ICICS-PCM 2003
15-18 December 2003
Singapore

PixSO: A System for Video Shot Detection

Chengcui Zhang[1], Shu-Ching Chen[1], Mei-Ling Shyu[2]

[1]School of Computer Science, Florida International University, Miami, FL 33199, USA
[2]Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33124, USA

Abstract

In this paper, a system called PixSO (Pixel-level comparison, Segmentation and Object Tracking) is presented for effective shot change detection using an unsupervised object segmentation algorithm and the technique of object tracking based on the segmentation mask maps. The detection method was tested on TV news, commercial, sports and documentary video sequences which contain different types of shots having different object and camera motions. The Our results have shown that the PixSO system can not only produce 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.

1. Introduction

In order to efficiently manage and retrieve the growing amount of digital video information, a video shot change detection method is pre-required to ease the content-based access to video library. A video shot is a video sequence that consists of continuous video frames for one camera action. Besides the natural shot cuts, there are also shot boundaries caused by special edit effects (fade in/out, etc.) and camera motions (panning, tilting, etc.) as well as object motions.

There are a large 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 [5], luminance or color histogram difference [7] and edge difference [6] to compare two consecutive frames. However, since luminance or color is sensitive to small changes, 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 [4], it uses the luminance histogram difference of DC images, which is very sensitive to luminance changes. Other recently proposed methods focused on long transition detection and temporal slice analysis can be found in [14] and [15], respectively. Recently, there also have been many research work done on the compressed video data domain such as the fast shot change detection [3] and the directional information retrieving [2] by using the discrete cosine transform (DCT) coefficients in MPEG video data.

While parsing the video data for analysis is time consuming, it is expected to produce as much information as it can in one pass for efficiency purpose. For example, the object extraction and key frame selection can be done together with the video shot segmentation. However, to our best knowledge, there is little work done in the literature trying to automate the process of obtaining the object level information in video scenes while doing video shot segmentation. The work proposed in [1] tries to employ object tracking into the scene cut detection, but the detection and tracking of the semantic objects of interest need to be specified manually, and a bunch of template frames containing the semantic objects were used for training purpose, which is not feasible for automatic and unsupervised processing. Moreover, the method proposed in [1] is domain-specific (news videos) instead of a more general framework. Generalized block matching methods allowing affine transformations in intensity have been used in [1] for object tracking purpose. However, affine transformation is still sensitive to luminance changes. In this paper, we try to apply an efficient unsupervised segmentation method to extract the semantic objects from video data without user intervention, which is totally independent of specific application domain.

The extraction of regions/objects of interest is regarded as the basis for extracting high level semantic meaning. In this paper, focusing on the uncompressed video data domain, we propose and implement an innovative shot change detection system called PixSO (Pixel-level comparison, Segmentation and Object Tracking) using an unsupervised image segmentation algorithm and the object tracking technique. By using the unsupervised image segmentation algorithm, the significant objects or regions of interests as well as the segmentation mask map of each video frame can be automatically extracted. The segmentation mask map, in other 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 computation cost for object tracking is almost trivial compared to those manual template-based object tracking methods. For efficiency purpose, we also apply the pixel-level comparison for pre-screening in addition to segmentation and object tracking, which is what 'PixSO' (Pixel level comparison, Segmentation and Object Tracking) stands for. The advantages of using unsupervised segmentation and object tracking are summarized below:

1. Extraction of regions/objects is fully unsupervised and efficient, without any user interventions or domain-dependent knowledge.
2. The algorithm for comparing two frames is simple and fast based on the segmentation results.
3. This method is not sensitive to small changes in luminance or contrast.
4. The unsupervised object level segmentation results can be further used for video indexing and content analysis.

This paper is organized as follows. In Section 2, we explain the details of the proposed PixSO system for video shot detection as well as the unsupervised segmentation algorithm and the object tracking technique. In Section 3, experimental results are analyzed and compared with other method such as standard histogram to show the effectiveness of the proposed method. Finally, conclusions are given in Section 4.

2. PixSO: System Description

2.1 Segment Information Extraction

Unsupervised segmentation: In this paper, we use an unsupervised segmentation algorithm called SPCPE (*Simultaneous Partition and Class Parameter Estimation)* [9] to partition the video frames. 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. This is illustrated in Figure 1. The gray areas and dark areas in the segmentation mask map (shown on the right side of Figure 1) represent two different classes respectively. Considering the gray class, there are in total two segments (namely the fish and the rod) within this class. Notice that each segment is bounded by a bounding box and has a centroid, which are the results of segment extraction.

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 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 $(a_{k0}, \ldots, a_{k3})^T$.

$$y_{ij} = a_{k0} + a_{k1}i + a_{k2}j + a_{k3}ij, \ \forall(i,j)\ y_{ij} \in c_k$$
$$Y_k = \Phi\, a_k; \quad \hat{a}_k = (\Phi^T \Phi)^{-1} \Phi^T Y_k$$

The best partition is estimated as that which maximizes the a posteriori probability (MAP) of the partition variable given the image data Y.

Figure 2 illustrate the basic workflow of SPCPE. 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.

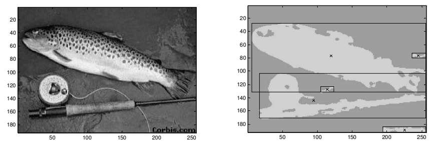

Figure 1: Examples of *classes* and *segments.*

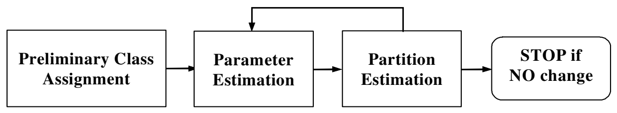

Figure 2: The flowchart of SPCPE algorithm.

Although SPCPE algorithm can handle multiple classes (more than two), we just use two classes in segmentation since two classes are efficient and good enough for our purpose in this application domain.

Efficiency improvements: In order to apply the object extraction into shot detection, the segmentation step has to be efficient. In this study, we adopt the following strategies to achieve computation efficiency for extracting objects from a video sequence:

♦ In order to achieve computation efficiency, we use the incremental computation together with parallel computation to speed up the clustering process. The basic idea of incremental computation is to compute the class parameters at the $(k+1)^{th}$ iteration using the intermediate results at the k^{th} iteration rather than calculate it from scratch, thus to reduce the computation significantly. To further improve the speed, the parallel computation is also applied on sub-images by using MPI (Message Passing Interface) and SPMD (Single Processor/Multiple Data) on Cluster Computing.

♦ Another strategy is that, it is not necessary to wait until there is no further change in the class partition. Instead, when the percent of pixels that change their class labels is less than a threshold (say 5%), the class partition can be deemed stable so that the iteration stops.

♦ Further, since the consecutive frames in video sequences are closely related in contents, incorporating the partition of the previous frame as the initial condition while partitioning the current frame can greatly reduce the computation cost up to 90%.

As a result, the combined speed-up factor can achieve 100~200. The time for segmenting one video frame ranges from 0.03~0.12 sec. Since a pre-screening step based on pixel comparison is used to filter out most of the video frames, the number of frames that need to do segmentation is small.

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. 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. 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 the 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 [10] 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.

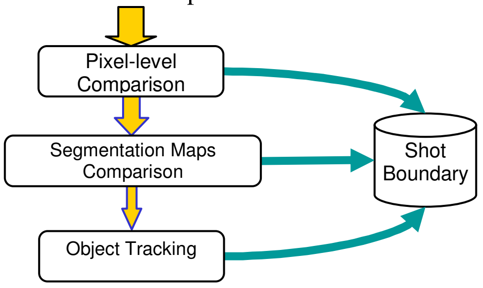

Figure 3: The workflow of the proposed method.

2.3 Shot Change Detection Mechanism

As shown in Figure 3, our proposed PixSO system combines three main techniques together, namely 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 object moving. On the other hand, the proposed segmentation and object tracking techniques are much less sensitive to the above factors. In PixSO, we use the pixel-level comparison for pre-screening. By applying a strict threshold for the percentage of changed pixels, we want to make sure that it will not introduce any incorrect shot cuts that are falsely identified by pixel-level comparison. The advantage to combining the pixel-level comparison is that it can further 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.

The steps are given in the following:

```
1. Do pixel-level comparison between the currently
   video frame and the immediate preceding frame.
   Let the percentage of change be change_per and
   the variance of the pixel-level differences be
   change_var. Check these two variables.
   If the current frame is not identified as a
   shot cut, which means that $change\_per<\delta_{ph}$ or
   $change\_var<\delta_{v}$, then go on to process the next
   video frame. Otherwise go to step 2.
   (The purpose of checking change_var is to pre-
   screen the fade in and fade out situations
   because they usually result in high change_per
   and low change_var. Although object tracking
   can deal well with both of the situations as
   you will see in Section 3, by conducting this
   will reduce the number of frames that need
   segmentation.)
2. If $change\_per>\delta_{pl,}$ the current frame is
   identified as a shot cut. Go to step 1 and
   process the next frame.
   If $change\_per<=\delta_{pl,}$ go to step 3.
3. Do the segmentation on the previous frame only
   if the previous frame has never been segmented.
   If the previous frame has been segmented
   before, obtain its segmentation mask map
   directly and use it as the initial partition
   for segmenting the current frame. 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.
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 nonzero elements in
   diff;
   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.
5. Check the variable diff_percent.
   If $diff\_percent < Low\_Th_1$
      Not shot change. Go to step 1 and process
      the next frame.
   Else
      If $Low\_Th_1<diff\_percent<Low\_Th_2$ and
      $change\_percent<\delta_{pm}$
        Not shot change. Go to step 1 and process
        the next frame.
      Else
        Tracking object 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.
        If (A/the area of the frame)<Area_thresh
          Not shot change. Go to step 1 and
          process the next frame.
        Else
          The current frame is identified as shot
          cut.
          Go to step 1 and process the next frame.
        End if;
      End if;
   End if.
```

(Here, δ_{ph}, δ_{pl}, δ_{v}, δ_{pm}, Low_Th_1 and Low_Th_2 are threshold values for variables *change_percent* and *diff_percent* and they 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), music MTV video, commercial video, documentary video, and sports video such as the soccer game [11]-[12]. The average size of each frame in the sample video clips is 170 rows and 240 columns. Table I gives the statistics of five example video clips from five different video types. When choosing the method for the comparative study, we select the color-histogram based method proposed in [7] because it is recognized as a well-balanced method and has good overall performance [13]. In this study, we present a comparison of our algorithm with this method, with the view to understand the limits faced by the two different methods.

Table I: Video data used for experiments

Name	Type	Number of Frames	Number of Shots
V1	News	1262	5
V2	MTV	886	25
V3	Commercial	1294	29
V4	Sports	750	5
V5	Sports	1715	16
V6	Sports	1405	14
V7	Documentary	15326	113
V8	Documentary	1798	18

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.

The summary of the PixSO system compared with the method proposed in [7] is shown in Table II via the *precision* and *recall* parameters. In our experiments, the overall values of *recall* and the *precision* are both above ninety percent. As seen in Table II, the *recall* results for the PixSO system seem more stable and promising because most of the *recall* values are 100 percent, while the histogram-based method tends to miss more shot cuts. Another observation is that, by using the proposed method, the precision values for MTV and commercial videos are a little lower than other types of videos because there are lots of fast movements and fancy transformation between successive frames. We also realize that the histogram-based method has the advantages in handling sports videos because a histogram is less sensitive to the object motions. However, both of them suffer from the fast object movements together with the camera panning and tilting. Also, the long gradual transitions (more than 2 seconds) are difficult to identify.

Table II: The Precision and Recall Parameters

Type	Name	PixSO		Color Histogram	
		Precision	Recall	Precision	Recall
N	V1	100%	100%	100%	80%
M	V2	92%	95%	69%	56%
C	V3	82%	100%	71%	53%
S	V4	100%	100%	60%	60%
S	V5	80%	100%	85%	88%
S	V6	79%	92%	86%	85%
D	V7	99%	98%	86%	90%
D	V8	82%	85%	70%	80%
Overall		**92.2%**	**96.7%**	**80.6%**	**79.3%**

As mentioned before, the method of using low-level features is very sensitive to luminance and color change, but our segmentation-based mechanism is not. Figure 4 gives an example video sequence for shots fading in. Figure 4(a) is the original video sequence and Figure 4(b) shows the corresponding segmentation mask maps for (a). As seen from Figure 4(b), the segmentation results are very stable while changing the luminance and contrast. This is a good example to show that the proposed segmentation together with object tracking technique is not sensitive to luminance changes. According to our test, the proposed method is very effective in detecting fade in/out and abrupt shot cuts.

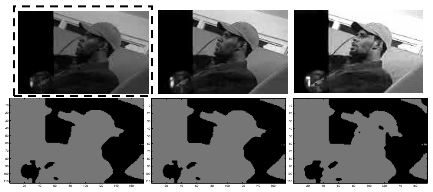
Figure 4: An example video sequence for fading in as well as its corresponding segmentation mask maps.

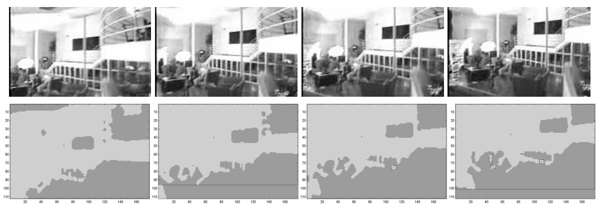
Figure 5: An example video sequence for camera panning and tilting as well as its corresponding segmentation mask maps.

Figure 5 gives an example of the camera panning while tilting. 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 can be seen from Figure 5, 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. Hence, 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.

It should be pointed out that even though 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 observation 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 so that the computation brought by segmentation has been significantly alleviated. Our current system can achieve near real-time processing for video sequences with small-size video frames. However, it is highly possible to directly apply the segmentation method on the down sampled video frames, which will be investigated in our future work. Moreover, the process produces not only the shot cuts, but also the object level segmentation results. Each detected shot cut frame is selected as a key frame and has been modeled by the features of its segments such as the bounding boxes and centroids. Although it is not practical to automatically identify all the objects-of-interest, tremendous manual efforts can be alleviated with aid of unsupervised segmentation and object tracking. Based on the extracted object information, we can further structure the video content using some existing multimedia semantic model such as the multimedia augmented transition network (MATN) model [8].

4. Conclusions and Future Work

In this paper, we presented an innovative shot change detection system called PixSO 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. By using the pixel-level comparison in pre-screening, the key idea of the matching process in shot change detection is to compare the segmentation mask maps between two successive video frames when necessary, 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 with little overhead. Unlike many methods using the low-level features of the video frames, the PixSO system is not sensitive to the small changes in luminance or color. Moreover, it has high precision and recall values as shown in our experiment results. Although this paper is focusing on the video segmentation in uncompressed data domain, it can be easily applied to the compressed data domain. For example, the proposed algorithm can operate directly on the DC image, which is a small fraction of the compressed data and can be easily extracted without full frame decompression [4].

Acknowledgement

For Shu-Ching Chen, this research was support in part by NSF EIA-0220562 and NSF HRD-0317692. For Mei-Ling Shyu, this research was support in part by NSF ITR (Medium) IIS-0325260.

References

[1] B. Gunsel, A. M. Ferman, and A. M. Tekalp, "Temporal Video Segmentation Using Unsupervised Clustering and Semantic Object Tracking," Journal of Electronic Imaging, 7(3), pp. 592-604, 1998.
[2] T.-H. Hwang and D.-S. Jeong, "Detection of Video Scene Breaks Using Directional Information in DCT Domain," Proc. the International Conference on Image Analysis and Processing, pp. 882-886, 1999.
[3] 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.
[4] D. Swanberg, C. F. Shu, and R. Jain, "Knowledge Guided Parsing in Video Database," Proc. SPIE'93, Storage and Retrieval for Image and video Databases, vol. 1908, pp. 13-24, San Jose, CA, 1993.
[5] 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.
[6] R. Zabih, J. Miller, and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks," Proc. ACM Multimedia '95, pp. 189-200, 1995.
[7] H. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic Partitioning of Full-Motion Video," Multimedia System, vol. 1, no. 1, pp. 10-28, 1993.
[8] 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.
[9] 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.
[10] D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, 34(1), pp. 46-58, April 1991.
[11] http://www.ibroxfc.co.uk
[12] http://hsb.baylor.edu/courses/Kayworth/fun_stuff/
[13] R. Lienhart, "Comparison of Automatic Shot Boundary Detection Algorithms," in Image and Video Processing VII 1999, Proc. SPIE 3656-29, pp. 290-301, Jan. 1999.
[14] W. J. Heng, "Shot Boundary Refinement for Long Transition in Digital Video Sequence," IEEE Trans. On Multimedia, Vol. 4, No. 4, pp. 434-445, December 2002.
[15] C.-W. Ngo, T.-C. Pong, and H. Zhang, "On Clustering and Retrieval of Video Shots through Temporal Slices Analysis," IEEE Trans. On Multimedia, vol. 4, no. 4, pp. 446-458, December 2002.