<td>0.50</td>
<td>0.67</td>
</tr>
<tr>
<td>Star Trek Deep Space Nine</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>All My Children (Soap Opera)</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>General Hospital (Soap Opera)</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Flintstones (Cartoon)</td>
<td>0.75</td>
<td>0.50</td>
</tr>
<tr>
<td>Jerry Springer (Talk Show)</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Sally (Talk Show)</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Divorce Court (Talk Show)</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>TV Commercials</td>
<td>0.71</td>
<td>0.61</td>
</tr>
<tr>
<td>National (NBC)</td>
<td>0.86</td>
<td>0.67</td>
</tr>
<tr>
<td>Local (ABC)</td>
<td>0.67</td>
<td>0.92</td>
</tr>
<tr>
<td>News Conference</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Brave Heart</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>ATF</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>Simon Birch</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Wag the Dog</td>
<td>0.98</td>
<td>0.96</td>
</tr>
<tr>
<td>Tennis (1999 U.S. Open)</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Mountain Bike Race</td>
<td>0.80</td>
<td>0.64</td>
</tr>
<tr>
<td>Football</td>
<td>0.70</td>
<td>0.55</td>
</tr>
<tr>
<td>Today's Vietnam</td>
<td>0.76</td>
<td>0.65</td>
</tr>
<tr>
<td>For All Mankind</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Kobe Bryant</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Alabama Song</td>
<td>0.67</td>
<td>0.33</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.63</strong></td>
<td><strong>0.60</strong></td>
</tr>
</tbody>
</table>

Table 3: Effectiveness of BGT, CHD and ECR under various videos.
Figure 7: Frames in a shot of 'ATF (movie)'

Figure 8: Frames in a shot of 'Brave Heart (movie)'

22
4.3 BGT Is Content-Based

Let us look at three examples to have a better feel of why BGT is content-based. In Figure 7, we show eight frames from a shot of ’ATF (Movie)’. In this shot, a woman is pushing a door and coming out of a room. The camera and its angle are fixed at one position throughout this shot. Only BGT is capable of detecting this shot since it focuses on the background (the door and the wall). The other two techniques cut the shot between frame #1796 and #1797 because the color histogram difference and the edge change ratio between those two frames are more than 89% and 80%, respectively.

Another shot is given in Figure 8 showing eight frames from a shot of ’Brave Heart (movie)’. In this shot, a man is running very fast through the woods with the camera following him. This is a good example to show that BGT can also handle situations in which the background is changing fast while the objects in the foreground change little. In frame #96, the thick woods is the background, but it becomes thin quickly and we can see the sky throughout most background area in frame #97. The color histogram difference and the edge change ratio between frame #96 and #97 are more than 75% and 78%, respectively. As a result, they cut the shot between those two frames. In contrast, BGT successfully tracked the sky and the trees in the background. It correctly detects this sequence as a shot.

An interesting example is shown in Figure 9 which presents the eight frames from a shot of ’Alabama Song (Music Video)’.
In this shot, a singer is standing still while she is singing. The camera and its angle are also fixed. During frame #176 and #177, a very bright light illuminates the object and its surroundings. As a result, the color histogram difference between frame #175 and #176, and frame #177 and #178 are 77% and 78%, respectively. The edge change ratios between those frames are 79% and 72%, respectively. Consequently, both of these schemes mistook this sequence as three shots (a cut between frame #175 and #176, and another cut between frame #177 and #178). This example shows that colors and edges can change significantly due to deviation in the lighting condition. However, BGT can still recognize this kind of sequence as a shot since it focuses on the background (the night sky in this case).

The above examples illustrate the effectiveness of background tracking in capturing the semantics of the videos. The performance results confirm that this semantic-based approach is more tolerant of fast changes in the background, foreground, or lighting condition. It is much more robust than existing methods.

5 Concluding Remarks

The contribution of this paper is a new approach to detect and classify shot changes in video sequences. It is much less expensive and more reliable than existing techniques.

Our solution is less expensive since we process only the background pixels, which are 400% less than the number of pixels must be examined by existing methods. This fact makes our technique suitable for large video databases. The performance advantage of our approach is due to the fact that we take into account not only the differences in the pixel values but also the semantic similarity between video frames based on their background. We gave three experimental examples to illustrate this benefit. They showed that the proposed scheme is much more tolerance of fast changes in the background, foreground, and lighting conditions.

To assess the effectiveness of our technique, we compared it with techniques using color histograms and edge change ratios. Our experiments were based on six categories of 26 videos. The results indicate that our method is more robust. It is much less sensitive to the user-selected threshold values. We also observed that the new method reduced the number of incorrect detections at least in half.
References


Biographies

- **Kien A. Hua** received the B. S. degree in Computer Science, M.S. and Ph.D. degrees in Electrical Engineering, all from the University of Illinois at Urbana-Champaign, in 1982, 1984 and 1987, respectively. From 1987 to 1990, he worked for IBM, where he led a project to implement a highly parallel computer. This was a precursor to the series of products known as SPx (i.e., SP1, SP2, etc.) While at IBM, Dr. Hua also led another project to design a high-performance processor for mainframe computers. This work led to a Best Paper Award and a Best Presenter Award from IEEE for his presentation at the International Conference on Computer Design in 1990. Dr. Hua joined the University of Central Florida in 1990. He is currently an associate professor in the School of Electrical Engineering and Computer Science, and the Director of the Database Systems Laboratory. His current research interests include multimedia databases, parallel databases, multimedia communications, and Web computing. He and his students have coauthored over 70 papers including two recognized as best papers, and one as a top paper. Dr. Hua has served on various conference program committees for the ACM and IEEE including the ACM Multimedia conferences.

- **JungHwan Oh** is currently a Ph.D. candidate in the School of Electrical Engineering and Computer Science, University of Central Florida. He received the M.S. degree in computer science from the same university in 1998. He has published 9 journal and conference papers. His research topics include very large multimedia database management system, and adaptive multicast data delivery in wired and wireless Environment.