Video Shot Change Detection for Multimedia Database Systems

Shu-Ching Chen and Chengcui Zhang

Technical Report

No. 2001-11

Video Shot Change Detection for Multimedia Database Systems

Shu-Ching Chen Chengcui Zhang

School of Computer Science
Florida International University
Miami, FL 33199, USA

Abstract

The purpose of this report is to present an effective shot change detection method for multimedia database systems using an unsupervised segmentation algorithm and the technique of object tracking based on the *segmentation mask maps*. Compared with other state-of-art techniques such as the method of DC images and some feature based technique, our results have shown that this method can not only perform accurate shot change detection, but also obtain object level information of the video frames, which is very useful for video content indexing and analysis in multimedia databases.

Keywords: shot change detection, segmentation mask map, video indexing, multimedia databases, object tracking.

1. INTRODUCTION

Recently, multimedia information has been made overwhelmingly accessible with the rapid advances in communication and multimedia computing technologies. The requirements for efficiently accessing the mass amounts of multimedia data are becoming more and more important. 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 is popular in many applications such as education and training, video conferencing, video on demand (VOD), news service, and so on. Traditionally, when users want to search certain contents in videos, they need to fast forward or rewind to get a quick overview of interest on the videotape. This is a sequential process and users do not have a chance to choose or jump to a specific topic directly. How to organize video data and provide the visual content in compact forms becomes important in multimedia applications [22]. Therefore, users can browse a video sequence directly based on their interests so that they can get the necessary information quicker and the amount of data transmission can be reduced. Also, users should have the opportunity to retrieve video materials using database queries. Since video data contains rich semantic information, database queries should allow users to get high level content such as *scenes* or *shots*.

Video shot change detection is a fundamental operation used in many multimedia applications such as digital libraries and video on demand, and it must be performed prior to all other processes [15, 25]. Video data can be divided into different shots. A shot is a video sequence that consists of continuous video frames for one action. Shot change detection is an operation that divides video data into physical shots. There are a number of methods for video shot change detection in the literature. The matching process between two consecutive frames

is the essential part of it. Many of them use the low-level global features such as the luminance pixel-wise difference [24], luminance or color histogram difference [18] and edge difference [23] to compare two consecutive frames. Other recent work related to low level features includes the orientation histogram [13]. Especially, Zabih et al.'s [23] edge image based method works well in many cases when the detection with intensity histograms is difficult to work out. However, since luminance or color is sensitive to small change, these low-level features cannot give a satisfactory answer to the problem of shot change detection. For example, in the method of using DC image [21], it uses the luminance histogram difference of DC images, which is very sensitive to luminance changes. Recently, there have been many research work done on the compressed video data domain such as the fast shot change detection [11] and the directional information retrieving [10] by using the discrete cosine transform (DCT) coefficients in MPEG video data. Besides all the above techniques, some research work has been done on the dynamic and adaptive threshold determination problem such as [1, 19], which can be used to enhance the accuracy and robustness of the existing techniques in shot cuts detection.

In this report, focusing on the uncompressed video data, we propose an innovative shot change detection method using an unsupervised image segmentation algorithm and the object tracking technique. By using the image segmentation algorithm, the segmentation mask map of each video frame can be automatically extracted. The segmentation mask map, in another word, can be deemed as the clustering feature map of each frame. In such a way, the pixels in each frame have been grouped into different classes (for example, 2 classes). Then two frames can be compared by checking the difference between their segmentation mask maps. In addition, in order to better handle the situation of camera panning and tilting, the object

tracking technique based on the segmentation results is used as an enhancement to the basic matching process. Since the segmentation results are already available, the cost for object tracking is almost trivial. Moreover, our key frame representation uses the information of the segmentation results such as the bounding boxes and the positions of the segments within that frame. 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 advantages of using unsupervised segmentation and object tracking are:

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

This technical report is organized as follows. In Section 2, we explain the shot change detection method as well as the mechanism of the unsupervised segmentation algorithm and the object tracking technique. In Section 3, experimental results are analyzed to show the effectiveness of the proposed method. Section 4, we review related works in shot change detection. Finally, conclusions are given in Section 5.

2. SHOT CHANGE DETECTION METHOD

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

2.1 Segmentation Information Extraction

In this report, 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 follows this description; while 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 in total four segments within this class (the CDs, for example). Notice that each segment 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 Section 2.2.

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 [3, 5, 4]. 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.

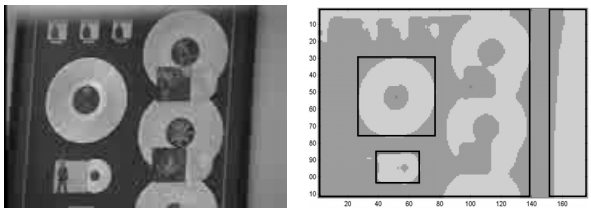


Figure 1: Examples of *classes* and *segments.* The original video frame is on the left and the segmentation mask map of the left frame is on the right.

Suppose there are two classes -- *class1* and *class2*. 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 $(a_{k0}, \ldots, a_{k3})^T$.

$$y_{ij} = a_{k0} + a_{k1} i + a_{k2} j + a_{k3} ij, \ \forall (i, j)\, y_{ij} \in \boldsymbol{c_k} \tag{1}$$

$$\boldsymbol{Y_k} = \Phi\, a_k \tag{2}$$

$$\hat{a}_k = (\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{Y_k} \tag{3}$$

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

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

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

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

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

The problem of segmentation thus becomes the problem of simultaneously estimating the class partition and the parameter for each class. About 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. About 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:

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

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

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.

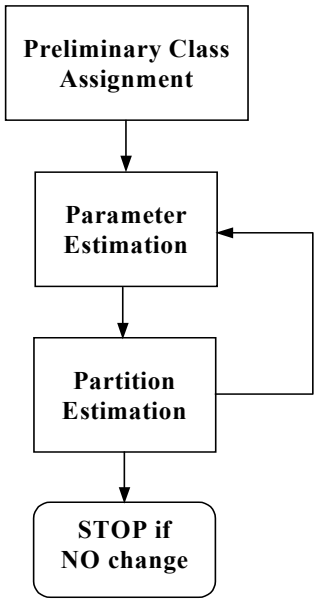


Figure 2: the flowchart of SPCPE algorithm

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 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.

2.2 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.

Figure 3 (a): Example video sequence

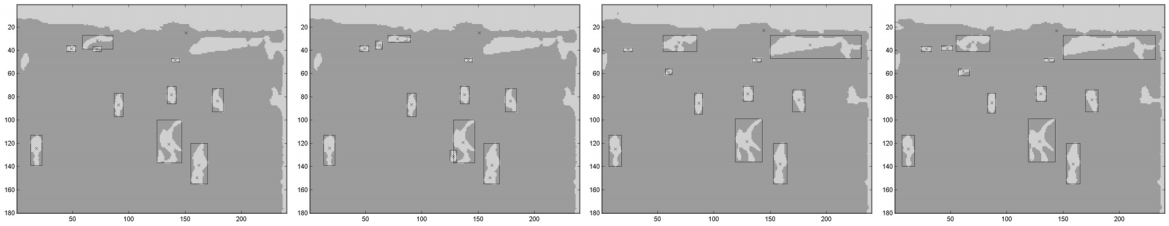


Figure 3 (b): Segmentation mask maps and bounding boxes for a.)

Figure 3: Object tracking

- *Line Merging Algorithm (LMA) for Extracting Segments:*

Unlike the traditional way to do segment extraction such as the *seeding and region growing* method used in [16], we use a computationally simple and fast method called *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 we have only one big segment now. Similarly, at row 8, two lines belong to the same segment because they can be merged with the same line in row 7.

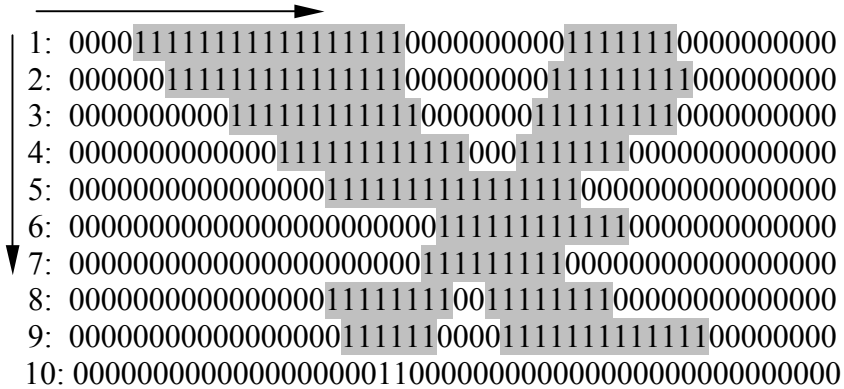


Figure 4: The segmentation mask map.

The pseudo codes for *line merging algorithm (LMA)* are listed below:

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: 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
```

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 indeterminate 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 [16]. In

another word, the Euclidean distances between the centroids of the segments in adjacent frames are used as the criteria to track the related segments. Besides, 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 [6, 7] 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.

2.3 Shot Change 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 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 combine 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.

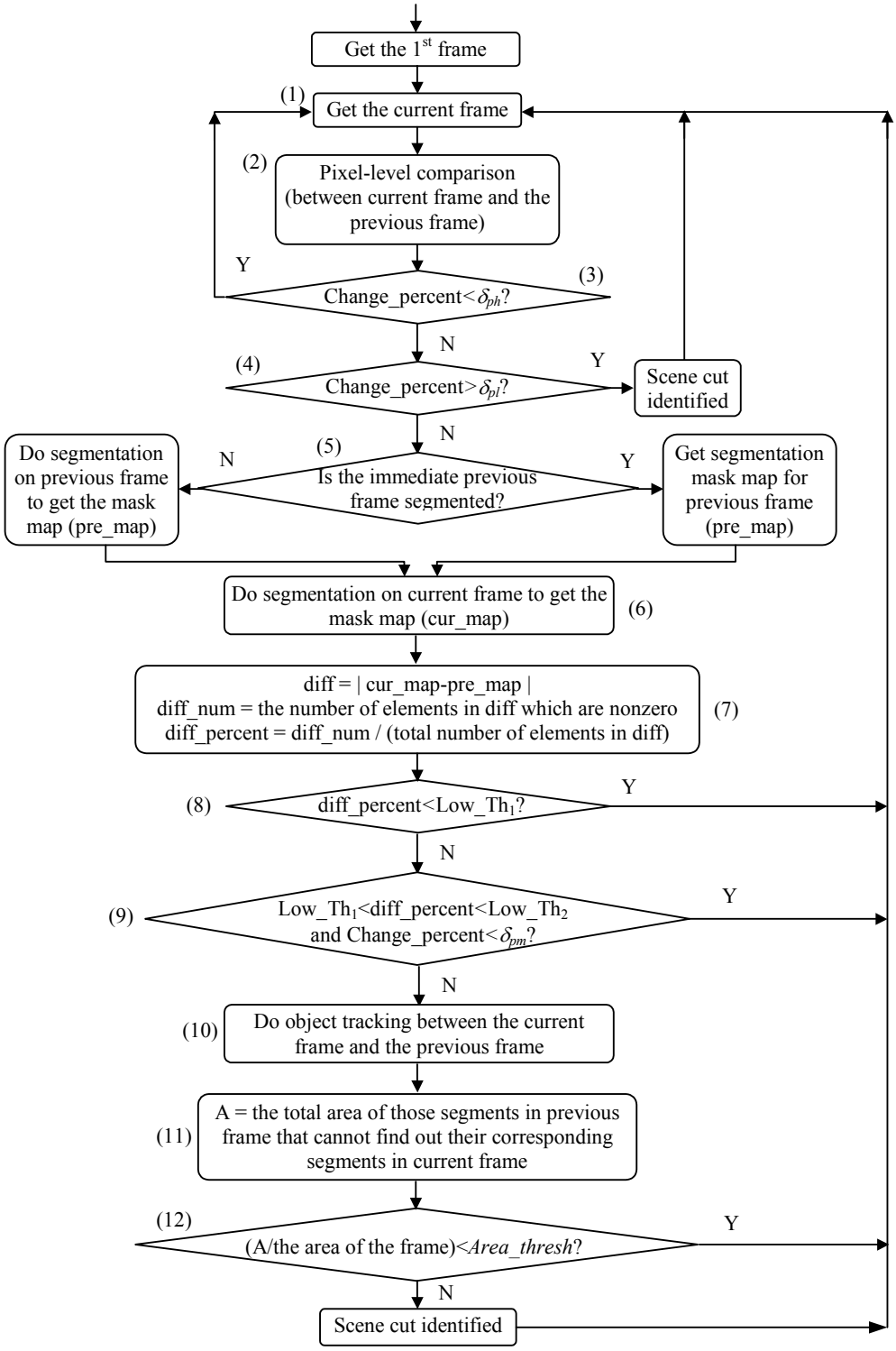


Figure 5: the flowchart of the proposed shot change detection method

Figure 5 shows the flowchart of the proposed shot change 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*$<\delta_{ph}$, 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 change. Go to step 1 and process the next frame (chart box 1).

Else

If $Low\_Th_1 < diff\_percent < Low\_Th_2$ and $change\_percent < \delta_{pm}$ // chart box 9

Not shot change. 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 change. 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.)

3. 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 [26, 27, 28, 29]. Table I gives the statistics of all the video clips used. The experimental results demonstrate the effectiveness of the proposed shot change detection algorithm. Next we will see how the proposed method detects the different types of shot changes that cannot be correctly identified by the traditional pixel-level comparison method.

Table I: 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

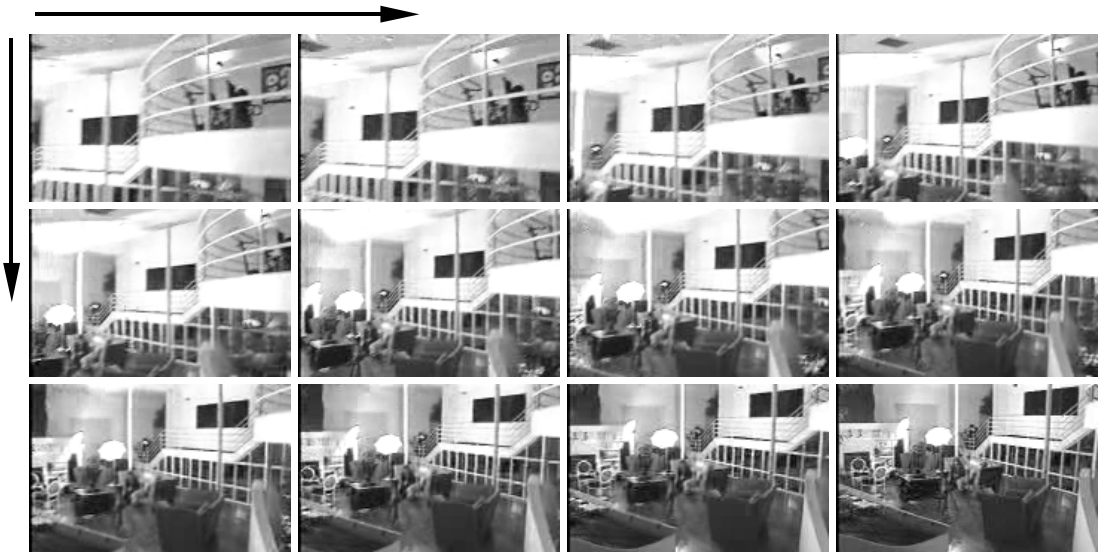


Figure 6 (a): An example video sequence for camera panning and tilting. The temporal order of the sequence is from the top-left to the right-bottom.

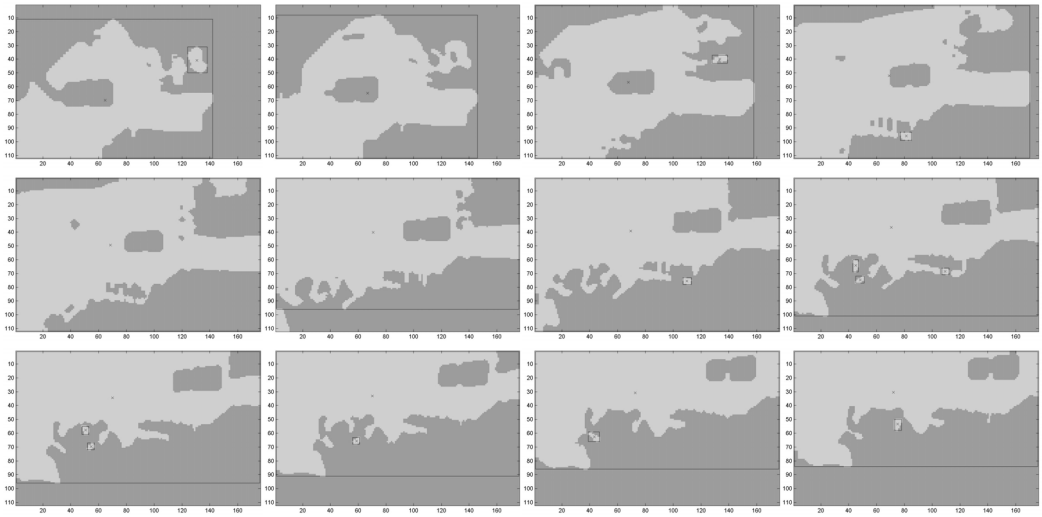


Figure 6 (b): The corresponding segmentation mask maps for the video sequence shown in a).

Figure 6: An example video sequence for camera panning and tilting.

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 moves and turns 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 change if the segments in two successive frames can be tracked and matched well according to the object tracking method mentioned in Section 2.2.

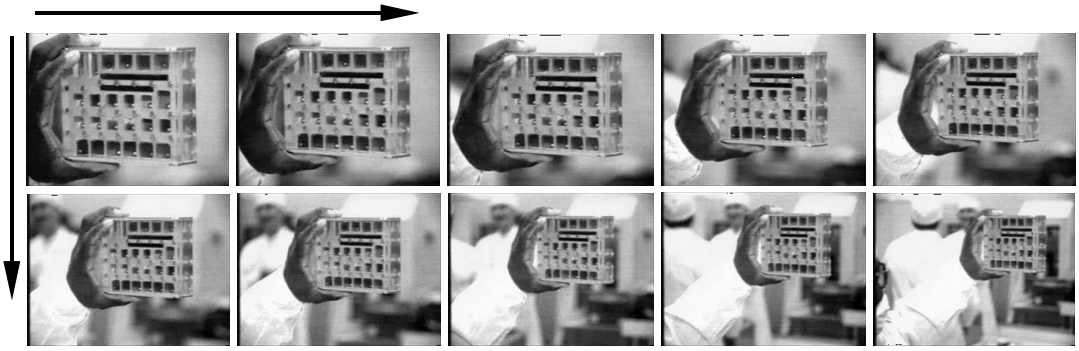


Figure 7 (a): An example video sequence of zooming out. The temporal order of the sequence is from the top-left to the right-bottom.

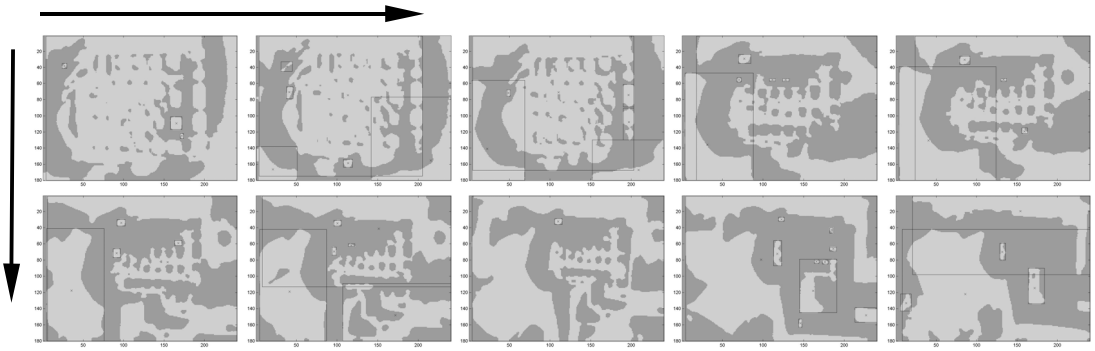


Figure 7 (b): The corresponding segmentation mask maps for the video sequence shown in a).

Figure 7: An example video sequence of zooming out.

CASE 2: Zoom In and Zoom Out

Figure 7 gives an example 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 out an example 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.

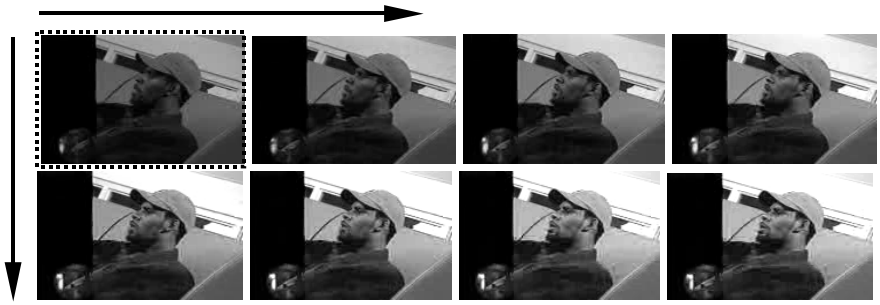


Figure 8: An example video sequence of fading in. The frame with dotted border is shot cut detected by the proposed method.

In Figure 9, a more fancy example 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 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.

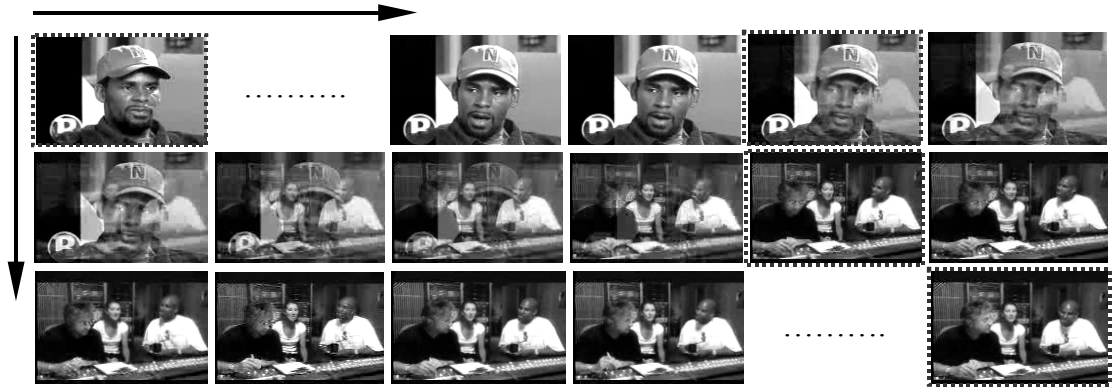


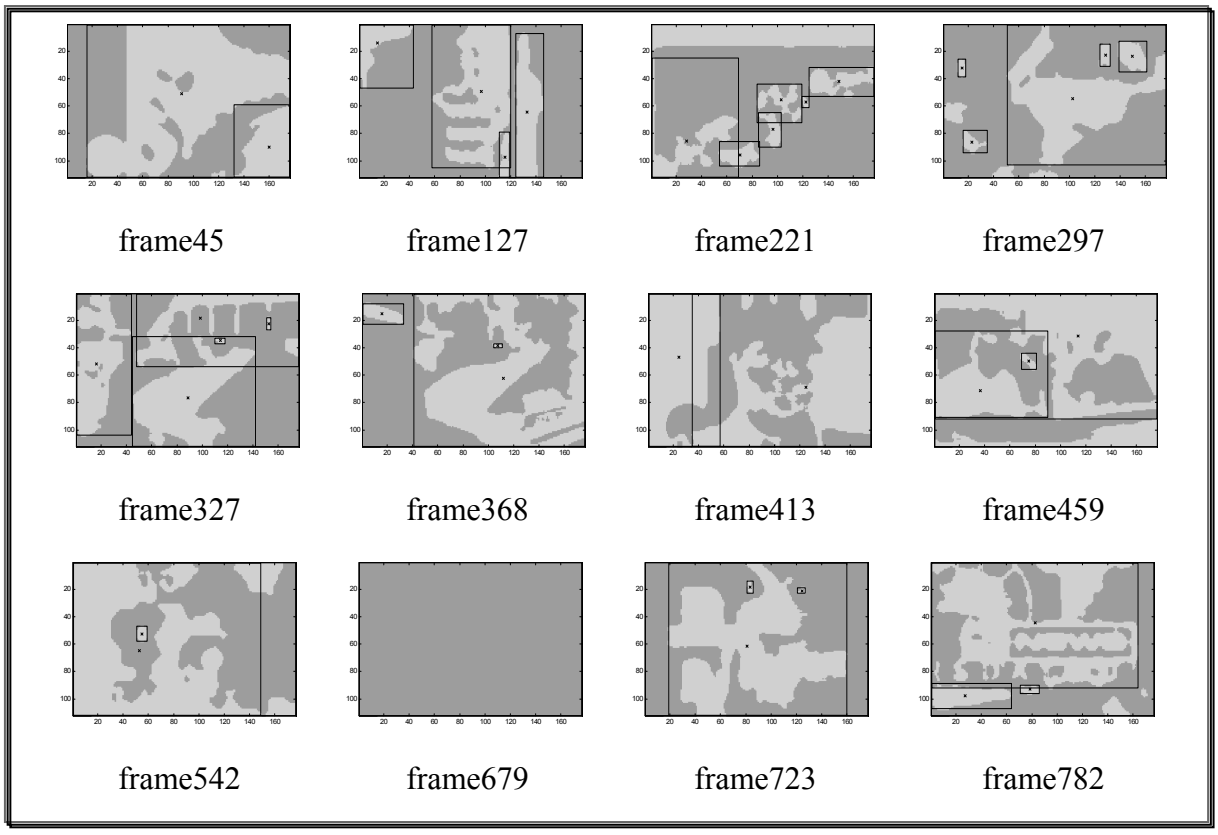
Figure 9: An example video sequence of continuously transforming from one shot to another shot. The frames with dotted borders are shot cuts detected by the proposed method.

Figure 10 gives an example shot cuts detection results for an ABC 7 NEWS video. Figure 10(a) shows the original video frames that have been detected as the shot cuts, and Figure 10(b) shows the example segmentation mask maps for the shot cuts in Figure 10(a). The performance is given in terms of *precision* and *recall* parameters. N_C means the number of correct shot change detections, N_E means the number of incorrect shot change detections, and N_M means the number of missed shot detections.

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

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

a). The example shot cuts for the ABC 7 NEWS video sequence.



b). The example segmentation mask maps for the shot cuts in a).

Figure 10: The example shot cuts and their segmentation mask maps.

The summary of the proposed method is shown in Table II and Figure 11 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 [11] and DC method in [21]. 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 II, the precision values for sports and commercial videos are a little lower (but still above ninety 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 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 II, 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. As can be seen from Figure 10(b), 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 multimedia semantic model such as the multimedia augmented transition network (MATN) model [4].

Table II: The 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%

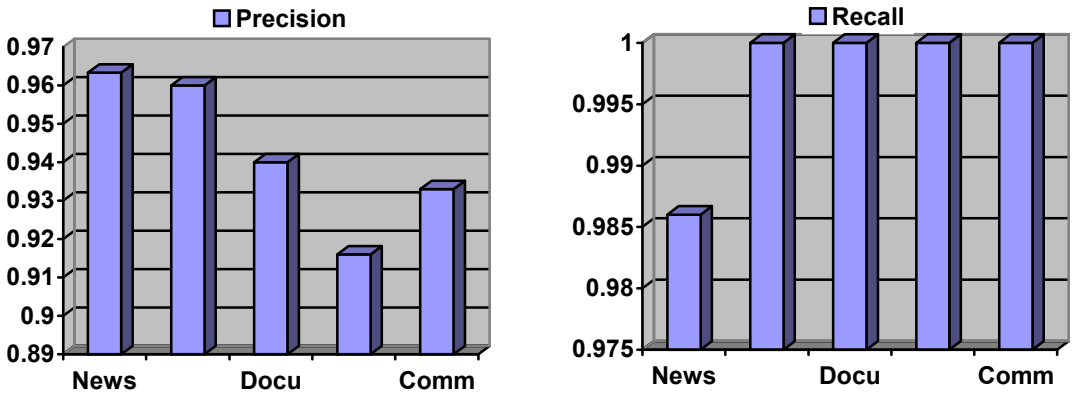


Figure 11: The average results of parameters *Precision* and *Recall* for different types of video clips (News, MTV, Documentary, Commercial and Sports).

4. RELATED WORK

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. Usually, the existing techniques for shot change detection

can be grouped into the methods operating on uncompressed data and those operating on compressed data.

Gargi et al. [8] gave a survey on video indexing, as well as the video segmentation techniques used in uncompressed data domain. In uncompressed domain, the shot change detection algorithms process uncompressed video, and a similarity measure between successive frames is defined. Algorithms in this category include [12, 24]. The basic idea in pixel-level comparison is to compute the differences in values of corresponding pixels between two successive frames. It uses one threshold to tell if the value of the corresponding pixels has changed or not, and uses another threshold 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. In our method, we embed this simple method into the techniques of object tracking and image segmentation in order to overcome its shortcomings, and at the same time to reduce the computation. Another kind of comparison technique used in uncompressed domain is the block-wise comparison. Instead of pixel-by-pixel matching, block-wise comparison methods use the local characteristics (such as the mean and variance intensity values) of blocks to reduce the sensitivity to object and camera movements. In this kind of approaches, 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, since the mean and variance values of a block are not good enough to represent the block's characteristics, it is highly possible to introduce incorrect matching between two blocks that have the same mean

and variance values but with totally different contents [20]. In our method, we partially adopt the idea of block matching in object tracking technique. Instead of dividing the frame into fixed size of blocks absolutely, we employ an innovative image segmentation method 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.

A further improved method in order to reduce sensitivity to camera and object movements is the histogram-based comparison. Since the object moving between two successive frames is relatively small, their histograms will not have big differences. It is more robust to small rotations and slow variations [14, 17]. But as we know, the histogram-based method has its potential problems. That is, two successive frames will possibly have the similar histograms but with different contents. Another approach based on the low-level features of images is proposed by Zabih et al. [23]. Their proposed approach used the intensity edges between successive frames to detect shot cuts. However, as the authors have pointed out, the weaknesses of their approach are the false positives due to the limitations of the edge detection method.

In addition to the research on the methods of similarity measures between successive frames, there have been other works towards solving the problem of threshold estimation. In [9], the unsupervised clustering algorithm proposed a generic technique that does not need threshold setting and allows multiple features to be used simultaneously. Another interesting work done by Truong et al. in [19] proposed an adaptive threshold determination method that is to reduce the artifacts created by noise and motion in shot change detection.

There are many shot change detection algorithms in compressed domain, especially in MPEG format videos. Since the encoded video stream already contains many features such as the DCT coefficients and motion vectors, it is suitable for video shot change detection. In [2], it used the DCT (discrete cosine transform) coefficients of I frames as the similarity measure between successive frames. Yeo and Liu [21] used the dc-images 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. In [10], Hwang and Jeong utilized the changes of directional information in the DCT domain to detect the shot breaks automatically. Lee et al. [11] further improved the DCT coefficient-based method. They used the binary edge maps as a representation of the key frames so that two frames can then be compared by calculating a correlation between their edge maps. The advantage of this method is that it gives directly the edge information such as orientation, strength, and offset from the DCT coefficients.

5. CONCLUSIONS

In this report, we proposed an innovative shot change detection method using the unsupervised segmentation algorithm and object tracking technique, and showed the precision and recall performance using the different types of sample MPEG-1 video clips. The key idea of the matching process in shot change detection is to compare the segmentation mask maps between two successive video frames, which is simple and fast. In addition, the object tracking technique is employed as a complement to handle the situations of camera panning and tilting without any extra overhead. Unlike many methods using the low-level features of the video frames, the proposed method is not sensitive to the small changes in luminance or color. Moreover, it has high precision and recall values as shown in our experiments.

6. ACKNOWLEDGEMENT

This research was supported in part by NSF CDA-9711582.

7. REFERENCES

[1] A. M. Alattar, "Detecting Fade Regions in Uncompressed Video Sequences," in *Proceedings of 1997 IEEE International Conference on Acoustics Speech and Signal Processing*, pages 3025-3028, 1997.

[2] F. Arman, A. Hsu, M.-Y. Chiu, "Image Processing on Compressed Data for Large Video Databases," in *Proc. First ACM Intl. Conference on Multimedia*, 1993, pp. 267-272.

[3] S.-C. Chen, S. Sista, M.-L. Shyu, and R. L. Kashyap, "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),* pp. 175-182, November 9-11, 1999.

[4] S.-C. Chen, M.-L. Shyu, and R. L. Kashyap, "Augmented Transition Network as a Semantic Model for Video Data," *International Journal of Networking and Information Systems*, Special Issue on Video Data, vol. 3, no. 1, pp. 9-25, 2000.

[5] S.-C. Chen, S. Sista, M.-L. Shyu, and R. L. Kashyap, "An Indexing and Searching Structure for Multimedia Database Systems," *IS&T/SPIE conference on Storage and Retrieval for Media Databases 2000*, pp. 262-270, January 23-28, 2000.

[6] B. Furht, S. W. Smoliar, and H. J. Zhang, Video and Image Processing in Multimedia Systems, Kluwer Academic Publishers, 1995.

[7] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Communications of the ACM* 34(1): 46-58, April 1991.

[8] U. Gargi, R. Kasturi, S. Antani, "Performance Characterization and Comparison of Video Indexing Algorithms," in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, 559-565, 1998.

[9] B. Gunsel, A. M. Ferman, A. M. Tekalp, "Temporal Video Segmentation Using Unsupervised Clustering and Semantic Object Tracking," *Journal of Electronic Imaging*, 7(3), pp. 592-604, 1998.

[10] T.-H. Hwang and D.-S. Jeong, "Detection of Video Scene Breaks Using Directional Information in DCT Domain," *Proceedings of the 10th International Conference on Image Analysis and Processing*, 1998.

[11] S.-W. Lee, Y.-M. Kim, and S.-W. Choi, "Fast Scene Change Detection using Direct Feature Extraction from MPEG compressed Videos," *IEEE Trans. on Multimedia*, vol. 2, No. 4, pp. 240-254, Dec. 2000.

[12] A. Nagasaka, Y. Tanaka, "Automatic Video Indexing and Full-video Search for Object Appearances," in Visual Database Systems II, pp. 113-127, Elsevier, 1995.

[13] C.-W. Ngo, T.-C. Pong and R. T. Chin, "Motion-Based Video Representation for Scene Change Detection," *Proceedings of the International Conference on Pattern Recognition (ICPR'00)*, vol. 2, pp. 827-830, 2000.

[14] G. Pass, R. Zabih, "Comparing Images Using Joint Histograms," *ACM Multimedia Systems*, 1999.

[15] B. Shahraray, "Scene change detection and content-based sampling of video sequences," in *Proc. SPIE'95, Digital Video Compression: Algorithm and Technologies*, vol. 2419, pp. 2-13, San Jose, CA, 1995.

[16] S. Sista and R. L. Kashyap, "Unsupervised Video Segmentation and Object Tracking," *IEEE Int'l Conf. on Image Processing*, 1999.

[17] M. J. Swain, "Interactive Indexing into Image Databases," in *Proc. SPIE Conference Storage and Retrieval in Image and Video Da*tabases, pp. 173-187, 1993.

[18] D. Swanberg, C. F. Shu, and R. Jain, "Knowledge guided parsing in video database," in *Proc. SPIE'93, Storage and Retrieval for Image and video Databases,* vol. 1908, pp. 13-24, San Jose, CA, 1993.

[19] B. T. Truong, C. Dorai, S. Venkatesh, "New Enhancements to Cut, Fade, and Dissolve Detection Processes in Video Segmentation," in *Proceedings of the 8th ACM International Conference on Multimedia*, pp. 219-227, 2000.

[20] W. Xiong, J. C.-M. Lee, "Efficient Scene Change Detection and Camera Motion Annotation for Video Classification," *Computer Vision and Image Understanding*, 71(2), pp. 166-181, 1998.

[21] B. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Trans. Circuits Systems Video Technol*., vol. 5, no. 6, pp. 533-544, 1995.

[22] B. Yeo and M. Yeung, "Retrieving and visualization video," *Comm. of the ACM*, vol. 40, no. 12, pp. 43-52, December 1997.

[23] R. Zabih, J. Miller, and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks," *in Proc. ACM Multimedia '95*, 1995, pp. 189-200.

[24] H. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," *Multimedia System*, vol. 1, pp. 10-28, 1993.

[25] H. Zhang and S. W. Smoliar, "Developing power tools for video indexing and retrieval," in *Proc. SPIE'94, Storage and Retrieval for Image and video Databases II*, vol. 2185, pp. 140-149, San Jose, CA, 1994.

[26] www.ibroxfc.co.uk

[27] http://hsb.baylor.edu/courses/Kayworth/fun_stuff/

[28] www.cincinnatidockers.com

[29] www.mormino.net/videos/index.php3