ROBUST VIDEO SHOT CHANGE DETECTION

Rakesh Dugad, Krishna Ratakonda and Narendra Ahuja

Abstract - In this paper we present a novel improvement to existing schemes for abrupt shot change detection. Existing schemes declare a shot change whenever the frame to frame histogram difference (FFD) value is above a particular threshold. In such an approach, a high value for the threshold results in a small number of false alarms and a large number of missed detections while a low value for the threshold decreases the number of missed detections at the expense of increasing the false alarms. We attribute this situation to the fact that the FFD cannot be reliably used as the sole indicator for the presence of a shot change. In the proposed method a two step shot detection strategy is used which selectively uses a likelihood ratio (computed directly from the frames and not from the histograms) to confirm the presence of a shot change. Such a two-step checking increases the probability of detection without increasing the probability of false alarm. The improvement proposed is simple and computationally cheap. Tests with a wide variety of video sequences prove the efficacy of the proposed approach.

INTRODUCTION

In this paper we are concerned with detecting shot changes in a video sequence. A shot refers to a sequence of contiguous frames with *continuous* action. A shot change refers to an abrupt change of frame content (note that we are not considering soft shot changes like dissolves and fades). Detection of shot change is useful for many applications including video browsing and retrieval, video compression, statistical characterization of video in terms of different attributes of a shot and global clustering of video documents [1].

A commonly used scheme in literature [2] detects shot changes in video by using a locally computed threshold on the frame to frame histogram difference (FFD) values. The problem with this approach is that using a high threshold increases the number of misses and using a lower threshold increases the number of false alarms. Why does this problem arise? We attribute this to the fact that the FFD values used as the sole indicators of a shot change give unreliable results.

In this paper we propose to use a two step process to get around this problem. First we check if the FFD values are above a certain high threshold. If yes a shot change is declared. If not then we check if the FFD is above a lower threshold. If that is true then the second step of our scheme consists of comparing the two frames directly. For this the frames are divided into blocks and the blocks in one frame are compared with neighboring blocks in other frame using a likelihood ratio. The result of such a test is used to confirm the presence of a shot change. Results indicate that such a two-step scheme is very effective in increasing the number of detected shot changes without increasing the number of false alarms. We note that the relatively expensive second step is invoked only for a very minute fraction of the video frames and hence does not impose much computational burden.

Section briefly describes the approach in [2] and also shows some results using that approach. The results show the merits of the algorithm but also highlight some problems with that approach. Section describes our novel improvement to this approach. Results are also presented in that section to show the improved performance.

PREVIOUS APPROACH

The algorithm which is a precursor to our method was proposed in [2] and incorporates ideas from [3]. Let $H_k(i)$ denote the intensity histogram of the kth frame where i is one of the G possible intensity (grayscale) values. Then y_k defined as the l_1 norm of $H_{k+1} - H_k$ is used to measure the similarity between the kth and $k+1$th frame.

$$y_k = \sum_{i=1}^{G} \mid H_{k+1}(i) - H_k(i) \mid \tag{1}$$

y_k will be called as the observations or samples.

Consider a window (say of size 21) and consider testing for shot change at the middle sample of the window. Compute the sample means ($\hat{\mu}_{left}$ and $\hat{\mu}_{right}$) and standard deviations ($\hat{\sigma}_{left}$ and $\hat{\sigma}_{right}$) of the first and last 10 samples respectively in this window. The middle sample represents a shot change iff

1. It is maximum in the window.
2. It is greater than $\max(\hat{\mu}_{left} + h\hat{\sigma}_{left}, \hat{\mu}_{right} + h\hat{\sigma}_{right})$ where h is taken as 5. This condition ensures that only the frames in the immediate *local* neighborhood of a given frame decide whether or not there is a shot change at that frame.

Results with Previous Approach

For the computation of the histogram we use only the 6 Most Significant Bits (MSBs) of intensity value – hence there are 64 bins in the histogram. Figure 3(a) shows the y_k and the thresholds (with h=5) computed at local maxima in y_k values. The video is that of a car racing competition. The camera pans from the cars to the competitors (drivers) to the commentators. When a competitor is shown the camera is focused on his face and there is lot of motion of the face during such shots. Most of the spikes seen in the figure (e.g. those between frames 290 and 350) but not detected as shot changes are due to this motion. Examining the actual video shows that there are no false alarms. Also most of the shot changes have been detected. However there is a miss at frame number of 839 (the shot between frames 839 and 840 is missed) as can be clearly seen in the figure also. Figure 3(b) shows a closeup of this portion of the graph. We see that the miss happens because the frames *immediately after* 839 have really large histogram differences. The corresponding frames are shown in the top row of figure 2.

Figure 3(c) shows similar graph for another video clip. Again there are no false alarms but a couple of misses. Figure 3(d) shows a closeup of this graph. There is actually a significant shot change from frame 183 to 184 but it is missed. This happens because *immediately before* this shot change there is a peak at frame number 181 due to large motion and change in intensity of the frame (due to sun). Similarly the shot at 201 is missed due to the pick at 200 (here due to fast zoom of the camera). The corresponding frames are shown in the bottom row of figure 2.

PROPOSED IMPROVEMENT

Figure 1 shows a block diagram of our method. Results in the previous section suggest that a few shot changes are missed by the algorithm. One way to avoid this is to lower the threshold but that would increase the number of false alarms. Hence we need a way to get around the false alarms. The following metric called the *likelihood ratio* has been proposed [4, 5, 1] for comparing two frame regions :

$$L = \frac{\left[\frac{S_k+S_{k-1}}{2} + \left(\frac{m_k-m_{k-1}}{2}\right)^2\right]^2}{S_k * S_{k-1}} \tag{2}$$

where m_k and S_k denote the mean and variance (of intensity) of a given region in frame k. It is easy to see that this ratio attains its minimum value of 1 if and only if $S_{k-1} = S_k$ and $m_{k-1} = m_k$. Using this likelihood ratio we propose a novel way to increase the probability of detection without a corresponding increase in number of false alarms. The proposed approach is also computationally efficient. We use two thresholds T_L and T_H say $T_L = \mu + 3\sigma$ and $T_H = \mu + 5\sigma$ computed as before. If $y > T_H$ then a shot change is declared right away. Else if $y > T_L$ then a comparison is made between the previous and current frame using the likelihood ratio L (in a manner described below) to investigate if this is a false alarm or a actual shot change. If $y < T_L$ no shot change is declared.

Computing likelihood ratio between two frames

We compare frames $k-1$ and k using L as follows. The frames used for our experiments are of size 240(height) $\times$ 352(width). So each frame is divided into an 8×8 array of blocks each of size 30×44 as shown in figure 1. A two block thick border is left out in frame k and the 16 middle blocks (enclosed in thick lines) are considered one by one. Consider the block (call it B_x) marked with an $\times$ in frame k. Its 9 neighbors in frame $k-1$ have been numbered from 1 to 9 (call them B_1 etc.). Compute the likelihood ratio between B_x and each of $B_1, \dots, B_9$ using equation (2). Let L_x denote the *minimum* of these ratios. Similarly we can compute L_x for each of the remaining 15 blocks enclosed within thick lines in frame k. Let these be denoted as $L_1, \dots, L_{16}$. Note that each block has its own set of 9 neighbors in frame $k-1$. Let $\mathcal{L} = \sum L_i/16$ i.e. the average of $L_1, \dots, L_{16}$. Then if $\mathcal{L} > t$ where t is some suitably chosen threshold we declare a shot change between frames $k-1$ and k else it is a false alarm. $t = 10$ turns out to a good choice in practice. Also as we shall see $\mathcal{L}$ has a large dynamic range and hence the result is not very sensitive to the choice of t.

We have left out the border blocks because they are likely to get out of or get in the frame whenever there is large object or camera motion, pan or zoom and hence are not reliable for comparison. Note that we compute the likelihood ratio with respect to all the 9 neighbors $(B_1, \dots, B_9)$ of a given block and then pick the one that is minimum of them. In the event of a shot change the minimum value itself is going to be large. On the other hand if there is no shot change we expect one of the pairs to be significantly smaller depending on the direction of any object or camera motion.

Computational Complexity

To compute the $\mathcal{L}$ between frames $k-1$ and k we proceed as follows: compute and *store* the means and variances for the 25 blocks enclosed within thick lines in frame $k-1$. Then consider one of the blocks enclosed in thick lines in frame k and compute its mean and variance. Compute the L values for comparison with the 9 neighboring blocks in previous frame using equation (2). The minimum value gives L_1. Similarly compute $L_2, \ldots, L_{16}$. To compute the means and variances of a given block we first compute its 256-bin [1] histogram and then compute the means and variances from the histogram which is computationally faster. Moreover we need to compute the $\mathcal{L}$ values only when the FFD values are between the two thresholds (T_L, T_H). Such frames represent a very small fraction of the total number of frames in the video. Hence we see that the procedure is computationally inexpensive.

RESULTS WITH IMPROVED SCHEME

Figure 3(b) shows that the shot change between frames 839 and 840 has been captured by the lower threshold. The $\mathcal{L}$ value between these frames is 34.507 which is much higher than the threshold $t = 10$. Similarly figure 3(d) shows how the missed shots are captured by the lower threshold value. The $\mathcal{L}$ value between frames 183-184 and between frames 201-202 are 467.882 and 244.045 respectively. Both of them are way above the threshold of $t = 10$. The $\mathcal{L}$ values for the cases where the y value is between the two thresholds but there is actually no shot change is in the range 1.0 to 4.0 - much below $t = 10$. Hence we see that the $\mathcal{L}$ value has large dynamic range which makes it easier to choose the threshold t.

References

[1] I. K. Sethi and N. Patel, "A statistical approach to scene change detection," in *Proceedings of SPIE*, vol. 2420, pp. 329–337, 1995.

[2] A. Hanjalic, M. Ceccarelli, R. L. Lagendijk, and J. Biemond, "Automation of systems enabling search on stored video data," in *Proceedings of SPIE*, vol. 3022 of *Storage and Retrieval for Image and Video Databases V*, (San Jose, California), Feb. 1997.

[3] B. Yeo and B. Liu, "Rapid scene change analysis on compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 533–544, 1995.

[4] R. Kasturi and R. Jain, *Computer Vision: Principles*, ch. Dynamic Vision, pp. 469–480. Washington: IEEE Computer Society Press, 1991.

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

[1]Note that the computation of frame to frame histogram difference y uses only 64-bin histograms but 256-bin histograms are used for the computation of the means and variances in the likelihood ratio

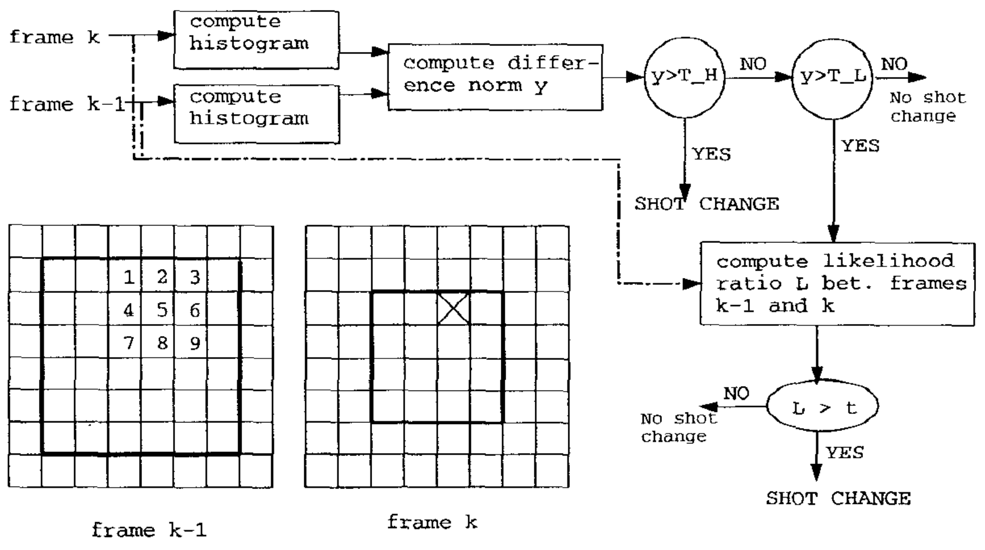

Figure 1: Block diagram of the proposed scheme for shot change detection. T_H is taken as $\mu + 5\sigma$ and $T_L = \mu + 3\sigma$. $t = 10$. For comparing the frames using the likelihood ratio each frame is divided into an 8×8 array of blocks as shown. For our frames each block is of size 30 x 44. Consider one of the blocks (say the one marked with a cross) enclosed within the thick lines in frame k. Compute its likelihood ratio with each of its neighbors 1 to 9 in frame k-1 using equation (2). Let L_1 denote the minimum of these 9 values. Similarly compute $L_2, \ldots, L_{16}$ using the remaining 15 blocks enclosed in thick lines in frame k. Then $\mathcal{L} = \sum L_i/16$ is defined as the likelihood ratio between the two frames.

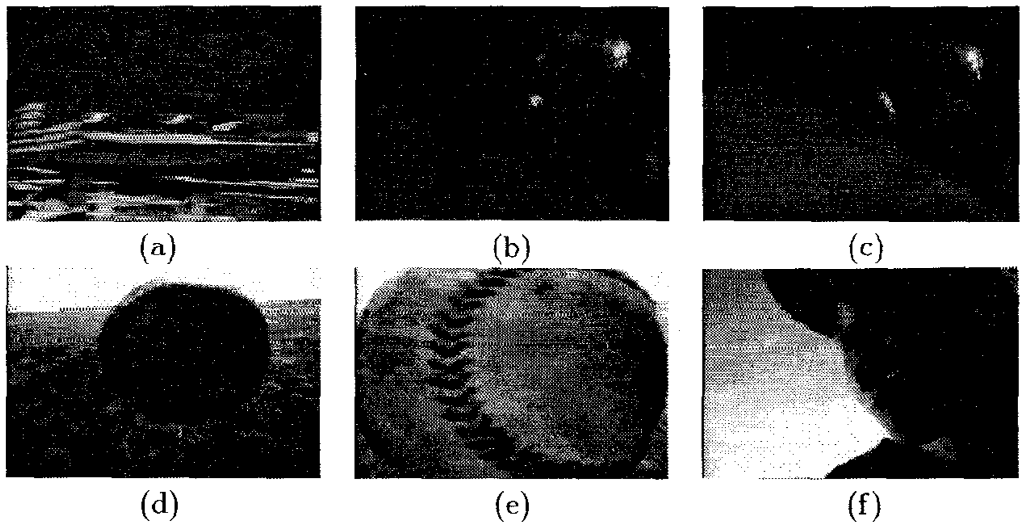

Figure 2: (a)-(c) shows frames 839,840 and 841 of a video clip. There is a shot change from frame 839 to 840 which is missed due to large motion from frame 840 to 841. Frames 840 and 841 show the face of the competitor. (d)-(f) shows frames 200,201 and 202 of another clip. There is a shot change from frame 201 to 202 which is missed due to fast zoom from frame 200 to 201.

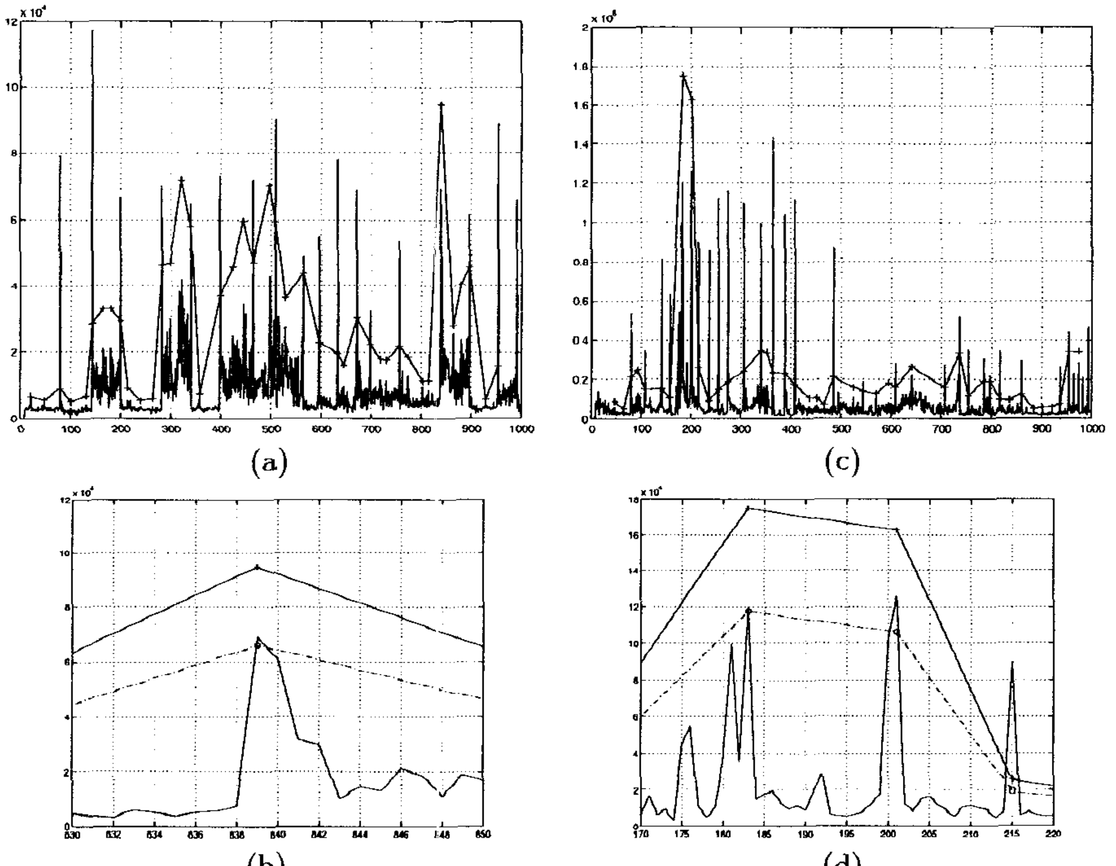

Figure 3: (a)&(c) The y value at x shows the norm of histogram difference between frames x and x+1. The + show the thresholds ($\mu+5\sigma$) evaluated at local maxima in y values. (b) & (d) Closeup of (a) & (c). Here the o's show lower thresholds($\mu+3\sigma$). + still shows the higher threshold. See how the lower threshold captures the shots missed by the upper threshold. But we need to take care of false alarms.