14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

SHOT DETECTION METHOD FOR LOW BIT-RATE H.264 VIDEO CODING

*J. Sastre, P. Usach, A. Moya, V. Naranjo and J.M. López**

Communications Department, Polytechnic University of Valencia
Camino de Vera s/n, 46022, Valencia, Spain
phone: +34963879719, fax: +34963877309, email: jorsasma@dcom.upv.es, web: www.gpiv.upv.es
*Mobile Network Multimedia Technology Division
Telefónica I+D, Telefónica, Madrid, Spain

ABSTRACT

This paper presents a low-complexity shot detection method for real-time, low bitrate video coding. Aimed at compression efficiency instead of frame indexing or other purposes, it is based on the macroblock intra/inter decision and the use of two thresholds. The first threshold is fixed and the second one is adaptive, providing robust scene change detection on almost all conditions (camera motion, zoom, high motion scenes or low number of frames per second). This method introduces the minimum number of necessary (but expensive in terms of bitrate) refreshing points in the video stream. The algorithm has been implemented and tested in an H.264 coder used to encode QCIF format sequences at a low, constant bitrate, for real-time, low delay communications.

1. INTRODUCTION

In low bitrate video coding the proper selection of key-frames in the video sequence is specially important in order to achieve the best possible quality in the coded stream.

Other applications, such as frame indexing, fast retrieval or scene analysis also need the proper detection of shot changes [1, 2, 3]. To support these purposes, reliable scene change detection algorithms have been developed [4, 5, 6, 7].

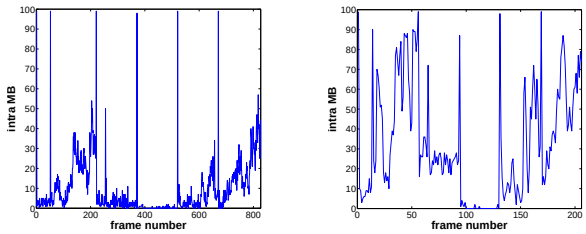
The main objective of the algorithm presented here is to detect the first picture of each shot and to encode it as an I-frame, as shot changes represent the best choice to insert key-frames in the video sequence. The aim is to encode the next frames of the new shot based on the first one via motion compensation and prediction (inter-frames or P-frames).

Our algorithm describes a real-time method to automatically detect shot changes during the video coding process, inserting the key-frames on the most productive positions to obtain the best quality in the coded stream.

2. SHOT DETECTION ALGORITHM

When a cut or scene change appears in a video sequence and the coder tries to encode it as a P-frame, coding efficiency and quality decrease. The main reason is the low correlation between the new frame and the last picture of the previous shot.

The proposed algorithm analyzes the compression process of each frame to extract information that allows it to detect a shot change before the picture is completely inter-coded. The information used to detect a shot change consists of the number of intra macroblocks (I-MB) used when encoding the frame as a P-frame. If this number of I-MB exceeds certain thresholds under certain conditions, then the algorithm detects a shot change, stops the encoding of the P-frame and encodes it as an I-frame. Previous algorithms used a fixed, absolute threshold [8], but some researches proved the advantages of a two thresholds method, one fixed and the other adaptive. This second option was successfully applied on H.263/H.263+ coders [9], with motion estimation algorithm of [10]. In this paper we apply the shot detection method on H.264 encoders [11] and propose some modifications to improve its performance.



a) 12.5 fps b) 3.125 fps

Figure 1: Number of intra MB versus number of frame for a sequence encoded at 12.5 fps and 3.125 fps.

To illustrate the basics of the algorithm, Figure 1 shows the number of intra MB used to encode each frame of a generated sequence that consists of successive shots of the well known test sequences Flower Garden, Fun Fair, Coastguard, Mobile & Calendar, Stephan and Snow. As it can be seen, in each shot change the number of I-MB used to encode the first frame of the new shot is practically the total number of MB of the frame. In such a case the frame should be coded as an I-frame better than a P-frame.

The number of I-MB grows when coding a high motion sequence (as in Fun Fair or Snow) and keeps a low value in medium and low motion sequences (Mobile or Coastguard).

The number of I-MB also increases when the number of frames per second decreases, due to the resultant relative increasing motion, as seen in the right side of Figure 1. Another effect which causes an increase in the number of I-MB is the camera motion as in panings, zooms, etc. When such kind of situations happen (for example in Fun Fair), a single threshold cannot offer good results on the detection of scene changes [9], because a single threshold that provides zero false and zero missed cuts does not exist .

The proposed algorithm consists of two basic thresholds, one fixed and the other adaptive, and some auxiliary parameters. These thresholds are expressed as a percentage of the total number of MB of a coded picture (99 MB in the case of QCIF format).

A fixed threshold is used as a security measure, setting a high value to assure that any frame whose number of I-MB exceeds the threshold is coded as an I-frame, independently of the rest of the scheme's conditions.

To sharpen the shot detection, an adaptive threshold is added. It depends on the average of I-MB of all the pictures encoded since the last I-frame and forces a frame to be coded as an I-frame if its number of I-MB exceeds the average of intra MBs of the previous frames in a certain quantity.

This work has been supported by a *Ministerio de Educación y Ciencia - FEDER* program under grant TEC2005-07751-C02-01, by the Polytechnic University of Valencia (UPV) interdisciplinary project 5607-2004, and by the UPV *"Programa de incentivo a la investigación"*.

To determine the average of I-MB of the previous frames within the shot, a smoothing algorithm with memory is used. This memory avoids tracking the number of intra MB too fast and provides a stable value for the desired average.

Moreover, a limit for the adaptive threshold has been added. This limit is placed below the fixed security threshold to prevent a frame from being encoded as a P-frame when almost all of its MB are intra-coded.

Finally, to avoid shot detections too close in time a span parameter is used. The span establishes a period of time after a shot detection when only the fixed security threshold is active. During this period, the adaptive threshold cannot cause the insertion of a key-frame.

The parameters that define the behavior of the algorithm are:

- T_f: Fixed security threshold
- T_a: Adaptive threshold
- $0 < \alpha < 1$: Memory parameter
- T_{lim}: Limit for the adaptive threshold
- S: span parameter in milliseconds

With all these parameters, the shot detection algorithm works as follows when a new frame is ready to be encoded:

- Use the number of I-MB of the last coded picture to update a weighted average m_k of the number of I-MB of all the frames encoded since the last I-frame: $m_k = m_{k-1}(1-\alpha) + MB_k\alpha$, $m_0 = 0$. k is the number of coded pictures since the last intra one and MB_k is the number of I-MB in the last encoded frame.
- Compare the frame number k with the span parameter S to determine which thresholds are active: if $k < n$, with $n = \frac{fps \times S}{1000}$, only T_f will be active during the encoding of the next frame. Else, if $k \geq n$ both T_f and T_a will be active. *fps* is the number of frames per second of the coded sequence.
- After the intra/inter decision of each MB of the new picture $(k+1)$ being encoded, compare the number of I-MB of the current frame (MB_{k+1}) with the active thresholds: if only T_f is active the frame will be detected as a key-frame if $MB_{k+1} \geq T_f$; else, the frame will be detected as an I-frame if $MB_{k+1} \geq min((m_k + T_a), T_{lim})$, with $T_{lim} < T_f$.
- In case of detecting the picture as an I-frame, the encoding of the picture as a P-frame is aborted, the algorithm parameters are reset $(k = 0)$ and the frame is re-coded as an I-frame.

Let's note that exceeding the fixed threshold (T_f) forces a frame to be encoded as an I-frame, independently of the coding type (intra/inter) of the previous ones. This only occurs during the span time after a previous shot detection, because after this span the adaptive threshold T_a will always match the re-coding condition before the fixed threshold T_f is reached. This fixed threshold is used to detect very fast and clear shot changes very timewise.

On the other hand, the value of $m_k + T_a$, limited by T_{lim}, is only active after the span time S. It is an adaptive threshold that follows the increase of I-MB in the case of high motion, zoom, high camera motion, etc. and produces a detection in case of an abrupt increase over the average number of I-MB after the span time.

The limit T_{lim} is fixed to assure that a frame with almost all its MB coded as I-MB is coded as an I-frame rather than as a P-frame. The objective of this limitation is to reduce the number of missed cuts not detected by the original, unlimited algorithm.

With these parameters, the main number of shot detections will be made by means of the adaptive threshold T_a while the security threshold T_f only works in very close and clear shot changes.

3. PARAMETER SELECTION

Once the algorithm has been introduced, in this section we are going to describe the selection of the values for the parameters which define the performance of the present scheme.

We must emphasize that this method is focused on the detection of shot changes in order to improve the quality of the compressed video using the H.264 codec under the condition of low bitrate, low frame rate and real-time, low delay communications. No bidirectional frames are used, and the coding scheme is IPPP...

The condition of constant, low bitrate forces the use of low frame rates to maintain an acceptable quality in the coded sequence. Each frame of the sequence must be encoded with approximately the same number of bits, so a frame using too many bits forces the next ones to use less bits and therefore to be coded with less quality to maintain a low bitrate. This situation could happen when a key-frame is inserted (an I-frame usually uses more bits than a P-frame) or when a high motion scene is being coded and a high amount of information has to be encoded.

The tests to obtain the values for the parameters of the algorithm have been carried out using an H.264 coder with fixed values of the bitrate of 20, 50 and 100 kbps. On the other hand, the values for the frame rate have been 12.5 and 6.25 frames per second (*fps*).

To study the different behavior of the parameters with diverse configurations of the algorithm, four different groups of test sequences have been used:

- Low Motion and Steady Camera (*LM&SC*): sequences filmed with a fixed camera with relatively low motion on the scene (dialogues, etc.).
- Moderate Motion and Moving Camera (*MM&MC*): sequences with moderate motion on the scene and some pannings and other smooth camera movements.
- High Motion and High Camera Motion (*HM&HC*): sequences with high motion scenes filmed with a camera which is in continuous movement (action scenes, etc.).
- All conditions (*ALL*): sequences that merge all kind of situations, from low to high motion on scene, different kinds of transitions and camera movements and fast shot changes that deteriorates the behavior of the algorithm (commercials, etc.).

To obtain enough shot changes, the test sequences have been extracted from diverse movies, commercials and from the test sequences widely used in the field of video compression. Nearly one thousand cuts have been analyzed, combining clear shot changes, transitions such as fades, dissolves, wipes, etc. and other situations.

The behavior of the algorithm has been measured in terms of the number of false alarms, FA's, (i.e. the use of an I-frame in a position where a real cut does not exist) and the number of missed detections, MD's, where the first frame of a new shot is encoded as a P-frame instead of as an I-frame.

The statistical parameters used to obtain the best parameters for the algorithm have been MD and FA expressed as *Recall* and *Precision* with respect to the total number of cut detections (D) [9]:

$$Recall = \frac{D}{D + MD's}, \quad Precision = \frac{D}{D + FA's} \tag{1}$$

Recall and *Precision* show the percentage of missed detections and false alarms and their analysis helps to obtain the best value for the parameters that fix the behavior of the shot detection algorithm: *Recall* reduces when the number of missed detection grows and *Precision* reduces when the number of false detections increases.

In general, when the thresholds T_f and T_a decrease, the algorithm gets more sensitive to the detection of new shots, but at the same time, the number of false detections gets also higher, obtaining worse values for the *Precision* parameter. On the other hand, when the thresholds increase there are less detected cuts, and the number of missed detections increases, decreasing the *Recall* parameter values.

With these considerations and the restrictions of low bitrate and low frame rate, the desired behavior of *Precision* and *Recall* is obtained when they get similar and high values, i.e. the number of missed and false detections are similar and sufficiently low. Anyway, to maintain the bitrate under control, missed detections are preferred to false detections, so *Recall* should be slightly lower than *Precision*.

The criterion to determine which cuts are well detected and which ones are false or missed detections must take into account some considerations such as the correlation between consecutive

coded frames in situations of low frame rate and high scene motion. In such situations, the detection of a false shot change must not be considered an error of the algorithm but a right detection, because the insertion of an I-frame helps to maintain the quality of the coded sequence.

A situation where a theoretically false cut should be considered as a proper detection could be a panning scene coded at a very low frame rate. In such a situation consecutive frames are very low correlated and therefore the use of I-frames could be justified.

frame 35 frame 36 frame 37

Figure 2: Three consecutive frames of "Foreman" sequence encoded at 3.125 fps.

In Figure 2 three consecutive frames of the test sequence Foreman encoded at 3.125 fps are shown. The panning movement of the camera and the low frame rate forces the insertion of a key-frame in frame number 36. The differences between consecutive frames cannot be compensated by the motion compensation algorithm due to the high motion of the camera during the interval between the capture of the two pictures, increasing the number of I-MB.

On the other hand, some cuts may be lost by the algorithm if the last frame of the old shot and the first one of the new shot are quite similar. This could happen, for example, in a dialogue sequence with a fixed and an homogeneous background or in a dark scene. In these cases the encoder may be able to encode the new frame as a P-frame with a number of I-MB below the threshold, achieving an acceptable quality near to an I-frame.

To fix the best values for the thresholds T_f and T_a, several combinations of values for both parameters have been tested with all video sequences (*LM&SC*, *MM&MC*, *HM&HC* and *ALL*).

Figure 3 shows the behavior of *Precision* and *Recall* in a typical situation depending on the values of the thresholds (T_a between 35% and 50%, T_f between 85% and 100%). As it can be seen, *Recall* (upper graph) decreases when the thresholds increase, producing a higher number of missed detections. On the other hand, *Precision* (lower graph) has the inverse behavior, increasing its value when the thresholds increase, producing less false detections.

The intersection of both graphs gives the values for the thresholds which provide the desired behavior, obtaining similar, high values for *Precision* and *Recall* (over 92% in Figure 3).

The observed behavior of *Precision* and *Recall* depending on

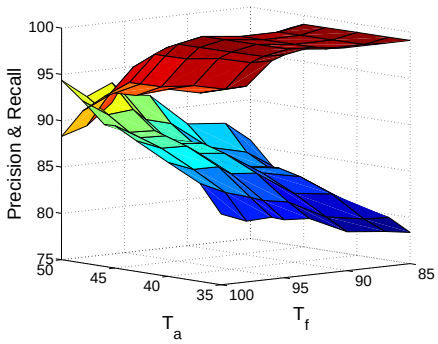


Figure 3: Precision and Recall depending on the values of the thresholds T_a and T_f.

		LM&SC		*MM&MC*		*HM&HC*		*ALL*	
fps	kbps	T_a	T_f	T_a	T_f	T_a	T_f	T_f	T_a
12.5	20	47	100	46	98	47	98	46	96
	50	49	99	48	98	48	98	49	98
	100	49	100	50	98	50	98	49	98
6.25	20	46	98	43	93	45	98	46	97
	50	49	100	47	97	47	98	47	98
	100	49	100	48	98	48	98	47	97

Table 1: Best intersection value of T_f and T_a for each combination of frame rate (*fps*) and bitrate (*kbps*) and for each group of video sequences. The T_f and T_a values are expressed as a percentage of the total number of MB of an encoded frame.

	bitrate		
framerate	20 kbps	50 kbps	100 kbps
12.5 fps	48%	49%	50%
6.25 fps	45%	47%	48%

Table 2: Final selection of the values for the adaptive threshold T_a.

the thresholds is roughly the same for the different values of the bitrate and frame rate. However, the optimal values for the thresholds depend on the bitrate and the frame rate, as seen in Table 1. This table shows for each group of video sequences and for each combination of bitrate and frame rate the values of the thresholds T_f and T_a that better fulfill the condition of providing similar and high values of *Precision* and *Recall*.

Figure 4 shows the average PSNR gain obtained encoding a video sequence with shot detection with regard to the encoding of the same sequence without detection. The axis show the value of the thresholds T_a (from 35% to 50%) and T_f (from 85% to 100%).

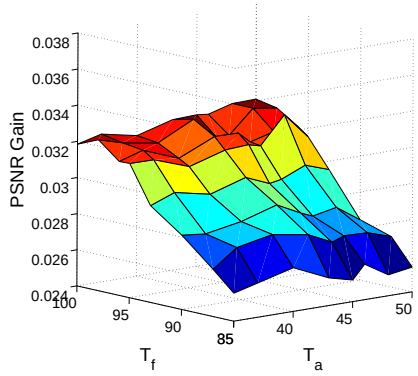


Figure 4: Average PSNR gain encoding the same video sequence with and without shot detection for the different values of the thresholds T_a and T_f.

An analysis of the PSNR obtained encoding the sequences with the different configurations of T_f and T_a shows the clear tendency of Figure 4: PSNR grows slightly when the security threshold T_f increases and meanwhile, the adaptive threshold shows a more constant behavior in terms of PSNR gain. These results are in agreement with the selection shown in Table 1, especially with regard to the security threshold T_f that requires values around 98%, which has been the final selection: $T_f = 98\%$ for all the combinations of bitrate and frame rate.

Finally, the selected values for the threshold T_a are shown in Table 2 as a function of the bitrate and the frame rate. The optimal T_a value increases with the bitrate and with the frame rate to obtain approximately the same effect with all configurations.

		LM&SC		*MM&MC*	
fps	kbps	*P*	*R*	*P*	*R*
12.5	20	98.12	99.09	100	93.90
	50	98.35	97.50	98.70	91.56
	100	97.20	99.77	95.40	96.51
6.25	20	93.13	96.88	94.31	96.51
	50	93.56	98.39	96.47	97.61
	100	93.27	99.08	90.81	100
		HM&HC		*ALL*	
fps	kbps	*P*	*R*	*P*	*R*
12.5	20	95.71	91.78	90.44	92.68
	50	97.10	93.05	88.83	94.15
	100	91.56	98.70	88.45	95.51
6.25	20	88.23	94.93	88.85	93.20
	50	88.35	97.53	86.20	94.19
	100	86.81	97.53	85.56	93.75

Table 3: *Precision* (P) and *Recall* (R) for the different groups of video sequences, bitrates and frame rates.

The selection of the memory parameter (α) has been obtained from a simulation of the algorithm before its final implementation on the H.264 coder. The simulation proved that a very high or a very low value of α forces the average m_k not to correctly follow the variations on the number of intra MB of the encoded frames, producing an increase in the number of false detections. Finally, the selected value for the memory parameter has been $\alpha = 0.25$.

Span determines the minimum gap between two consecutive I-frames, the second one detected by the adaptive threshold T_a. Increasing the *Span* time produces less cut detections (decreasing *Recall*) but also produces less false detections (increases *Precision*).

A large *Span* avoids the detection of consecutive shot changes in situations such as fades or dissolves. A span time of 500 miliseconds has been selected, which means that after a shot detection, a window of 6 frames at 12.5 fps or a window of 3 frames at a frame rate of 6.25 fps avoids the activation of the adaptive threshold T_a, producing a shot detection only if the number of I-MB of the current frame exceeds the security threshold T_f. If the coded video sequence contains very fast shot changes (as in commercials or movie trailers), a lower value of *Span* (for example 350 ms) should be used in order to decrease the number of missed shot changes.

Finally, the limit for the T_a threshold, T_{lim}, is fixed to prevent a frame from being encoded as a P-frame when almost all its MB are coded as I-MB. The selected value is slightly lower than the fixed security threshold T_f, taking a final value of $T_{lim} = 95\%$, which in the case of a QCIF format means a limit of 94 I-MB in an inter-coded frame. A higher value of this parameter produces a minimum increase on *Precision* parameter, and decreases *Recall*. The contrary occurs when adopting lower values for the threshold limit.

4. RESULTS

4.1 Precision and Recall

Table 3 shows the *Precision* (P) and *Recall* (R) results obtained with different groups of video sequences, bitrates and frame rates.

The first two video groups correspond with low and moderate motion sequences and the achieved results are very satisfactory, obtaining average *Precision* and *Recall* values of around 97%.

On the other hand, the last two groups consist of high motion scenes and adverse detection conditions, such as fades or fast shot changes. Here the values of *Precision* and *Recall* are slighty lower but still satisfactory, maintaining an average over 90% and achieving good *Precision* values.

4.2 PSNR

Different sequence conditions need different amounts of information to be encoded, so this information is coded with different quality depending on the available bitrate. The introduction of an I-frame in the sequence is expensive in terms of bitrate, but it also provides a refreshing point in the stream that solves possible transmission errors.

Figure 5 shows the PSNR of each encoded frame within the same sequence using the shot detection algorithm (solid line) and without shot detection (dashed line). Three shot detections appear in the figure, in frames number 806, 828 and 877.

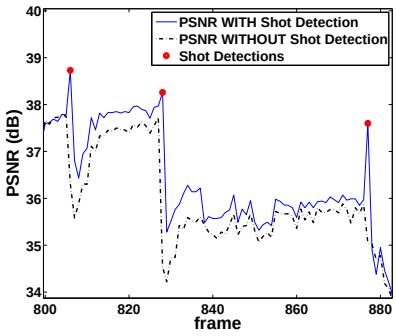


Figure 5: PSNR for 80 frames of the coded sequence with shot detection (solid) and without detection (dashed) versus the frame number. Shot detections are marked with a dot.

Detecting a shot change and encoding the first frame of the new shot as an I-frame produces a gain of up to four decibels in the PSNR of that frame, with an average local gain of 1.5 dB. The new I-frame is used as reference for the encoding of the next pictures within the shot and the better the PSNR of the reference, the better the quality of the next inter-coded frames.

The use of I-frames usually consumes more bits than the use of P-frames, but when the I-frame is placed in a good position (a shot change) the number of bits used to encode the frame is very similar to the P-frame case.

In other situations, when the I-frame consumes a lot of bitrate, the next pictures should be encoded with a few bits and therefore local quality decreases. But this reduction is lower than the quality gain obtained selecting the proper key-frames and the average quality of the encoded sequence increases with respect to the case without shot detection. This quality increase is near 0.5 dB of PSNR.

4.3 Processing Time

Comparing the time needed to encode a detected I-frame (including the time needed to detect the shot change while encoding the picture as a P-frame) with the time needed to encode the same frame as a P-frame without shot detection produces a time gain of up to 6% in the best case or a time penalty of less than 11% in the worst case, depending on the kind of sequence being encoded.

Figure 6 shows the time needed to encode different frames of the same sequence with and without the shot detection algorithm. Four different shot changes appear in frames number 828, 877, 904 and 920. The first two shot changes correspond with low motion scenes and the last ones with high motion shots. With these considerations in mind, it can be noted that:

- In low motion shot changes the time used to re-code the frame as an I-frame is lower than the time used to encode the same frame as a common P-frame.
- In high motion scenes the inter-coding time is lower than the intra-coding time.
- The time needed to encode the next frame after the insertion of an I-frame is always lower when the detection algorithm is used, because the reference used to encode the next P-frame is better and a lot of MB are skipped.

Table 4 shows the average processing time relation between encoding the same shot change frame as a P-frame (t_P) and detecting

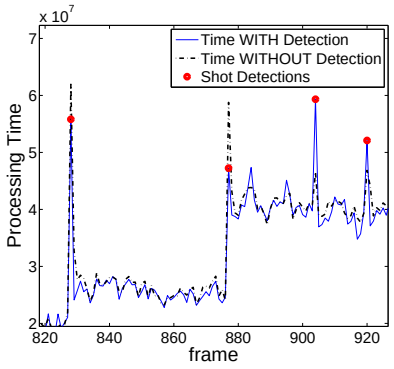


Figure 6: Processing Time needed to encode a frame with shot detection (solid) and without detection (dashed).

LM&SC	*MM&MC*	*HM&HC*	*ALL*
0.94	0.98	1.11	1.04

Table 4: Average processing time relation (t_I/t_P) between the encoding of a detected I-frame and the encoding of the same frame as a P-frame for each video group.

and re-coding the same frame as an I-frame (t_I).

It must be noted that the above-mentioned behavior is fulfilled and a time gain ($\frac{t_I}{t_P} < 1$) is achieved when coding low motion scenes whereas a little penalty appears ($\frac{t_I}{t_P} \geq 1$) in high motion scenes.

The main reason for this behavior is that in low motion scenes, the average number of I-MB of an inter-coded frame is lower than in high motion scenes, and then, the adaptive threshold T_a produces a shot detection with a lower number of coded macroblocks when the frame is being P-encoded.

Moreover, the time needed to encode a P-frame with a very low correlation with its reference is usually very high, because a lot of prediction and motion compensation work has to be done. That is why shot detection and re-coding produces a time gain in situations of low motion scenes.

On the other hand, Table 5 shows the relation between the average time needed to encode a detected I-frame ($\hat{t}_I$) and the average time needed to encode a common P-frame ($\hat{t}_P$).

As it can be seen, $\hat{t}_I$ is always higher than the average $\hat{t}_P$ because it includes the time needed to detect the shot and the re-coding time.

Nevertheless, the use of the detected key-frame as reference to encode the rest of the frames within the shot not only increases the quality of the above-mentioned frames but also decreases the time needed to encode them. This way, a detected I-frame uses more processing time than the average, but the next frames usually use less time. As a whole, the algorithm produces a gain in total processing time when a sequence is being coded.

Table 6 shows the increase of the total processing time of coding with and without the shot detection method ($\Delta T = \frac{T_D - T_{ND}}{T_{ND}}$), where T_D is the total processing time with shot detection and T_{ND} is the total processing time without detection.

	bitrate		
framerate	20 kbps	50 kbps	100 kbps
12.5 fps	1.96	1.69	1.63
6.25 fps	1.63	1.53	1.51

Table 5: Average processing time relation between the encoding of a detected I-frame and a P-frame ($\hat{t}_I/\hat{t}_P$) as a function of the bitrate and the frame rate.

	bitrate		
framerate	20 kbps	50 kbps	100 kbps
12.5 fps	-4.55%	-3.55%	-2.77%
6.25 fps	-3.64%	-2.62%	-1.70%

Table 6: Total processing time relation (ΔT) with and without shot detection as a function of the bitrate and the frame rate.

As it can be seen, a gain of up to 5% of the total processing time is achieved when the shot detection algorithm is applied although the time used to detect a shot change and encode that frame as an I-frame is higher than the average time used to encode a P-frame.

5. CONCLUSION

In this paper we present a shot detection method for low delay, low bitrate video coding. Oriented on compression efficiency, a simple, two thresholds algorithm is proposed and the optimal parameter values for H.264 video coding are discussed. The final results show the goodnesses of the proposed algorithm, achieving high detection rates for all the testing video sequences and an average PSNR gain. Moreover, a total processing time improvement is obtained.

Future work is focused on extending the framework of the algorithm with other picture formats (CIF, etc.) and other GOP configurations, taking B-frames into account (i.e. IPBPBP...).

REFERENCES

[1] Philippe Aigrain, HongJiang Zhang, and Dragutin Petkovic, "Content-based representation and retrieval of visual media: A state-of-the-art review," *Multimedia Tools and Applications*, 3(3), 1996, 179–202.

[2] G. Ahanger and T.D.C. Little, "A survey of technologies for parsing and indexing digital video," *Journal of Visual Communications and Image Representations*, 7(1), 1996, 28–43.

[3] R. Brunelli, O. Mich, and C.M.Modena, "A survey on the automatic indexing of video data," *Journal of Visual Comunication and Image Representation*, 10, 1999, 78–112.

[4] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull, "A unified approach to scene change detection in uncompressed and compressed video," *IEEE Transactions on Consumer Electronics*, 46, 2000.

[5] J. Calic and E. Izquierdo, "Towards real-time shot detection in the mpeg compressed domain," in *WIAMIS'2001-Workshop on Image Analysis for Multimedia Interactive Services*, Tampere, Finland, 2001.

[6] B. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, 5 5(6), 1995, 533–544.

[7] Antonio Albiol, Valery Naranjo, and Jess Angulo, "Low complexity cut detection in the presence of flicker," in *Proc. of Int. Conference of Image Processing 2000*, IEEE, Ed., 2000.

[8] V. Kobla, D. Doermann, and A. Rosenfeld, "Compressed domain video segmentation," CfAR Technical Report CAR-TR-839 (CS-TR-3688), 1996, citeseer.nj.nec.com/vikrant96compressed.html.

[9] J. Sastre, G. Castelló, V. Naranjo, J.M. López and A. Ferreras, "Shot detection method for low bitrate video coding," in *Proceedings of the IASTED International Conference Visualization, Imaging and Image Processing 2005*, IASTED, 2005.

[10] J. Sastre, A. Ferreras, and J.F. Hernández-Gil, "Motion vector size-compensation based method for very low bit rate video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, 10(7), 2000, 1192–1197.

[11] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services," ITU-T, 2003.