Chapter IX

Innovative Shot Boundary Detection for Video Indexing

Shu-Ching Chen, Florida International University, USA

Mei-Ling Shyu, University of Miami, USA

Chengcui Zhang, Florida International University, USA

ABSTRACT

Recently, multimedia information, especially video data, has been made overwhelmingly accessible with the rapid advances in communication and multimedia computing technologies. Video is popular in many applications, which makes the efficient management and retrieval of the growing amount of video information very important. Toward such a demand, an effective video shot boundary detection method is necessary, which is a fundamental operation required in many multimedia applications. In this chapter, an innovative shot boundary detection method using an unsupervised segmentation algorithm and the technique of object tracking based on the segmentation mask maps *is presented. A series of experiments on various types of video types are performed, and the experimental results show that our method can obtain object-level information of the video frames as well as accurate shot boundary detection, which are very useful for video content indexing.*

INTRODUCTION

Unlike traditional database systems that have text or numerical data, a multimedia database or information system may contain different media such as text, image, audio, and video. Video, in particular, has become more and more popular in many applications such as education and training, video conferencing, video-on-demand (VOD), and news services. The traditional way for the users to search for certain content in a video is to fast-forward or rewind, which are sequential processes, making it difficult for the users to browse a video sequence directly based on their interests. Hence, it becomes important to be able to organize video data and provide the visual content in compact forms in multimedia applications (Zabih, Miller, & Mai, 1995).

In many multimedia applications such as digital libraries and VOD, video shot boundary detection is fundamental and must be performed prior to all other processes (Shahraray, 1995; Zhang & Smoliar, 1994). A video shot is a video sequence that consists of continuous video frames for one action, and shot boundary detection is an operation to divide the video data into physical video shots. Many video shot boundary detection methods have been proposed in the literature. Most of them use low-level global features in the matching process between two consecutive frames for shot boundary detection, for example, using the luminance pixel-wise difference (Zhang, Kankanhalli, & Smoliar, 1993), luminance or color histogram difference (Swanberg, Shu, & Jain, 1993), edge difference (Zabih et al., 1995), and the orientation histogram (Ngo, Pong, & Chin, 2000). However, these low-level features cannot provide satisfactory results for shot boundary detection since luminance or color is sensitive to small changes. For example, Yeo and Liu (1995) proposed a method that uses the luminance histogram difference of DC images, which is very sensitive to luminance changes. There are also approaches focusing on the compressed video data domain. For example, Lee, Kim, and Choi (2000) proposed a fast scene/shot change detection method, and Hwang and Jeong (1998) proposed the directional information retrieving method by using the discrete cosine transform (DCT) coefficients in MPEG video data.

In addition, dynamic and adaptive threshold determination is also applied to enhance the accuracy and robustness of the existing techniques in shot cuts detection (Alattar, 1997; Gunsel, Ferman, & Tekalp, 1998; Truong, Dorai, & Venkatesh, 2000). In Gunsel et al. (1998), the unsupervised clustering algorithm proposed a generic technique that does not need threshold setting and allows multiple features to be used simultaneously; while an adaptive threshold determination method that reduces the artifacts created by noise and motion in shot change detection was proposed by Truong et al. (2000).

In this chapter, we present an innovative shot boundary detection method using an unsupervised image-segmentation algorithm and the object-tracking technique on the uncompressed video data. In our method, the image-segmentation algorithm extracts the segmentation mask map of each video frame automatically, which can be deemed as the clustering feature map of each frame and where the pixels in each frame have been grouped into different classes (e.g., two classes). Then the difference between the segmentation mask maps of two frames is checked. Moreover, due to camera panning and tilting, we propose an object-tracking method based on the segmentation results to enhance the matching. The cost for object tracking is almost trivial since the segmentation results are already available. In addition, the bounding boxes and the positions of

the segments within each frame obtained from the segmentation are used for our key frame representation. In order to reduce the computational cost, we also apply the traditional pixel-level comparison for pre-processing, in addition to segmentation and object tracking. The basic idea in pixel-level comparison is to compute the differences in values of corresponding pixels between two successive frames. One threshold is used to determine whether the value of the corresponding pixels has changed, while another threshold is used to measure the percentage of changed pixels between two successive frames. If the percentage of changes exceeds some pre-defined threshold, then a new shot cut is detected. This method is very simple, but the disadvantage is that it is very sensitive to object and camera movements. To overcome its shortcomings, pixel-level comparison is embedded into the techniques of object tracking and image segmentation in our method. The advantages of our shot boundary detection method are:

1. It is fully unsupervised, without any user interventions.
2. The algorithm for comparing two frames is simple and fast.
3. The object-level segmentation results can be further used for video indexing and content analysis.

We begin with a literature review and the motivations of the proposed framework. Then the unsupervised image-segmentation and object-tracking techniques are introduced. After that, our experimental results are presented, and the future trends are discussed. Finally, a brief conclusion is given.

BACKGROUND

In this section, the existing approaches for video shot detection and their relative advantages and disadvantages are discussed. Video segmentation is the first step for automatic indexing of digital video for browsing and retrieval. The goal is to separate the video into a set of shots that can be used as the basis for video indexing and browsing. Most of the algorithms process uncompressed video data, while some of them operate directly on the compressed video data.

A survey on video indexing and video segmentation in uncompressed data domain was presented by Gargi, Kasturi, and Antani (1998). The shot boundary detection algorithms in the uncompressed domain process uncompressed video, and a similarity measure between successive frames is defined (Nagasaka & Tanaka, 1995; Zhang et al., 1993). Lots of the approaches use pixel-level comparison to compute the differences in values of corresponding pixels between two successive frames; however, it is very sensitive to object and camera movements. In our method, pixel-level comparison is embedded into the techniques of object tracking and image segmentation in order to overcome its shortcomings and to reduce the computation cost. Other kinds of comparison techniques used in the uncompressed domain are block-wise comparison and histogram-based comparison. Block-wise comparison reduces the sensitivity to object and camera movements by utilizing the local characteristics (e.g., mean and variance intensity values) of the blocks. In this kind of approach, each frame is divided into several blocks that are compared with their corresponding blocks in the successive frame. If the number of changed blocks exceeds some threshold, then a shot cut is detected. This

method is more robust, but it is still sensitive to fast object movement or camera panning. Moreover, it is also highly possible to introduce an incorrect matching between two blocks that have the same mean and variance values but with totally different contents, due to the fact that the mean and variance values of a block are not good enough to represent the block's characteristics (Xiong & Lee, 1998). In our method, the idea of block matching is partially adopted in the object-tracking technique. Instead of dividing a frame into fixed size of blocks absolutely, an innovative image-segmentation method is employed to cluster the pixels in a frame into multiple classes (normally two classes) and obtain the segments (blocks). These segments (blocks) are then tracked and matched between two successive frames. On the other hand, histogram-based comparison is based on the premise that since the object moving between two successive frames is relatively small, there will not be a big difference between their histograms. It is more robust to small rotations and slow variations (Pass & Zahib, 1999; Swain, 1993). However, two successive frames may have similar histograms but with different contents.

In the compressed domain, there are also many shot boundary detection algorithms, especially in the MPEG format. It is suitable for video shot boundary detection because the encoded video stream already contains many features, including the DCT coefficients, motion vectors, etc. Arman, Hsu, and Chiu (1993) use the DCT coefficients of I frames as the similarity measure between successive frames; while the dc-images are used to compare successive frames, where the (i,j) pixel value of the dc-image is the average value of the (i,j) block of the image (Yeo & Liu, 1995). Hwang and Jeong (1998) utilized the changes of directional information in the DCT domain to detect the shot breaks automatically. The DCT coefficient-based method was further improved by Lee et al. (2000), who used the binary edge maps as a representation of the key frames so that two frames could then be compared by calculating a correlation between their edge maps. Its advantage is that it gives directly the edge information such as orientation, strength, and offset from the DCT coefficients, and its disadvantage is similar to all the compressed domain-based methods, that is, sensitivity to different video compressing formats.

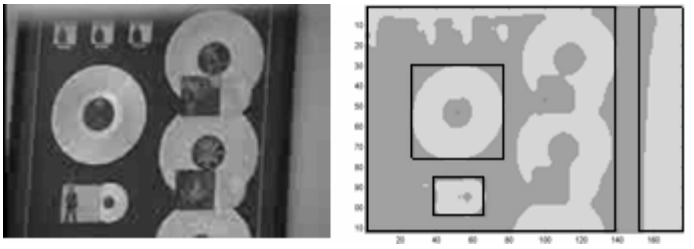
PROPOSED SHOT BOUNDARY DETECTION METHOD

In this section, we first explain how the unsupervised segmentation algorithm and object tracking work, and then provide the steps of the shot change detection method based on the discussion.

Segmentation Information Extraction

In this chapter, we use an unsupervised segmentation algorithm to partition the video frames. First, the concepts of a class and a segment should be clarified. A class is characterized by a statistical description and consists of all the regions in a video frame that follow this description; a segment is an instance of a class. In this algorithm, the partition and the class parameters are treated as random variables. This is illustrated in Figure 1. The light gray areas and dark gray areas in the right segmentation-mask map represent two different classes respectively. Considering the light gray class, there are a total of four segments within this class (the CDs, for example). Notice that each segment

Figure 1. Examples of classes *and* segments *(The original video frame is on the left, and the segmentation mask map of the left frame is on the right.)*



is bounded by a bounding box and has a centroid, which are the results of segment extraction. The details of segment extraction will be discussed in a later section.

The method for partitioning a video frame starts with a random partition and employs an iterative algorithm to estimate the partition and the class parameters jointly (Chen, Sista, Shyu, & Kashyap, 1999, 2000; Chen, Shyu, & Kashyap, 2000). The intuition for using an iterative way is that a given class description determines a partition and, similarly, a given partition gives rise to a class description. So the partition and the class parameters have to be estimated iteratively and simultaneously from the data.

Suppose there are two classes — *class*1 and *class*2. Let the partition variable be $\boldsymbol{c} = \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$, and the classes be parameterized by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$. Also, suppose all the pixel values y_{ij} (in the image data *Y*) belonging to class *k* (*k*=1,2) are put into a vector $\boldsymbol{Y}_k$. Each row of the matrix Φ is given by (1, *i*, *j*, *ij*) and a_k is the vector of parameters $(a_{k0}, \ldots, a_{k3})^T$.

$$y_{ij} = a_{k0} + a_{k1} i + a_{k2} j + a_{k3} ij, \ \forall (i, j)\, y_{ij} \in \boldsymbol{c}_k \quad \textbf{(1)}$$

$$\boldsymbol{Y}_k = \Phi a_k \quad \textbf{(2)}$$

$$\hat{a}_k = (\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{Y}_k \quad \textbf{(3)}$$

The best partition is estimated as that which maximizes the a posteriori probability (MAP) of the partition variable given the image data *Y*. Now, the MAP estimates of $\boldsymbol{c} = \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$ and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ are given by

$$\begin{aligned}(\hat{c}, \hat{\theta}) &= Arg \max_{(c,\theta)} P(c, \theta \mid Y) \\ &= Arg \max_{(c,\theta)} P(Y \mid c, \theta) P(c, \theta)\end{aligned} \quad \textbf{(4)}$$

Let $J(c, \theta)$ be the functional to be minimized. With the above assumptions, this joint estimation can be simplified to the following form:

$$(\hat{c}, \hat{\theta}) = Arg \min_{(c,\theta)} J(c_1, c_2, \theta_1, \theta_2) \quad \textbf{(5)}$$

$$J(c_1, c_2, \theta_1, \theta_2) = \sum_{y_{ij} \in c_1} -\ln p_1(y_{ij}; \theta_1) + \sum_{y_{ij} \in c_2} -\ln p_2(y_{ij}; \theta_2) \quad \textbf{(6)}$$

Thus, the problem of segmentation becomes the problem of simultaneously estimating the class partition and the parameter for each class. With regard to the parameter estimation, we can use equation (3) to directly compute the parameter for each assigned set of class labels without any numerical optimization methods. For the class partition estimation, we assign pixel y_{ij} to the class that gives the lowest value of $-\ln p_k(y_{ij} \mid \theta_k)$. The decision rule is:

$$y_{ij} \in \hat{c}_1 \text{ if } -\ln p_1(y_{ij}) \leq -\ln p_2(y_{ij}) \quad \textbf{(7)}$$

$$y_{ij} \in \hat{c}_2 \text{ otherwise} \quad \textbf{(8)}$$

Just as shown in Figure 2, the algorithm starts with an arbitrary partition of the data in the first video frame and computes the corresponding class parameters. Using these class parameters and the data, a new partition is estimated. Both the partition and the class parameters are iteratively refined until there is no further change in them. We note here that the functional *J* is not convex. Hence, its minimization may yield a local minimum, which guarantees the convergence of this iterative algorithm. Since the successive frames do not differ much due to the high temporal sampling rate, the partitions of adjacent frames do not differ significantly. The key idea is then to use the method successively on each frame of the video, incorporating the partition of the previous frame as the initial condition while partitioning the current frame, which can greatly reduce the computing cost.

We should point out that the SPCPE algorithm could not only simultaneously estimate the partition and class parameters, but also estimate the appropriate number of

Figure 2. Flowchart of SPCPE algorithm

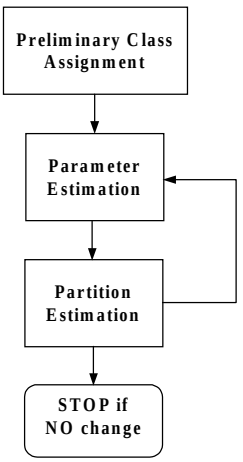
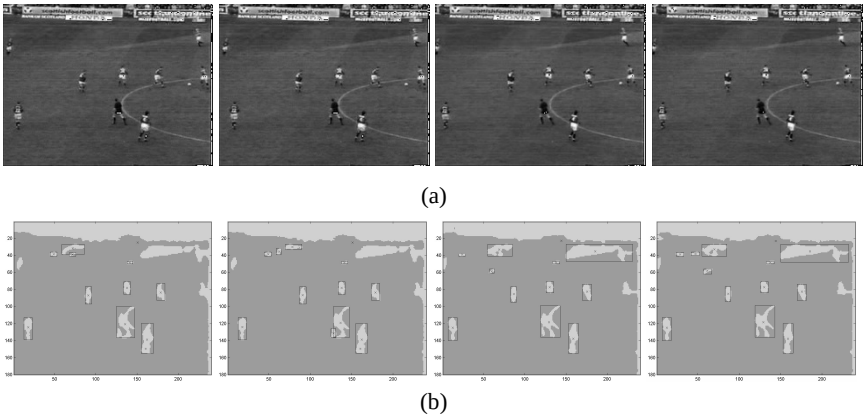


Figure 3. Object tracking: (a) Example of video sequence; (b) Segmentation mask maps and bounding boxes for (a)



the classes in the mean time by some easy extension of the algorithm. Moreover, it can handle multiple classes rather than two. In our experiment, we just use two classes in segmentation since two classes are efficient and good enough for our purpose in this application domain.

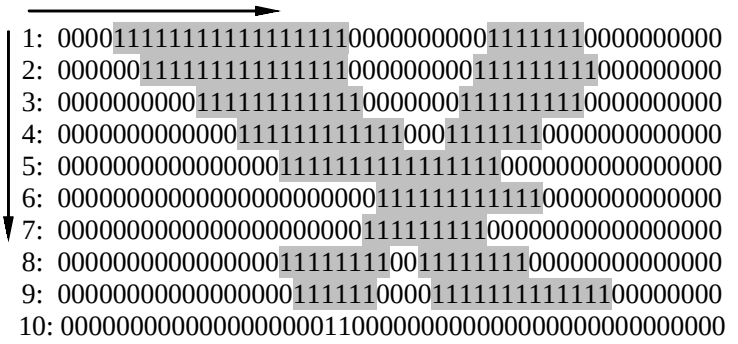
Object Tracking

The first step for object tracking is to identify the segments in each class in each frame. Then the bounding box and the centroid point for that segment are obtained. For example, Figure 3(b) shows the segmentation mask maps of the video sequence in Figure 3(a). In this figure, the player, soccer ball, and the signboard belong to Class 2 while the ground belongs to Class 1. As shown in Figure 3(b), the segments corresponding to the ball, player, and signboard are bounded by their minimal bounding boxes and represented by their centroid points.

Line Merging Algorithm (LMA) for Extracting Segments

Unlike the traditional way to do segment extraction such as the *seeding and region growing* method used by Sista and Kashyap (1999), we use a computationally simple and fast method called a *line merging algorithm (LMA)* to extract the segments from the segmented frames. The basic idea is to scan the segmented frame either row-wise or column-wise. If the number of rows (columns) is less than the number of columns (rows), then row-wise (column-wise) is used respectively. For example, as shown in Figure 4, suppose the pixels with value '*1*' represent the segment we want to extract; we scan the segmented frame row by row. By scanning the first row, we get two lines and let each line represent a new segment so that we have two segments at the beginning. In scanning Rows 2 to 4, we merge the new lines in each row with the lines in previous rows to form the group of lines for each segment. At Row 5, we get one line and find out that it can be merged with both of the two segments, which means we must merge the two previously obtained segments to form a new segment, so that now we have only one big segment.

Figure 4. Segmentation mask map



Similarly, at Row 8, two lines belong to the same segment because they can be merged with the same line in Row 7.

The pseudo codes for the *line merging algorithm (LMA)* are listed in Algorithm 1 and Algorithm 2.

Compared with the *seeding and region growing* method, the proposed algorithm extracts all the segments and their bounding boxes as well as their centroids within one scanning process, while the *seeding and region growing* method needs to scan the input data for an indeterminate number of times depending on the layout of the segments in the frame. Moreover, the proposed algorithm needs much less space than the *seeding and region growing* method.

The next step for object tracking is to connect the related segments in successive frames. The idea is to connect two segments that are spatially the closest in the adjacent frames (Sista & Kashyap, 1999). In other words, the Euclidean distances between the centroids of the segments in adjacent frames are used as the criteria to track the related segments. In addition, size restriction should be employed in determining the related segments in successive frames.

In fact, the proposed object-tracking method can be called a "block-motion tracking" method since it is an extension of the macroblock-matching technique used in motion estimation (Furht, Smoliar, & Zhang, 1995; Gall, 1991) between successive frames. The proposed object- tracking method is based on the segmentation results and goes much further than the macroblock- matching technique because it can choose the appropriate macroblocks (segments) within a specific frame by segmentation and track their motions instead of fixed-size and pre-determinate macroblocks.

Shot Boundary Detection Method

Our method combines three main techniques together: segmentation, object tracking, and the traditional pixel-level comparison method. In the traditional pixel-level comparison approach, the gray-scale values of the pixels at the corresponding locations in two successive frames are subtracted, and the absolute value is used as a measure of dissimilarity between the pixel values. If this value exceeds a certain threshold, then the

Algorithm 1.

Algorithm: GetSegments(V, i, A[i])→ to get the new lines of each row.

V: the input vector of segmented frame of row 'i';

'i': the current row we are scanning;

A[i]: a list to store the segments.

GetSegments(V, i, A[i])

```
1)  Number_of_segments = -1;
2)  Segment D[col/2]; /* D is the temporary variable to store the line segments
                         in row i. The maximal size of D is col/2. */
3)  for j from 1 to col
4)    if V[j] == 1
5)      if j == 1 /* if the first line segment is at the beginning of the current row,
                     add it to array D and increase the number of line segments. */
6)        number_of_segments++;
7)        D[number_of_segments].data = data; /* data contains the i and j values */
8)      else if V[j-1] == 0 /* detect a new line segment and add it to array D */
9)        number_of_segments++;
10)       D[number_of_segments].data = data;
11)   else D[number_of_segments].data += data;
       /* collect all the pixels belonging to the same line segment together. */
12)   end if;
13) end if;
14) for k from 0 to number_of_segments /* copy the line segments in D to the
                                          data structure in A[i]. */
15)     A[i].Add(D[k]);
16) end for;
```

Algorithm 2.

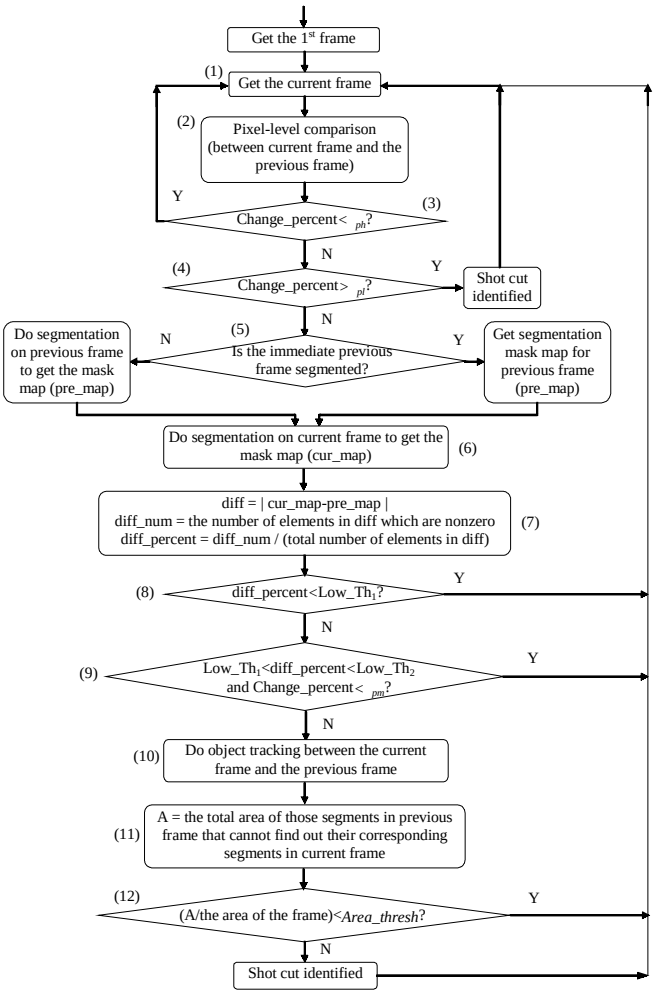
Algorithm: GetBoundingBox(m[row][col])→ to combine A[i] and A[i-1] by checking each line in A[i] and A[i-1] and combining those lines which belong to the same segment.

m[row][col]: the input matrix of segmented frame of size row by column.

GetBoundingBox(m[row][col])

```
1)  number_of_objects =0; /* initially there is zero object identified. */
2)  for k1 from 1 to row
3)      GetSegments(m[k1][col], k1, A[k1]) /* get the line segments in
                                              current row*/
4)      for k2 from 1 to A[k1].size
        /* between the current row and the previous row, check and merge the
           corresponding line segments in them which belong to the same object
           to one big segment. */
5)        for k3 from 1 to A[k1-1].size
6)          if Segment Sk1 in A[k1] ∩ Segment Sk2 in A[k1-1] != null
7)            combine Sk1 and Sk2 into one segment
8)        end for
9)      end for
10) end for
```

Figure 5. Flowchart of the proposed shot boundary detection method



pixel gray scale is said to have changed. The percentage of the pixels that have changed is the measure of dissimilarity between the frames. This approach is computationally simple but sensitive to digitalization noise, illumination changes, and the object moving. On the other hand, the proposed segmentation and object- tracking techniques are much less sensitive to the above factors. In our method, we use the pixel-level comparison for pre-processing. By applying a strict threshold for the percentage of changed pixels, we want to make sure that we will not introduce any incorrect shot cuts that are identified by pixel-level comparison by fault. The advantage to combining the pixel-level comparison is that it can alleviate the cost of computation because of its simplicity. In other word, we apply the segmentation and object-tracking techniques only when it is necessary.

Figure 5 shows the flowchart of the proposed shot boundary detection. The steps are given in the following:

1. Do the pixel-level comparison between the currently processed video frame and the immediate preceding frame (see chart boxes 1 and 2 in Figure 5). Let the percentage of change be *change_percent* and check this variable (chart box 3). If the current frame is not identified as a shot cut, which means that *change_percent*$<d_{ph}$,, then go on to process the next video frame (chart box 1). Otherwise go to step 2 (chart box 4).
2. If *change_percent*$>\delta_{pl}$ (chart box 4), the current frame is identified as a shot cut. Go to step 1 and process the next frame (chart box 1). Otherwise go to step 3 (chart box 5).
3. Do the segmentation on the previous frame only if the previous frame has never been segmented (chart box 5). If the previous frame has been segmented before, we only need to obtain its segmentation mask map directly. Then do segmentation on the current frame (chart box 6). Get the current and the previous segmentation mask maps for these two frames. Let the variable *cur_map* represent the current segmentation mask map's value and variable *pre_map* represent the value of the previous segmentation mask map. Note that the variables *cur_map* and *pre_map* can be deemed as two matrices. Go to step 4 (chart box 7).
4. *diff* = | *cur_map-pre_map* |; where the variable *diff* is the point-to-point subtraction between two successive segmentation mask maps.
   diff_num = the number of elements in *diff* which are nonzero;
   diff_percent = *diff_num* / (total number of elements in *diff*); where the variable *diff_percent* is the percentage of changes between the two successive segmentation mask maps.
   Go to step 5 (chart box 8).
5. Check the variable *diff_percent* (chart box 8).

   If *diff_percent* < Low_Th_1
   Not shot boundary. Go to step 1 and process the next frame (chart box 1).
   Else
   If Low_Th_1 < *diff_percent* < Low_Th_2 and *change_percent*$<d_{pm}$ // chart box 9
   Not shot boundary. Go to step 1 and process the next frame (chart box 1).
   Else
   Do object tracking between the current frame and the previous frame. Let variable *A* be the total area of those segments in the previous frame that cannot find out their corresponding segments in the current frame; // chart boxes 10, 11
   If (A/the area of the frame)<*Area_thresh* // chart box 12
   Not shot boundary. Go to step 1 and process the next frame (chart box 1).
   Else
   The current frame is identified as shot cut.
   Go to step 1 and process the next frame (chart box 1).
   End if;
   End if;
   End if;
   (Here, δ_{ph}, δ_{pl}, δ_{pm}, Low_Th_1 and Low_Th_2 are threshold values for variables *change_percent* and *diff_percent* that are derived from the experiential values.)

Table 1. Video data used for experiments

Name	Type	Number of Frames	Shot Cuts
News1	News	731	19
News2	News	1262	26
News3	News	4225	90
Labwork	Documentary	495	15
Robert	MTV	885	26
Carglass	Commercial	1294	29
Aussie2g2	Sports	511	19
Flo1	Sports	385	8
Flo2	Sports	406	10
AligoISA	Sports	418	11

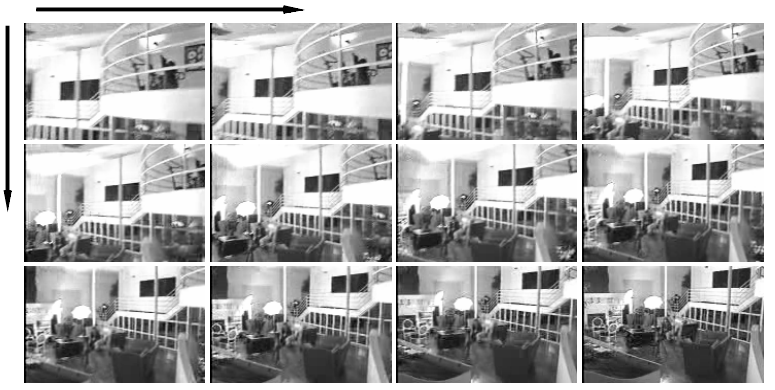
EXPERIMENTAL RESULTS

We have performed a series of experiments on various video types such as the TV news videos (in MPEG-1 format) that include FOX 25 LIVE NEWS, ABC 7 NEWS and WNBC NEWS. Other video types used in our experiment include the music MTV video, documentary video, and sports video such as the soccer game. The average size of each frame in the sample video clips is 180 rows and 240 columns. All the MPEG video clips are downloaded from the URLs listed in the Appendix. Table 1 gives the statistics of all the video clips used. The experimental results demonstrate the effectiveness of the proposed shot boundary detection algorithm. Next, we will see how the proposed method detects the different types of shot boundaries that cannot be correctly identified by the traditional pixel-level comparison method.

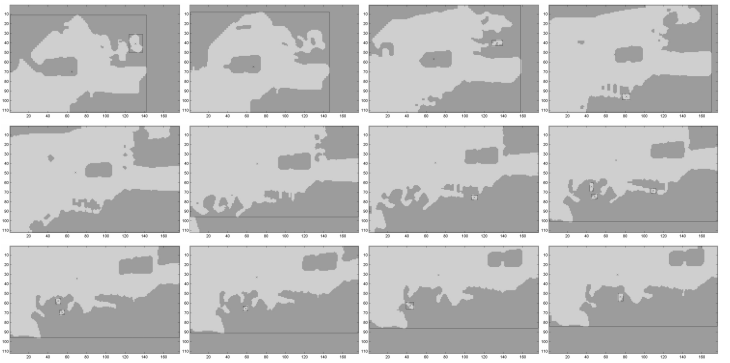
Case 1. Camera Panning and Tilting

Figure 6 gives an example of the camera panning while tilting. Figure 6(a) is the original video sequence, and Figure 6(b) is the corresponding segmentation mask maps for (a). In this case, the pixel-level comparison will identify too many incorrect shot cuts since the "objects" in the shot move and turn from one frame to another. But as we can see from Figure 6, the segmentation mask maps can still represent the contents of the video frames very well. Since the segmentation mask maps are binary data, it is very simple and fast to compare the two mask maps of the successive frames. Moreover, by combining the object-tracking method, most of the segment movements can be tracked so that we know that there is no major shot boundary if the segments in two successive frames can be tracked and matched well according to the object-tracking method mentioned in Section 2.2.

Figure 6. An example of a video sequence for camera panning and tilting: (a) the temporal order of the sequence is from the top-left to the right-bottom; (b) the corresponding segmentation mask maps for the video sequence shown in (a)



(a)



(b)

Case 2. Zoom In and Zoom Out

Figure 7 gives an example of a video sequence of camera zooming out. Similarly, we also apply the combination of the segmentation and object tracking to identify this sequence as a single shot.

Case 3. Fade In and Fade Out

Figure 8 gives an example of a video sequence for shots fading out. We still can identify this video sequence as one shot (the shot cut is marked by dotted border in Figure 8). This is a good example to show that the proposed segmentation, together with object-tracking technique, is not sensitive to luminance changes.

In Figure 9, a fancier example of a video sequence is given to show the effectiveness of the proposed method. In this example, one shot is fading in, while another shot is fading out continuously. By applying the proposed method, this sequence is divided into three

Figure 7. An example of a video sequence of zooming out: (a) the temporal order of the sequence is from the top-left to the right-bottom; (b) the corresponding segmentation mask maps for the video sequence shown in (a)

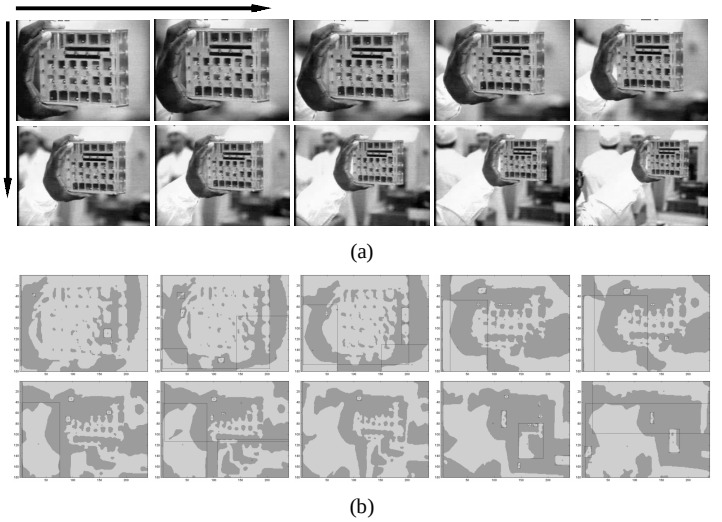
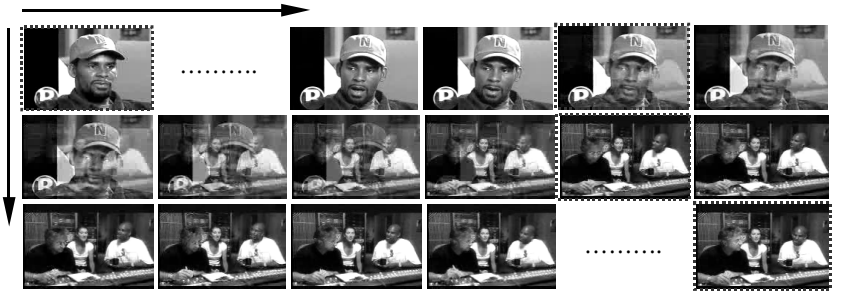


Figure 8. An example of a video sequence of fading in (the frame with dotted border is shot cut detected by the proposed method)

different shots, and the identified shot cuts are marked by dotted borders as shown in Figure 9. The first shot and the third shot are clearly and correctly identified, while the second shot cut represents the intermediate transforming process from the first shot to the third shot. In our experiments, this kind of video sequences can be divided into either two or three shots. In case of two shots, the intermediate transforming sequence belongs to either the previous shot or the following shot.

The performance is given in terms of *precision* and *recall* parameters. N_C means the number of correct shot boundary detections, N_E means the number of incorrect shot boundary detections, and N_M means the number of missed shot detections.

Figure 9. An example of a video sequence of continuously transforming from one shot to another shot (the frames with dotted borders are shot cuts detected by the proposed method)



$$precision = \frac{N_C}{N_C + N_E}$$

$$recall = \frac{N_C}{N_C + N_M}$$

The summary of the proposed method is shown in Table 2 and Figure 10 via the *precision* and *recall* parameters. In our experiments, the *recall* and the *precision* values are both above ninety percent. Our results are comparable to other techniques such as the PM method in Lee et al. (2000) and DC method in Yeo and Liu (1995). Moreover, the *recall* results seem very stable and promising because most of the *recall* results are 100 percent. The DC method is very sensitive to luminance and color change, but the proposed method is not. As seen in Table 2, the precision values for sports and commercial videos are a little lower (but still above 90 percent) than other types of videos because there are lots of fast movements and fancy transformation between successive frames. As mentioned before, the method of using low-level features is very sensitive to luminance and color change, but our segmentation-based method is not. One thing that should be mentioned here is that even it is efficient to simply compare the segmentation mask maps, the employment of the object-tracking technique is very useful in case of camera panning and tilting. It helps to reduce the number of incorrectly identified shot cuts. Another thing is that by combining the pixel-level comparison, the number of the video frames that need to do segmentation and object tracking is greatly reduced. As can be seen from Table 2, the percentage of the reduced frames that do not need segmentation and object tracking is between fifty percent and eighty percent.

Moreover, the process produces not only the shot cuts, but also the object-level segmentation results. Each detected shot cut is selected as a key frame and has been modeled by the features of its segments such as the bounding boxes and centroids. Based on this information, we can further structure the video content using some existing

Table 2. Precision and Recall Parameters

Name	Type	Precision	Recall	Computation Reduce by Pixel-level Comparison
News1	News	0.95	1.00	72%
News2	News	0.96	0.96	75%
News3	News	0.98	1.00	80%
Labwork	Documentary	0.94	1.00	80%
Robert	MTV	0.96	1.000	70%
Carglass	Commercial	0.933	1.000	60%
Aussie2g2	Sports	0.950	1.000	70%
Flo1	Sports	0.889	1.000	60%
Flo2	Sports	0.909	1.000	67%
AligoISA	Sports	0.910	1.000	53%

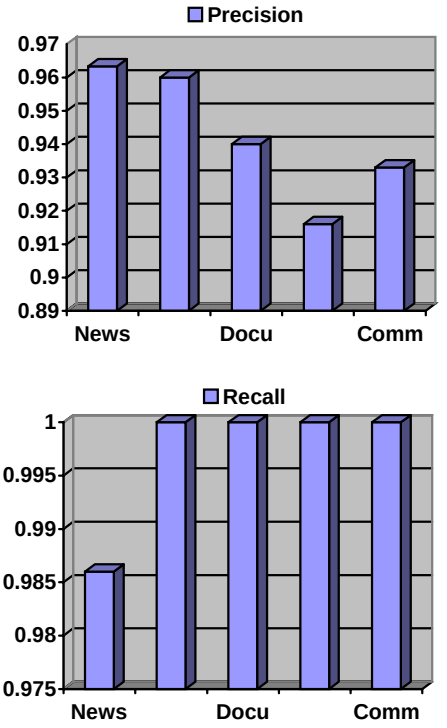
multimedia semantic model such as the multimedia augmented transition network (MATN) model (Chen, Shyu, & Kashyap, 2000).

FUTURE TRENDS

Video shot segmentation is the first step towards automatic annotation and indexing of digital video for efficient browsing and retrieval. It is an active area of research combining the techniques from image processing, computer vision, pattern recognition, etc. Some limitations of the existing work, that imply the future trends, are summarized as follows:

1. *Distinguish gradual transitions from object and camera motions*: While early work focused on shot cut detection, more recent work tries to deal with gradual transition detection. However, although many existing algorithms can confirm the existence of gradual transitions, they still have difficulties in recognizing the different types of gradual transitions due to the following three reasons: (a) the increasing varieties in gradual transition styles; (b) the similar temporal change patterns between gradual transitions and camera/object motions; and (c) the existence of long transitions in which the differences between frames decreases when the transition length increases. In addition to transition detection, the recognition of camera motion is another important issue in video segmentation. Further improvement to these issues can be achieved by combining multi-modal information such as audio and text (closed captioned) information. Moreover, since one single technique cannot capture all the rich information of a video, an integrated approach combining multiple techniques is a possible and practical solution to this problem.

Figure 10. Average results of parameters Precision *and* Recall *for different types of video clips (News, MTV, Documentary, Commercial and Sports)*



2. *Benchmark video sequences and evaluation criteria*: It is critical to develop a unified evaluation criteria and benchmarks for video segmentation and video database management systems that allows the evaluation and comparison of various techniques. The benchmark video sequences should contain various types of videos including enough representatives of different types of object motions, camera operations, and gradual transitions. The evaluation criteria need to be quantified and take into consideration the special requirements of specific application domains.
3. *Adjust the thresholds automatically*: Since there are many thresholds involved in video shot boundary detection methods, especially in those methods combining several techniques, how to adjust the thresholds automatically with respect to different characteristics of different video sequences is a big challenge. The development of the methods that can automatically adjust the thresholds by the self-learning process is desired.
4. *Independence of specific video formats*: Almost all of the "real-time" video shot boundary detection methods are conducted in compressed-domain, especially in

the MPEG format. Some embedded parameters, such as the DC coefficients, can be directly used for shot boundary detection such that the effort of decoding the video data can be reduced. However, this kind of methods are highly encoder-dependent in that different types of compressed video data (MPEG, RealPlayer, etc.) need different techniques for video parsing and shot detection even though they contain the same content. A format-independent technique is more desirable when considering the fast emergence of new compressed video formats.

CONCLUSIONS

In this chapter, an innovative video shot boundary detection method is presented. Our shot boundary detection method uses the unsupervised segmentation algorithm, object-tracking technique, and a matching process that compares the segmentation mask maps between two successive video frames. The unsupervised segmentation algorithm is applied to automatically extract the significant objects (or regions) of interests and the segmentation mask maps of each video frame. The object-tracking technique is employed as a complement to handle the situations of camera panning and tilting without any extra overhead. Experiments on various different types of sample MPEG-1 video clips were performed. The experimental results show that, unlike other methods that use the low-level features of the video frames, our method is not sensitive to the small changes in luminance or color. Also, our method has high precision and recall values. Most importantly, our method can obtain object-level information of the video frames and accurate shot boundary detection, which are very useful for video content indexing.

ACKNOWLEDGMENTS

For Shu-Ching Chen, this research was supported in part by NSF CDA-9711582, NSF EIA-0220562, NSF HRD-0317692, and the office of the Provost/FIU Foundation. For Mei-Ling Shyu, this research was supported in part by NSF ITR (Medium) IIS-0325260.

REFERENCES

Alattar, A.M. (1997). Detecting fade regions in uncompressed video sequences. In *Proceedings of 1997 IEEE International Conference on Acoustics Speech and Signal Processing*, 3025-3028.

Arman, F., Hsu, A., & Chiu, M.-Y. (1993). Image processing on compressed data for large video databases. In *Proceedings of First ACM International Conference on Multimedia*, 267-272.

Chen, S.-C., Shyu, M.-L., & Kashyap, R. L. (2000). Augmented transition network as a semantic model for video data. *International Journal of Networking and Information Systems, Special Issue on Video Data*, 3(1), 9-25.

Chen, S.-C., Sista, S., Shyu, M.-L., & Kashyap, R.L. (1999). Augmented transition networks as video browsing, models for multimedia databases and multimedia information systems. *11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99)*, 175-182, November 9-11.

Chen, S.-C., Sista, S., Shyu, M.-L., & Kashyap, R. L. (2000). An indexing and searching structure for multimedia database systems. *IS&T/SPIE conference on Storage and Retrieval for Media Databases 2000*, 262-270, January 23-28.

Furht, B., Smoliar, S.W., & Zhang, H.J. (1995). *Video and image processing in multimedia systems*. Boston, MA: Kluwer Academic Publishers.

Gall, D.L. (1991). MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(1), 46-58.

Gargi, U., Kasturi, R., & Antani, S. (1998). Performance characterization and comparison of video indexing algorithms. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 559-565.

Gunsel, B., Ferman, A.M., & Tekalp, A.M. (1998). Temporal video segmentation using unsupervised clustering and semantic object tracking. *Journal of Electronic Imaging*, 7(3), 592-604.

Hwang, T.-H., & Jeong, D.-S. (1998). Detection of video scene breaks using directional information in DCT domain. *Proceedings of the 10th International Conference on Image Analysis and Processing*, 882-886.

Lee, S.-W., Kim, Y.-M., & Choi, S.-W. (2000). Fast scene change detection using direct feature extraction from MPEG compressed videos. *IEEE Trans. on Multimedia*, 2(4), 3178-3181.

Nagasaka, A., & Tanaka, Y. (1995). Automatic video indexing and full-video search for object appearances. *In Visual Database Systems II*, 113-127. New York: Elsevier.

Ngo, C.-W., Pong, T.-C., & Chin, R. T. (2000). Motion-based video representation for scene change detection. *Proceedings of the International Conference on Pattern Recognition (ICPR'00)*, 1827-1830.

Pass, G., & Zabih, R. (1999). Comparing images using joint histograms. *ACM Multimedia Systems*, 7(3), 234-240.

Shahraray, B. (1995). Scene change detection and content-based sampling of video sequences. In *Proceedings of SPIE'95, Digital Video Compression: Algorithm and Technologies*, 2419, 2-13, San Jose, CA.

Sista, S., & Kashyap, R.L. (1999). Unsupervised video segmentation and object tracking. *IEEE International Conference on Image Processing*.

Swain, M. J. (1993). Interactive indexing into image databases. In *Proceedings of SPIE Conference Storage and Retrieval in Image and Video Da*tabases, 173-187.

Swanberg, D., Shu, C.F., & Jain, R. (1993). Knowledge guided parsing in video database. In *Proceedings of SPIE'93, Storage and Retrieval for Image and video Databases*, 1908, 13-25, San Jose, CA.

Truong, B. T., Dorai, C., & Venkatesh, S. (2000). New enhancements to cut, fade, and dissolve detection processes in video segmentation. In *Proceedings of the 8th ACM International Conference on Multimedia*, 219-227.

Xiong, W., & Lee, J.C.-M. (1998). Efficient scene change detection and camera motion annotation for video classification. *Computer Vision and Image Understanding*, 71(2), 166-181.

Yeo, B., & Liu, B. (1995). Rapid scene analysis on compressed video. *IEEE Trans. Circuits Systems Video Technol.*, 5(6), 533-544.

Zabih, R., Miller, J., & Mai, K. (1995). A feature-based algorithm for detecting and classifying scene breaks. In *Proceedings of ACM Multimedia '95*, 189-200.

Zhang, H., Kankanhalli, A., & Smoliar, S.W. (1993). Automatic partitioning of full-motion video. *Multimedia System*, 1, 10-28.

Zhang, H., & Smoliar, S.W. (1994). Developing power tools for video indexing and retrieval. In *Proceedings of SPIE'94, Storage and Retrieval for Image and video Databases II*, 2185, 140-149, San Jose, CA.

APPENDIX

URL1. www.ibroxfc.co.uk
URL2. http://hsb.baylor.edu/courses/Kayworth/fun_stuff/
URL3. Cincinnati Dockers, www.cincinnatidockers.com
URL4. www.mormino.net/videos/index.php3