Copyright 2005 EURASIP. Published in Proceedings of 5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Service, June 29 – July 2 2005 Smolenice, Slovak Republic.

SCENE CHANGE DETECTION FOR H.264 USING DYNAMIC THRESHOLD TECHNIQUES

Anastasios Dimou, Olivia Nemethova and Markus Rupp

Institute for Communications and RF Engineering
Vienna University of Technology
Gusshausstrasse 25/389, 1040 Vienna, Austria
A.Dimou@tue.nl, {onemeth, mrupp}@nt.tuwien.ac.at

Abstract: *Several scene-change detection algorithms have been proposed in literature up to now. Most of them use fixed thresholds for the similarity metrics used to decide if there was a change or not. These thresholds are obtained by empirically or they must be calculated before the detection after the whole sequence is obtained. Performance of most scene change detectors decreases considerably for videos with high scene complexity and variation. In this paper, we review previous works in this area, and study the correlation between local statistical characteristics, scene duration and scene change. Based on this analysis, we further propose and implement a novel scene change detection method for H.264 codec, defining an automated, dynamic threshold model which can efficiently trace scene changes. Experimental results on QCIF videos indicate very good performance with significantly improved accuracy combined with minimum complexity.*

Index Terms: *Dynamic threshold model (DTM), scene change detection, automated threshold, scene duration, H.264*

1. INTRODUCTION

Scene change detection is of great importance for a number of applications today. Video indexing techniques, which are necessary for video databases, rely on it. It is also necessary for the extraction of high level semantic features. Moreover, it provides information for video preprocessing, compression codecs and error concealment techniques.

Many types of scene change detection schemes have been proposed in literature. Computational schemes define a similarity measure between two consecutive frames. When this measure reveals a big enough change, a scene change is declared. These schemes define a threshold, typically a fixed one. If the value of the measure exceeds the threshold, a scene change is detected. However, a fixed threshold value cannot perform well for all videos mainly due to the diversity of their characteristics. The key problem is to obtain an optimal value for such fixed threshold. If it is set too high, there is high probability that some cuts remain undetected. If it is too low, the detection scheme produces false detections. In real-world videos, we can have both cases simultaneously.

To solve the threshold selection problem, many approaches have been proposed. In order to overcome the detection problem a double threshold (high – low) was proposed to eliminate missed scene changes and dismiss false ones [1]. Although it improved the efficiency, results are not sufficient, especially in real-world videos with high motion like sport games. In addition to this method, a function-based lowering of the threshold, after a scene change was used to decay from high to lower threshold [2]. This technique was used to avoid false detections close to a real scene change, assuming that scene changes cannot occur immediatelly after each other. However, in most of these methods an optimal threshold (or two thresholds) had to be determined for each video in advance. Other methods were proposed to find automatically an optimal static threshold e.g. using histogram differences [3], entropy [4] or the Otsu method [5], but they still have the disadvantage of a static threshold and therefore they are not suitable for real-time applications. A truly dynamic threshold is presented in [6], where the input data is filtered by a median filter and then a threshold is set using the filtered output and standard deviation. However, it is not suitable for real-time applications, as the median filter uses future frames as well. A different approach for variable bit rate video is presented in [7], where the bit-rate used in each frame is the "change" metric. It uses statistical properties of the metric values in a single shot, together with the shot's length, to define a threshold.

In this paper we focus on a dynamic threshold model for real-time scene change detection in different video sequences. The method we use is based on the extraction of the sum of absolute differences between consecutive frames from the H.264 codec. These differences serve as a criterion for the choice of the compression method as well as for the temporal prediction. We use a sliding window to extract local statistical properties (mean value, standard deviation) which we further use to define a continuously updating automated threshold. This paper is organized as follows. In Section 2, we describe the compression standard H.264 and its features we use. In Section 3 we define the dynamic threshold model for video streams. Section 4 includes the simulation results, while in Section 5 conclusions and final remarks can be found.

2. H.264 ENCODER

H.264 [8] is the newest high compression digital video codec standard proposed by the ITU-T Video Coding Experts Group together with the ISO/IEC Moving Picture Experts Group as the product of a collective partnership effort known as the Joint Video Team. This standard is is also known as Advanced Video Coding (AVC). The H.264 encoder has three different types of frames defined: spatially predicted frames (I), from previous frames predicted frames (P), bi-directionally predicted frames (B). Each color input frame contains both chrominance and luminance data. Each frame is tiled in macroblocks which are separately encoded spatially or temporally. Macroblocks in H.264 are further tiled in smaller blocks. Each block can be compared with the respective block in the previous frame. As a similarity metric Sum of Absolute Differences (SAD) is used:

$$SAD(n) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left| F_n(i,j) - F_{n-1}(i,j) \right|, \tag{1}$$

where F_n is the n-th frame of size NxM, i and j denote the pixel coordinates. SAD values are further stored and used to evaluate the best compression scheme. The encoder can use a pre-defined number of previous frames as a reference to make the best decision. In our work we extract the SAD values between every block and the respective block in the first reference frame. For P frames the previous P or I frame and for the B frames any previous frame can be used as a reference. Then we add up all the differences in order to acquire the SAD between those frames. Thus, we have a measure to detect scene changes.

The biggest advantage in this procedure is the almost zero added complexity to our encoder so far. Another big advantage is its scalability. Due to the segmentation of the frame in 4x4 blocks, we can add up the SADs only in a part of the frame and detect with accuracy local scene changes (i.e. news videos with a screen or a window behind the newscasters). This can be achieved again with minimum complexity. Another application could be the use of part of the frame to increase computational efficiency for scene change detection in high resolution videos.

3. AUTOMATED DYNAMIC THRESHOLD MODEL

A scene is a block of frames between two scene changes. The sequence of SAD values that are computed, has statistical properties that are valuable in our effort to detect scene changes. On scene changes we obtain a high SAD value which makes them detectable. However, we can also have high SAD values during rapid movement, sudden light changes and transition effects such as zoom in/out, dissolve etc. Moreover, we can have scene changes with very different value levels. A scene break, where both scenes have similar background does not give a peak as high as if they had different ones. Consequently, a thresholding function is needed, which will be able to adapt to the character of the scene without the need for a previous input. If we want to preserve the real-time character of the algorithm, we should only use the previous sequence values, without looking in the future.

For the proposed thresholding function, we use local statistical properties of the sequence. Let X_i be a random variable which models the SAD value in frame i. We use a sliding window of length N. If the actual frame number is n, then the window lies between $\left[n-(N+1),n-1\right]$. We compute the empirical mean value m_n and the standard deviation σ_n of X_i in the defined window as follows:

$$m_n = \frac{1}{N}\sum_{i=n-N-1}^{n-1} X_i \,, \tag{2}$$

$$\sigma_n = \sqrt{\frac{1}{N-1}\sum_{i=n-N-1}^{n-1} (X_i - m_n)^2} \,. \tag{3}$$

We use (2), (3) together with X_{n-1} to define the threshold $T(n)$ as follows:

$$T(n) = a\cdot X_{n-1} + b\cdot m_n + c\cdot\sigma_n \,, \tag{4}$$

where n denotes the current frame and a,b,c are constants. Alternatively, we can read (4) as $T(n)=a(X_{n-1}-m_n)+(b+a)m_n+c\sigma_n$, which is more illustrative for the following discussion about appropriate choise of the constants. Constants a,b,c are very important because they determine the character of the thresholding function. If b takes high values, the threshold will become more rigid, keeping high values without approaching the X_i sequence. This avoids wrong change detections like in case of intense motion scenes; but on the other hand, the detector can miss some low valued, difficult scene changes. A low value of b allows us to detect regions with high activity, which can provide valuable information to other applications (e.g. bit-rate control, semantics, error concealment etc.). As σ_n is the standard deviation, high values of c prevent from detecting intense motion as a scene change. On the contrary, a must have a small value because, as we have already discussed, the properties of X_i are not always welcome in scene change detection. The whole procedure of choosing a, b, c is a tradeoff and should be performed with respect to the intended application.

In addition to the dynamic threshold model described, a function based lowering of the threshold found in [2] is employed to avoid false detections immediatelly after a scene change. When a scene change is detected in frame *p*, the SAD value of this frame is assigned to the threshold. In this case, for the next *K* frames we further use the threshold $T_e(n)$ which is decaying exponentially, in order to avoid false scene change detection very close to the previous change:

$$T_e(n)=X_{n-1} \exp(-s(n-p)), \tag{5}$$

where *s* controls the speed of decaying.

4. EXPERIMENTAL RESULTS

In this section, we validate the proposed scheme for real-time encoding application. For this purpose we use the H.264 codec v9.2 found in [9] modified to extract the SAD values. We used video sequences with football content. They were chosen because they have scenes with intense motion, change of light conditions, high complexity and different types of scene changes. There are changes between the field and the crowd, which are easy to detect but there are also scene changes with the same background (the playground) which are more complicated. In football sequences also many visual effects like zoom in/out and transition effects like dissolve, fade in/out can be found. All these characteristics make football videos very challenging for a scene change detector.

The first video sequence (*"zibsport"*) is a collage of highlights from the Austrian league. It includes the cup ceremony, celebrations, football highlights and even some violent incidents in the watching crowd. Its resolution is 320 x 240, and its length is 4,000 frames. It was encoded with the H.264 codec with one I frame at the start and P and B frames in the format PBPBPBPB. The number of true scene changes in this sequence was 36. We obtained them 'manually' by watching the video and counting them.

The second video sequence (*"eurosport"*) and it is a typical football match recorded in 2004. Its resolution is 172 x 144 (QCIF) and it is encoded in the same way as the previous video. Its length is 20,000 frames. The number of true scene changes in this sequence was 64. The evaluation of the proposed method is performed by comparing with other methods and the ground truth. For this reason we employ the "recall" and "precision" ratios:

$$\text{Recall} = \frac{N_C}{N_C + N_M}, \tag{6}$$

$$\text{Precision} = \frac{N_C}{N_C + N_F}, \tag{7}$$

where N_C is the number of correct detections, N_F the number of false ones and N_M the number of missed ones.

Table 1 and Figure 1 show the results of the scene change detection with a fixed threshold, with the dynamic threshold with and without exponential decaying.

zibsport (320x240)	TRUE	FALSE	MISSED	RECALL	PRECISION
Fixed Threshold	29	10	7	0.8	0.74
Dynamic Threshold without exponential decaying	35	2	1	0.97	0.94
Dynamic Threshold with exponential decaying	35	1	1	0.97	0.97
eurosport (172x144)	TRUE	FALSE	MISSED	RECALL	PRECISION
Fixed Threshold	43	12	17	0.67	0.78
Dynamic Threshold without exponential decaying	60	6	4	0.93	0.91
Dynamic Threshold with exponential decaying	60	6	4	0.93	0.91

Table 1. Results of the scene change detection

The fixed threshold used for comparison was chosen optimally (the best case) – after having the SAD values for the whole sequence, we took the value that minimizes the number of missed and false detections. Dynamic threshold parameters *a, b, c* we set to the following values: *a=-1, b=2, c=2*. These values were chosen the same for both sequences although they have different resolution to test their sensitivity. We used a sliding window of size 20 frames and speed of exponential decaying *s=0.02*. Please note, that scene change detection is more difficult in small resolutions like QCIF and thus the results are better for the "zibsport" sequence. We could do better if we set the DTM parameters separately for the first and second sequence. However, despite the use of an optimal fixed threshold the DTM performs significantly better in both cases.

Optimal setting of the DTM parameters may also be slightly different for sequences with another character (for instance a TV discussion, video clip, etc.). Exponential decaying improves the results slightly, but the added value is rather low compared to the contribution of DTM itself against the fixed threshold.

5. CONCLUSIONS

In this paper we presented a novel automated dynamic threshold model for scene change detection. It is based on the local statistical properties of the video sequence. The method was designed and implemented for H.264 codec, but the idea could be used for any other codec or even for raw video sequences as well. Proposed method has the advantage of low complexity. Moreover, it uses only previous frames for the detection, which makes is suitable for real time applications. The method performs significantly better than an optimum fixed threshold setting and gives very good results also for low resolutions. It can be further enhanced to recognize and handle different kinds of transitions.

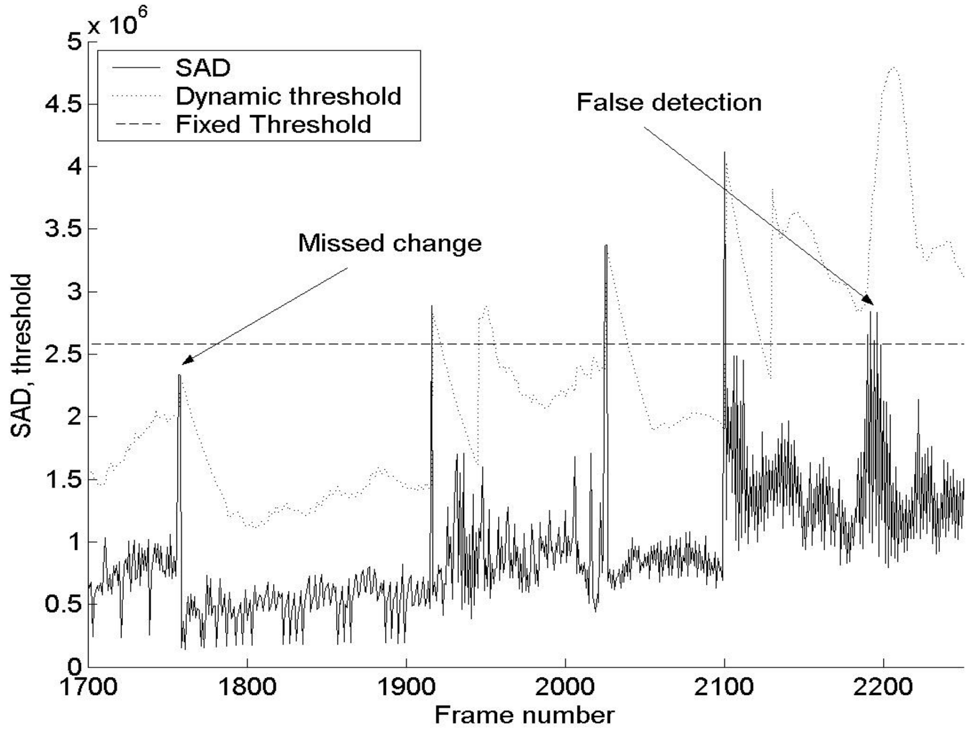

Figure 2. Results of the scene change detection: DTM allows correct detections even in the case where the best fixed threshold results in missed and false detections.

REFERENCES

[1] C.L.Huang, B.Y.Liao, *A Robust Scene-Change Detection Method for Video Segmentation*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 12, pp. 1281-1288, Dec. 2001.
[2] S.Youm, W. Kim, *Dynamic Threshold Method For Scene Change Detection*, ICME, vol. 2, pp. 337-340, 2003.
[3] X.Wang, and Z.Weng, *Scene Abrupt Change Detection*, Canadian Conference on Electrical and Computer Engineering, vol. 2, pp. 880-883, 2000.
[4] K.W.Sze, K.M.Lam, G.Qiu, *Scene Cut Detection Using The Colored Pattern Appearance Model*, ICIP, vol. 2, pp. 1017-1020, 2003.
[5] P. K.Sahoo and S.Soltani, A. K.C.Wong and Y.C.Chen, *A survey of thresholding techniques*, CVGIP, vol. 41, pp. 233-260, 1988.
[6] H.C.Liu, G.Zick, *Automatic Determination of Scene Changes in MPEG Compressed Video*, ISCAS, vol. 1, pp. 764-767, 1995.
[7] H.Li, G.Liu, Z.Zhang, Y. Li, *Adaptive Scene-Detection Algorithm for VBR Video Stream*, IEEE Transactions on Multimedia, vol. 6, no. 4, pp. 624-633, Aug. 2004.
[8] T.Wiegand, G.J.Sullivan, G.Bjontegaard, A.Luthra, *Overview of the H,264/AVC Video Coding Standard*, IEEE Trans. on Circuits and Systems for Video Technology, vol.13, no.7, pp. 560-576, July 2003.
[9] H.264/AVC Software Coordination, *JM Software*, v9.2, available in http://iphome.hhi.de/suehring/tml/.