REAL TIME TEMPORAL SEGMENTATION OF COMPRESSED AND UNCOMPRESSED DYNAMIC COLOUR IMAGE SEQUENCES

Sébastien Lefèvre[1,2], Jérôme Holler[1], Nicole Vincent[1]

[1]Laboratoire d'Informatique
Université de Tours / E3i
64, Avenue Jean Portalis - 37200 Tours - France

[2]Atos Services
19, rue de la Vallée Maillard - BP 1311 - 41013 Blois Cedex - France

e-mail : lefevre@univ-tours.fr, vincent@univ-tours.fr

Abstract: We present in this paper a new method to segment compressed or uncompressed image sequences in the temporal domain. The processed images are colour images and the sequence is dynamic (camera is moving). The shot change detection will be used as a pre-processing step for object tracking. So it has to be performed in real time. We use low resolution images to satisfy this constraint. In order to avoid illumination changes effects, the colour space is changed from RGB to HSV, where H and S are used to compare two successive frames. Detection of cuts and of other effects are done thanks to tracking of a inter-frame difference value and comparison with adaptative threshold.

Keywords

Shot change detection, Video parsing, Real time

1 Introduction

Multimedia data is more and more used in current applications. Many of these applications need tools to segment image sequences in the temporal domain. This is usually achieved thanks to detection of cuts or others more complex effects (fade, wipe, dissolve...). Segmentation is a compulsory step in many applications such as management of multimedia databases, automatic creation of abstracts from movies or TV sequences, compression of video sequences, or pre-processing for real time object tracking in dynamic scenes.

We are particularly concerned with the last application, that is to say real time object tracking. Algorithms for object tracking are more powerful and efficient if a good pre-processing is done. It means that we have to perform a shot change detection (camera change) to prevent the tracking algorithm from continuing tracking objects, which would be useless. The real time constraint on the tracking algorithm has to be extended to the temporal segmentation.

First some classical solutions for temporal segmentation will be presented. Then we will introduce our proposed method and detail the main steps of the algorithm. Finally some results will help us to show the robustness and efficiency of the presented method.

2 Some classical solutions

Most of these already published methods could be categorised under the following terms : pixel difference, histogram comparison, movement estimation. These different categories are presented below. For a more complete review, see ([1], [2], [4]).

Pixel difference methods detect a shot change thanks to calculating the difference between pixels of frame at time t and frame at time $t-1$. If the number of different pixels is greater than a fixed threshold, then we can assume that there is a cut or some other effect. These kinds of methods have several limitations, mainly due to noise or too an important movement in the scene.

Histogram based methods work on the histograms of consecutive frames. After calculation of the two histograms has been performed, difference between them is obtained and compared with some threshold. As in pixel based methods, a shot change is detected if the difference is greater than the threshold. The main problem of these methods is the possibility for two consecutive frames to be completely different but to have quite the same histogram. Then the cut is not detected.

Movement estimation methods use movement information as main criterion to detect shot changes. Optical flow is estimated on all the pixels of the image t, and then compared with the one obtained from image $t-1$. If too many movements are different between consecutive frames, then a shot change is detected. These methods are often unable to process video sequences in real time, because different movement estimation is time consuming.

Even if the different methods presented above have their own pros and cons, they are all limited due to fixed threshold. Fixed threshold needs a specific configuration for each kind of video, and so does not allow neither a non constrained data acquisition nor a non domain-specific method.

3 Proposed method

The method proposed here is concerned, in real time, with dynamic colour image sequences. We have elaborated a method that should be able to process both compressed or non compressed videos. The first section will be devoted to the necessary pre-processing step that allows us to get a spatial reduced image, which could be obtained either from the DC coefficients in the case of compressed data, or from a block representation for non compressed data. The smaller size of the images induces the reduction of the computation time.

Then we are to precise the three main steps that our method comprises. First the problem of the choice of colour space will be solved in order to decrease the illumination effect in RGB space, then the distance measure used in our method to compare two successive frames will be presented. The last step of the process, cuts and other effect detection, is based on temporal evolution of the inter-frame differences. This difference is compared with some adaptative threshold.

3.1 Low resolution images

Let us recall we are concerned with real time methods. Methods working at the pixel level are known to be very sensible to noise. They also often result in heavy calculation. In order to avoid these limitations, spatially reduced images are built. These images are obtained in two ways, depending input data is compressed or non compressed.

Compressed videos like MPEG or MJPEG use DC coefficients. With these coefficients, it is possible to reconstruct a spatially reduced image, as presented in [5]. We can easily get a rebuilt image built with one value for each uncompressed 8x8 block.

For uncompressed videos n blocks 8x8 pixels large are defined, and for each block is calculated the average level of the pixels. We then create a smaller image based on the average values calculated from the blocks.

In both cases, compressed and non compressed input data, we finally get a spatially reduced image, which is in fact a set of three small images, one for each colour component R,G,B. After having reduced spatial resolution for noise robustness and data size reduction, then we decrease colour resolution.

3.2 Modification of the colour space and definition of the distance measure

Illumination is an embarrassing factor when a general method has to be elaborated because of its sensitivity to the scene lighting, which could be more or less intense. So we are searching for a colour representation space which is more robust to illumination changes than usual RGB space. We then decide to modify the colour space and we choose HSV (Hue, Saturation, Value) space. Keeping only the H and S values allows us to eliminate the effect of illumination represented by the V component. Furthermore, we have once more decreased the amount of data to be processed. More precisely, we calculate, for each pixel i in a frame numbered t, the state P as a linear combination of H and S as presented in equation 1 :

$$P_i(t) = \alpha \cdot H_i(t) + (1-\alpha) \cdot S_i(t) \tag{1}$$

where α has to be determined. As a by effect, reducing size of data to process to a scalar value per pixel is very useful in a real time framework.

Shot changes are detected on a frame-to-frame difference basis. In our case we have to compare two spatially reduced images. In order to estimate the dissimilarity between two frames, which is linked to the probability to detect a shot change, we define the following distance measured between any two frames :

$$d(t_1,t_2) = \sum_i \left(|P_i(t_1) - P_i(t_2)|^n\right) \text{ with } n = 0,1,2 \text{ and the convention } 0^0 = 0. \tag{2}$$

The possible choice of n allows us to give more or less importance to the difference $P_i(t_1) - P_i(t_2)$. If we choose $n = 0$, then a low difference between two consecutive pixels in the time domain will have as much importance as a high difference (for instance 1 gives same result as 100). In other words, two pixels which have very close values of P will be considered as different as two pixels which have values of P very far one from the other. On the opposite, a high value of n ($n = 2$) will increase the influence of the difference $P_i(t_1) - P_i(t_2)$ in the calculation of $d(t_1,t_2)$.

This approach is well-suited in case of real time constraint, where it is necessary to process a minimal amount of data. Indeed we get only one value per difference, which allows short processing times.

Now it is possible to study the evolution of the different frames all along the video and to discriminate shot changes.

3.3 Temporal evolution of inter-frames differences for shot change detection

As presented in previous section, it is possible to simplify an inter-frame difference to a single value. In order to detect cuts or more complex effects, we track this value through time ([3],[6]). We will first present cuts detection, and afterwards we apply the method to more complex effect detection.

Shot changes detection is usually done thanks to parameters adjustement, which is often specific to the video domain or to the kind of shots present in the sequence. For instance, some parameters will fit well with far shots (where moving objects are small) and others will fit better with close shots (where sometimes moving objects occupy an important proportion of the image). However some videos (like football videos) are containing close shots and far shots.

It is then necessary to use a general method which adapt to different kinds of shot. The method we propose here is able to process different video domains and different kinds of shots, thanks to the use of adaptative threshold.

In order to detect cuts, we first calculate the differences $D(t)$ as defined below :

$$D(t) = |d(t, t-1) - d(t-1, t-2)| \tag{3}$$

We then compare these differences with a threshold $T1$ defined according to the distance that has previously been chosen. It is linked to the mean value of the states all over the block. For instance, if we consider the case $n = 1$, the threshold $T1$ is calculated as :

$$T1 = K \cdot \mu \cdot 0.05 \tag{4}$$

with K the number of blocks and μ the mean of P values. If the value $D(t)$ is greater than the threshold $T1$, then we assume that a shot change is present. This could be easily explained : if $d(t, t-1)$ and $d(t-1, t-2)$ are significantly different, the difference between frames t and $t-1$ is not the same as the one between frames $t-1$ and $t-2$. So there is an abrupt change in the sequence at time $t-1$, opposed to a gradual change, which could be observed with a camera movement or a gradual transition effect.

Furthermore, if a cut has not been detected, it is still possible to be in presence of others effects, which could be considered as long-time shot changes. These effects are more complex to be detected and a method based on a fixed threshold will be limited. So we are using an adaptative threshold which presents two main advantages. The first one is of course the possibility for the method to adapt to different video domains (sport, news, etc.). The second advantage concerns the capacity of the method to adapt along a same video to different kinds of shots (close or far shots). It is very useful in the case of a sport game video where there could be far shots (moving objects are small in this case, and so the threshold should be low) or close shots (moving objects are very large because of a zoom and so the threshold should be high in order to avoid false detections).

In order to detect these gradual effects, we analyse the successive values of $d(t, t-1)$. Contrary to $D(t)$, the value of $d(t, t-1)$ depends directly on the sequence analysed and will not be the same if we are in a close shot or a far shot. As we have seen before, using an adaptative threshold will allow us to analyze correctly the value of $d(t, t-1)$ independently of the situation. We take inspiration from [6] to detect these gradual effects : we set a low threshold $T2$ and compare it with the current value $d(t, t-1)$ representing the difference between frames t and $t-1$. If $d(t, t-1) > T2$, there is a non zero probability that there is a shot change. We then begin a cumulative calculation of $D(t)$ until we obtain a difference value $d(t, t-1)$ lower than the threshold $T2$. Finally we compare the cumulative value obtained for D with the threshold $T1$ previously described : if the cumulative value of D is greater, then a shot change is detected. Contrary to the threshold $T1$, the threshold $T2$ is adaptative and is defined as :

$$T2 = 0.75 \cdot T2(t-1) + 0.25 \cdot d(t, t-1) \tag{5}$$

The threshold $T2$ is recalculated at each frame. This is the way it adapts automatically to the studied video : its values are modified depending on the evolution of the values of $d(t, t-1)$ in order to avoid false detection and misdetection.

4 Results

We have experimented this method on videos from different domains. We present here only results on football game videos. We think they allow to highlight the capacities of our method because shot changes are more complex to detect on this kind of videos than on others as TV news. Results on other kinds of videos were always better than those on football videos.

Images presented here have been acquired from football videos with a frame rate of 25 images per second. The width and height of images are respectively 160 and 120 pixels. After spatial reduction step, the images to be processed are 20x15 pixels. These sequences are containing several effects (cut, fade, wipe, and also some special effect combining wipe and fade), but they also include noise due to global camera movement, moving objects, very large moving objects (in the case of a close shot), and illumination effects.

Some examples of shot changes are presented in figures 1 to 3. They show the different kinds of transitions described previously : cut (figure 1), fade (figure 2), and wipe (figure 3).

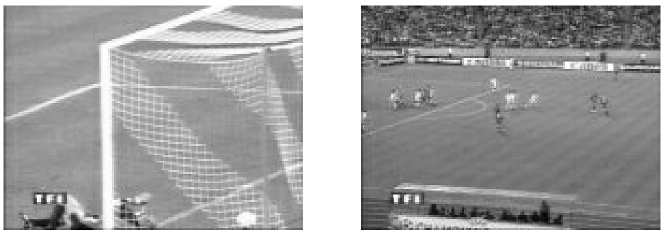

Figure 1: Successive frames representing a cut

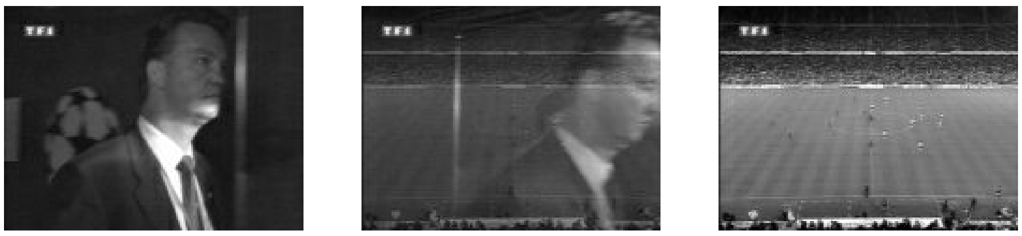

Figure 2: Successive frames representing a fade

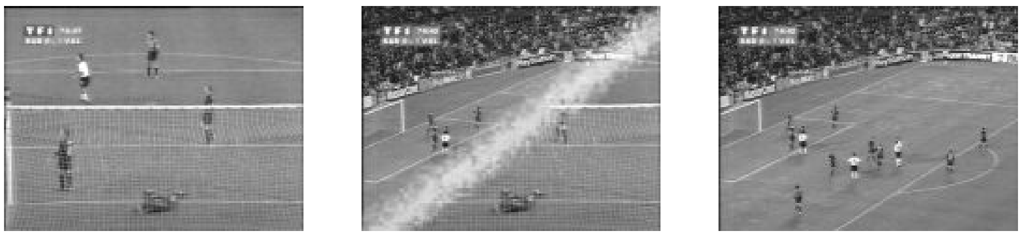

Figure 3: Successive frames representing a wipe

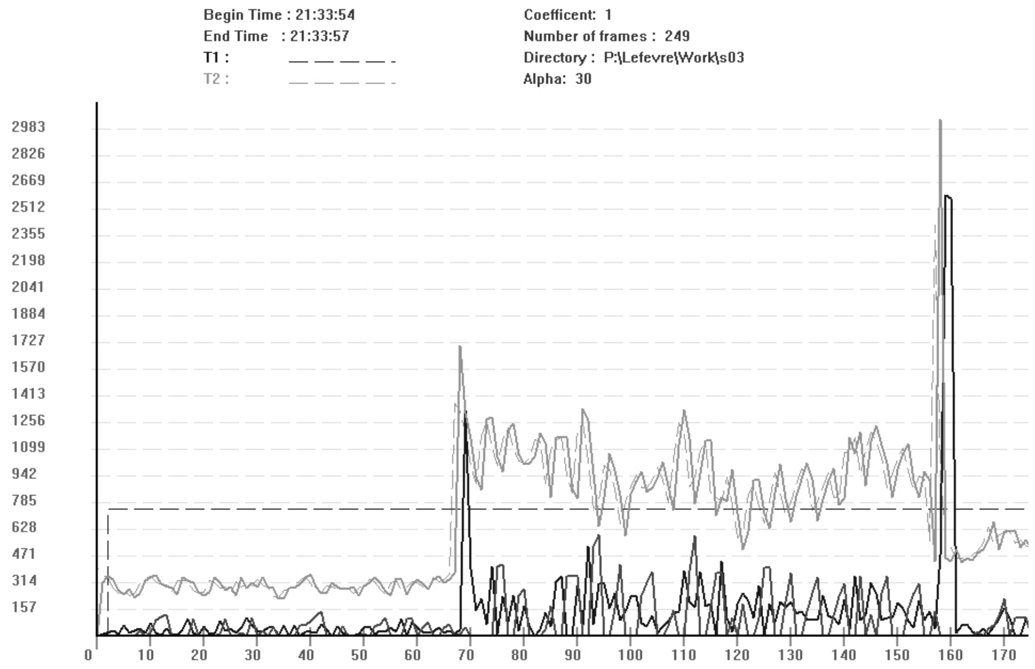

Figure 4: Evolution of $d(t)$ and $D(t)$ for a sequence containing two cuts at frames 69 and 160

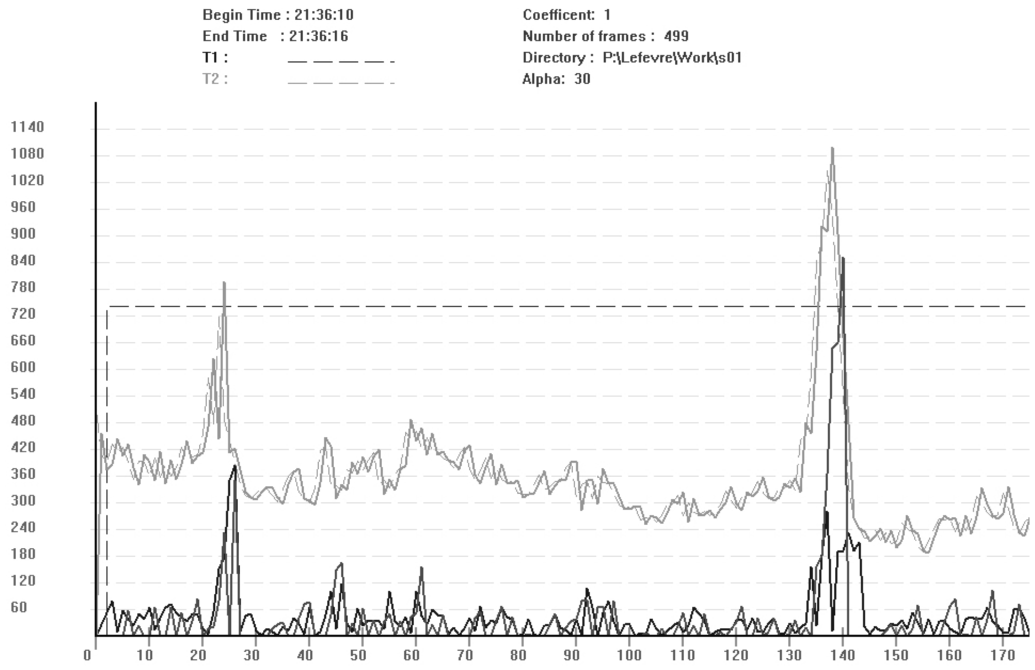

Figure 5: Evolution of $d(t)$ and $D(t)$ for a sequence containing a progressive effect (wipe) between frames 134 and 141

Quality of a shot change detection method could be evaluated thanks to several measures. We use the following ones :

$$Q_R = \frac{N_d}{N_d + N_m} \tag{6}$$

$$Q_P = \frac{N_d}{N_d + N_f} \tag{7}$$

$$Q_G = \frac{N_d - N_f}{N_d + N_m} \tag{8}$$

where Q_R, Q_P, Q_G are respectively the quality of robustness, the quality of precision, and the general quality, and where N_d, N_m, N_f are respectively the number of shot changes detected, the number of shot changes missed and the number of false detections.

Figures 4 and 5 present results obtained from two different image sequences. Parameters α and n were respectively set to 0.3 and 1. We test these parameters on about 20 sequences of 500 images each. With these parameters we obtain the values $Q_R = 70\%$, $Q_P = 97\%$, and $Q_G = 67\%$, which proves that our method is efficient.

5 Conclusions

The method proposed in this paper is able to detect, in real time, shot changes in a dynamic colour image sequence. It is also robust to noise, important movement, and illumination changes. Some results on football video sequences have been presented which show the efficiency of the method.

Further research will include testing of other difference measures, transformations in other colour spaces, and the implementation on a multiprocessor workstation. A direct use of the difference blocks is also tested as a way to track objects (at a coarse resolution).

References

[1] G. Ahanger and T.D.C. Little, "A Survey of Technologies for Parsing and Indexing Digital Video", *Journal of Visual Communication and Image Representation*, vol. 7, no. 1, Mar. 1996, pp. 28-43.

[2] R. Brunelli, O. Mich and C.M. Modena, "A Survey on the Automatic Indexing of Video Data", *Journal of Visual Communication and Image Representation*, vol. 10, no. 2, Jun. 1999, pp. 78-112.

[3] C.H. Demarty and S. Beucher, "Morphological tools for indexing video documents", in Proc. *IEEE International Conference on Multimedia Computing and Systems - ICMCS'99*, Jun. 1999, pp. 991-992.

[4] F. Idris and S. Panchanathan, "Review of Image and Video Indexing Techniques", *Journal of Visual Communication and Image Representation*, vol. 8, no. 2, Jun. 1997, pp. 146-166.

[5] J. Song and B.L. Yeo, "Spatially Reduced Image Extraction from MPEG-2 Video : Fast Algorithms and Applications", in Proc. *Storage and Retrieval for Image and Video Database VI*, Jan. 1998, pp. 93-107.

[6] H.J. Zhang, A. Kankanhalli, and S.W. Smolliar, "Automatic Partitioning of Full-Motion Video", *Multimedia Systems*, vol. 1, no. 1, 1993, pp. 10-28.