FAST VIDEO SEGMENTATION USING ENCODING COST DATA

Ricardo L. De Queiroz[a], Gozde Bozdagi[b] and Taha Sencar[b]

[a]Xerox Corporation
800 Phillips Rd., M/S 128-27E, Webster, NY 14580
queiroz@wrc.xerox.com

[b] Baskent University
Dept. Elec. and Electronics Eng., Ankara, 06530 Turkey
{bozdagi,taha}@baskent.edu.tr

ABSTRACT

This paper presents a simple and effective pre-processing method developed for the segmentation of MPEG compressed video sequences. The proposed method for scene-cut detection only involves computing the number of bits spent for each frame (encoding cost data), thus avoiding decoding the bitstream. The information is separated into I-, P-, B-frames, thus forming 3 vectors which are independently processed by a new peak detection algorithm based on overcomplete filter banks and on joint thresholding using a confidence number. Each processed vector yields a set of candidate frame numbers, i.e. "hints" of positions where scene-cuts may have occurred. The "hints" for all frame types are recombined into one frame sequence and clustered into scene cuts. The algorithm was not designed to distintuish among types of cuts but rather to indicate its position and duration. Experimental results show that the proposed algorithm is effective in detecting abrupt scene changes as well as gradual transitions. For precision demanding applications, the algorithm can be used with a low confidence factor just to select the frames that are worth being investigated by a more complex algorithm. The algorithm is not particularly tailored to MPEG and can be applied to most video compression techniques.

Keywords: MPEG, video segmentation, encoding cost data, cut detection

1. INTRODUCTION

Multimedia documents are becoming an effective part of the corporate life and video is an essential unit of any multimedia document.[1] Editting multimedia documents, however, is not a trivial task. For example, using a word processor one can easily copy, paste or find specific word or group of words in a text document. However, it is not easy to manage multimedia documents as the complex information conveyed in video data cannot be directly analyzed. Video and image contents have to be analyzed semantically in order to convey a scene description. Recently, the field of video partioning (scene cut detection) received increased attention in order to easy the management (editting and processing) of video streams. Most of the work in this area is performed over uncompressed video. However, if a video source is provided in the compressed form, these operations can not be performed until that representation has been decompressed. Some work, however, has been done to attempt to perform scene cut detection on compressed video. The most common information that is being utilized for video partioning in the compressed domain is composed of DCT's DC and AC coefficients and motion vectors. An example of scene cut detection in an MPEG stream that uses DC information is the method due to Meng, Juan, and Chang.[2] In this method, DC coefficients are used to detect cuts on I-frames and ratio of motion vectors are used to detect cuts on P- and B-frames. Another method that uses DC values has been proposed by Yeo and Liu.[3] They used the difference and histogram of DC images extracted from MJPEG and MPEG video for scene analysis. In another work,[4] DC images and coding information such as motion vectors are again used for scene cut detection. Feng[5] proposed an algorithm that utilizes AC information in addition to DC information. The significant change in DC and AC information is detected by comparing the number of bits for each macroblock. Arman et. al.[6] represent each frame of the video sequence by a subset of blocks and use predetermined collections of AC coefficients of these blocks to find the scene cuts. Zhang[7] proposed a hierarchical method where I frames are used as an initial step for cut detection. If higher resolution of shot boundary is required, motion vector count is used among the intervening B-frames.

In all of those methods, the bitstream of the compressed video has to be partially decoded up to a level where the required information can be extracted. If the utilized information increases, the decoding process takes more time. However, if an accurate pre-segmentation can be done by minimum decoding, further improvements can be done based on the resolution required by the user. In this paper, we propose a simple algorithm for this purpose. The algorithm depends on the number of bits (encoding cost) spent for each frame in an MPEG (or MJPEG) compressed data. Such data can be extracted without decompressing the stream in any form. Since all kinds of frames are being utilized, it is more robust than utilizing just I-frames as a pre-step. The method utilizes both the actual and the changes in the encoding cost, which increases its accuracy, as shown by the results.

2. SCENE CUT CHARACTERIZATION

Within a video sequence, the consecutive frames may differ from each oter due to object or camera movement, lighting changes, gradual changes (continuous cuts) or abrupt changes (camera cuts, discontinuous cuts). Although abrupt changes occur between two consecutive frames, gradual changes cover more than two frames. The most common discontinuous cuts found in video sequences are fade in, fade out and dissolve. In a fade, the first scene gradually disappears before the second scene gradually appears. In a dissolve, the first scene gradually disappears while the second scene gradually appears. We can model the intensity *E(x,y,t)* at time t within fade/dissolve as

- Fade-out
$$E(x,y,t) = I(x,y,t_1)\alpha(t) + O(x,y,t_2)[1-\alpha(t)] + B(x,y), \tag{1}$$
- Fade-in
$$E(x,y,t) = O(x,y,t_1)\alpha(t) + I(x,y,t_2)[1-\alpha(t)] + B(x,y), \tag{2}$$
- Dissolve
$$E(x,y,t) = I(x,y,t_1)\alpha(t) + I(x,y,t_2)[1-\alpha(t)] + B(x,y), \tag{3}$$

where $\alpha(t)$ is a decreasing function with $\alpha(t_1) = 1$ and $\alpha(t_2) = 0$; t_1 and t_2 are the starting and ending points of fade/dissolve; $I(x,y,t)$ is the intensity of pixel (x,y) ate time t; $B(x,y)$ is the background image, and $O(x,y,t_1) = O(x,y,t_2) = 0$, $\forall x, y$.

We can see from (1)-(3) that the function α is indeed a blurring function in the temporal direction. Alternatively, abrupt scene cuts, where $t_1 = t_2$ lack this temporal blurring effect. Nevertheless, they have to be similarly modelled in a cut-detection framework. The resulting analysis is inevitably complex, several frames have to be buffered and motion analysis (in some context) has to be performed.

3. FRAME ANALYSIS BASED ON ENCODING COST

3.1. Motivation

As we just discussed, scene cut detection demands large storage and complex operations, which we want to avoid. Instead of circumventing the process, we rather try to circumvent their operations by assuming someone else would do most of the work for us.

MPEG and other video compression schemes rely on removing intra-frame redundancies to some extent. However, most of its compression effectiveness comes from removing inter-frame redundancies, i.e. by performing frame prediction, assuming that an efficient motion estimation algorithm is applied. Video compressors typically will achieve better compression when scenes do not change and will likely fail when a scene-cut is encountered, i.e. they will compress less. If we start with the basic assumption that the MPEG compression scheme was well made, we can assume motion estimation and other compression-related techniques were already applied to the sequence. The output of this analysis process is reflected into how much compression was achieved. Hence, we start by "trusting" MPEG. Regardless of the scene contents, we assume scene-cuts would cause problems for MPEG, by increasing the amount of compressed data it generates for a particular frame. In other words, we do not ignore the computation of the frame contents, we just assume MPEG would do a good job performing it for us.

We will describe how the encoding cost (number of bits spent per frame) information can be used in different frame structures. If the content of the consecutive frames are similar, than the encoding cost will not change, however if there is a discontinuous or continuous cut between frames, the encoding cost vector experience peaks. The key to our algorithym is to identify peaks and recognize which of them represent real scene cuts.

3.2. MPEG frames

MPEG compression format requires the integrity of frame structures, which are composed of the group of pictures (GOP) units. Each GOP starts with an I-frame which are intra-frame DCT encoded using a JPEG-like algorithm. There are two types of interframe encoded frames, P- and B-frames. In these frames, the motion compensated prediction errors are DCT encoded. In the P-frames, only forward prediction is used relative to preceeding I- or P- frames. The prediction of the B-frames can be forward, backward or bidirectional relative to neighbouring I- or P- frames. Each frame is composed of macroblocks (MBs) and depending on the prediction error of the MB, there might be intra coded MBs within P- and B-frames. This occurs when the motion estimation is not accurate enough to compensate the MB.

As seen from the above description, the characteristics of I-, P-, B- frames are different and this difference can be used to detect different types of cuts in the proposed algorithm.

3.3. Cuts on P- and B-frames

P- and B- frames are motion predicted frames which depend on previous/future P- or I- frames. Based on the prediction error of the MB, there might be intra coded MBs within P- and B-frames. This occurs when the motion estimation is not accurate enough to compensate the MB.

When a discontinuous scene cut occurs between two consecutive P- and B- frames, most of the MBs will be intra-frame coded since they cannot be predicted from previous/future I- or P- frames. This is also valid for continuous cuts. During the interval where there is a continuous cut, the encoding cost for the P- and B- frames will be high due to increase in the number of intra coded MBs. However the change in the encoding cost between consecutive frames may not be as significant as discontinuous cuts. Although the encoding cost is very high, the change may not be significant. Due to this reason, in the proposed algorithm, we consider both the high-pass and low-pass components of the encoding cost data in order to detect the scene cuts.

3.4. Cuts on I-frames

The I-frames in an MPEG sequence are coded without using any information from the neighbouring frames. The number of I-frames that are going to be used for MPEG compression of a video sequence is defined before the coding starts. For example, if a GOP structure as $IBBPBBPBBPBBPBB$ is used to encode a video sequence, then every 15th frame will be coded as an I-frame.

If a discontinuous cut occurs between two consecutive I-frames, the content of these frames will be totally different. Therefore the encoding cost will increase if there is a change from a low-activity to a high-activity frame, and decrease in the opposite case. However, if the activity in two different scenes are the same, an analysis based solely on I-frames will not suffice to detect the cut.

The continuous cuts may also effect two consecutive I-frames based on the duration of the transition. However, it is not very likely that the peaks of the encoding cost data can be obtained just by looking at the I-frames. Due to these reasons, an algorithm which is solely based on I-frames can miss the actual scene cuts found in the video sequence. We try to cope with this problem by analyzing encoding cost for I-frames differentially, i.e. by computing the difference between costs of consecutive I-frames as opposed to their absolute value.

4. THE PROPOSED ALGORITHM

The suspected scene changes on I-, P-, B- frames are found by detecting the peaks in both high and low frequency components of the encoding cost of all frames in the coded sequence as follows:

- Compute the encoding cost data for each frame obtaining vector $V(n)$, where n is the frame number.
- Separate vector $V(n)$ into the corresponding components for frames I, P and B $(V_i(m), V_p(m), V_b(m))$, where m is the natural numbering for each individual vector.
- Make I-frame information differential, i.e. $V_i(m) \leftarrow V_i(m) - V_i(m-1)$
- For each of $V_i(m), V_p(m), V_b(m)$, obtain hints of where scene cuts occur (frame numbers m).

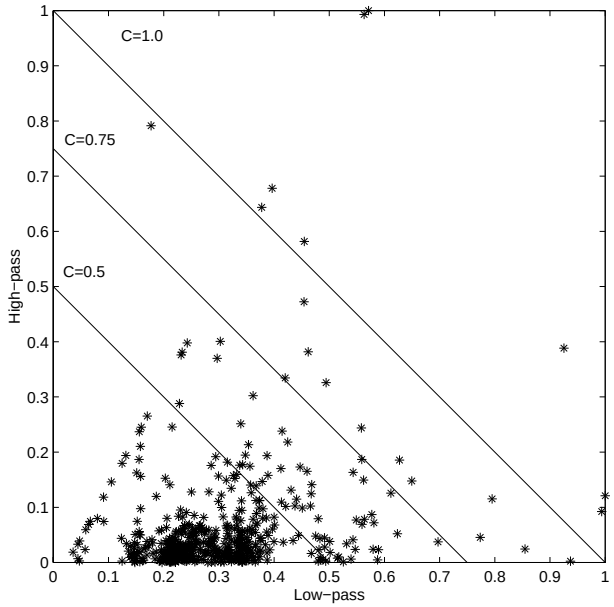

Figure 1. Scene cut decision based on low- and high-pass samples of encoding cost data for P-frames. Samples above diagonal decision line are considered as points of scene cut.

- Combine all detected scene-cut hint frame numbers, respecting original frame numbering n.
- Cluster frames into scene cut regions that are at least T frames apart (T can be set to the length of one GOP), i.e. hints that are T frames apart belong to the same cut.
- Present scene cuts as a 3-number index: start-of-cut, end-of-cut, average number.

The hints of where scene cuts occur in $V_i(m), V_p(m), V_b(m)$ are detected by the following steps:

- Select a low-pass filter. We used the normalized version of the Gaussian filter $[1, 3, 6, 7, 6, 3, 1]$.
- Low-pass filter the input vector $V_x(m)$ to obtain vector $L_x(m)$ where $x = i, p$ or b.
- Obtain high-pass vector $H_x(m) = V_x(m) - L_x(m)$.
- Compute maximum of $L_x(m)$ as M_L.
- Compute maximum absolute value of $H_x(m)$ as M_H.
- Detect a scene cut at moment m as the frame for which

$$|H_x(m)| > M_H \left(C - \frac{L_x(m)}{M_L} \right) \tag{4}$$

  where C is a confidence parameter (Fig. 1).

The key step in the cut detection process is the identification of regions of the encoding cost vector which present both low- and high-frequency contents. This is in effect a peak detector, which is very efficient for detecting the peaks which occur during scene cuts, while being less sensitive to some peaks which are due to other sequence characteristics. The second important step is to cluster the hints into larger regions. The hints are actually what the filtering perceives as several changes. Long transitions such as fades and dissolves can be detected only by clustering all the local hints into one. Also, transitions may propagate for the whole GOP, for that we have to ensure that multiple cuts in a GOP are clustered into one resulting scene cut.

5. ALGORITHM ANALYSIS

5.1. Confidence parameter and pre-processing

Because of its simplicity and of the fact that it does not analyse the image contents, the proposed algorithm is prone to some misclassifications. For example, when panning or another sudden camera motion occurs, MPEG would produce many bits and yet semantically it is arguable if this constitutes a new scene cut or not. False alarms may occur and some real cuts may be undetected.

For those reasons, the algorihtm is presented using a confidence parameter C, shown in Fig. 1. By changing C, one can change the sensitivity of the detected cuts. The lower the value C is, the more cuts are detected including more false positives. High values of C would ensure lower false alarms, but would also increase the number of undetected cuts. A careful balance needs to be achieved. Values around $C = 0.7$ to $C = 1.0$ have been shown to be good compromises in our tests.

If more precision is necessary, one can use the proposed algorithm as a pre-processor for a more complex algorithm by selecting a very low C. This would cause the detection an abundance of false alarms along with virtually all real scene-cuts. However, the number of frames detected is typically largely smaller that the number of frames in the sequence. Therefore, the complex scene-cut detection algorithm would only work on a small percentage of the frames, which translates in large computational savings.

5.2. Types of cuts

The algorithm does not detect types of cuts. It actually detects small abrupt changes that consume too many bits while clustering those changes into scene cut intervals. However, one may be able to estimate the type of the cut based on two factors: (i) duration; (ii) location on the high-pass vs. low-pass plane. Work in this respect is not discussed here.

5.3. Computational complexity

The computational complexity for the proposed algorithm is nearly negligible, compared to known scene-cut detection algorithms.

At the beginning of every frame, in MPEG, there is a local header which is initialized by the 4-byte sequence 0x00001000 (0010). This sequence is aligned and unique in the MPEG stream (i.e. it only occurs once per frame). Thus, if the idea is to count the number of bits spent per frame, we just have to skip and count bytes in the MPEG bitstream, until reaching the sequence 0010. Once the 4-byte sequence is found, the accumulated byte count is the number of bytes actually used to encode a frame (encoding cost). Hence, no decompression whatsoever is necessary. It is not even necessary to understand how MPEG compresses the data, except for the situation when the GOP sequence is not known, in which case one may easily extract this information from the global header.

The filtering computation requires about 11 multiplications and additions per frame, plus a few operation per frame to detect maximums and to test equation (4). Note that those operations are done per frame and it take few milliseconds to compute the cuts from the encoding cost vectors of a group of let us say 1000 frames, in modern desktop computers.

5.4. Experimental results

Several sequences were tested in our simulations. We present three MPEG-1 test sequences with image sizes of 320x240. Those sequences were specially composed to possess several scene cuts as well as some camera panning to demonstarte the performance of the proposed pre-segmentation algorithm. The sequences are recorded from commercially prepared video tapes using BRAVADO video capture card. All the sequences are about 750 frames long and contain both discontinuous and continuous cuts. They are encoded using the pattern $IBBPBBPBBPBBPBBI$. The confidence parameter C was set to $1.0, 0.7$ and 0.9 for I-, P- and B- frames, respectively.

Tables 1-3 summarizes the actual versus detected cuts on the MPEG sequences. The cuts are detected for I-, P- and B- frames separately and then combined to obtain the overall figures. Figs. 2-4 show the detected cuts on separate frames and on the complete encoding cost sequences. For comparison purposes the frame difference (sum of absolute pixel differences) obtained from the I-frames of decompressed $Seq3$ is also given in Fig. 5. It is clear that although the discontinuous cuts can easily be distinguished, it is not easy to locate the continuous cuts. Figures also

show that it is not possible to detect all the cuts just by using I frames. Therefore an algorithm that combines all I-, P- and B-frames is much more robust as a pre-step than using only I frames.[7]

The use of high and low pass components of the number of bits also enables us to obtain the cuts more robustly. In Fig. 6, it is shown the plot of the high-pass component of the number of bits obtained from $Seq3$. As seen from the figure, it is not possible to distinguish continuous cuts from the high pass component.

6. CONCLUSION

We propose an algorithm that uses the encoding cost data obtained from the number of bits spent for I-, P- and B-frames in an MPEG coded video sequence as a pre-step in video segmentation. As shown by the results, the proposed algorithm gives approximate position of the cuts. Also, when there is a motion followed by dissolve or fade, it is not possible to distinguish between these two different effects. However in the second run, these cuts can be improved using additional data such as chrominance information, motion vectors. The advantages of the algorithm is that it provides more robust results as a pre-step than other algorithms found in the literature and it has very low computational complexity. In fact, in the context of video processing, the computation is virtually negligible.

REFERENCES

1. A. Furth, S. Smoliar, and H. Zhang, *Video and Image Processing in Multimedia Systems*, Kluwer Academic Publishers, 1995.
2. J. Meng and S. F. Chang, "Tools for compressed-domain video indexing and editing," *SPIE* **2670**, pp. 180–191, 1996.
3. B. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Trans. on Circuits and Syst. for Video Tech.* , 1995.
4. Y. Nakajima, K. Ujihara, and A. Yoneyama, "Universal scene change detection on mpeg-coded domain," *SPIE* **3024**, pp. 992–2003, 1997.
5. J. Fang. et al, "Scene change detection algorithm from mpeg video sequences," *ICIP'96* **2**, pp. 821–824, 1996.
6. F. Arman, A. Hsu, and M. Chiu, "Image processing on compressed data for large video databases," *ACM Multimedia* , pp. 267–272, 1993.
7. H. Zhang. et. al., "Video parsing using compressed data," *SPIE* **2182**, pp. 142–149, 1995.

Table 1. The actual and the detected scene cuts for $Seq1$ where the numbers are the frame indices

	ACTUAL CUTS	START	END	MIDPOINT
Dissolve	10-24	10	30	20
Fade	148-169	152	166	159
Dissolve	616-655	616	661	638
Cut	680	676	676	676
Dissolve	724-737	718	739	728

Table 2. The actual and the detected scene cuts for $Seq2$ where the numbers are the frame indices

	ACTUAL CUTS	START	END	MIDPOINT
Cut	226	226	226	226
Dissolve	300-330	328	328	328
Dissolve and local motion	348-445	340	448	394
Dissolve	455-501	457	472	463
Cut	618	616	616	616
Dissolve	652-656	654	661	657
Dissolve	812-823	808	829	818

Table 3. The actual and the detected scene cuts for $Seq3$ where the numbers are the frame indices

	ACTUAL CUTS	START	END	MIDPOINT
Zoom	10-90	10	85	47
Cut	136	136	136	136
Dissolve	155-166	151	172	162
Cut	181	181	181	181
Cut	200	196	196	196
Fade	254-266	254	263	258
Wipe	320-360	320	351	335
Wipe	420-470	428	471	449
Local and Global motion	502-520	506	519	512

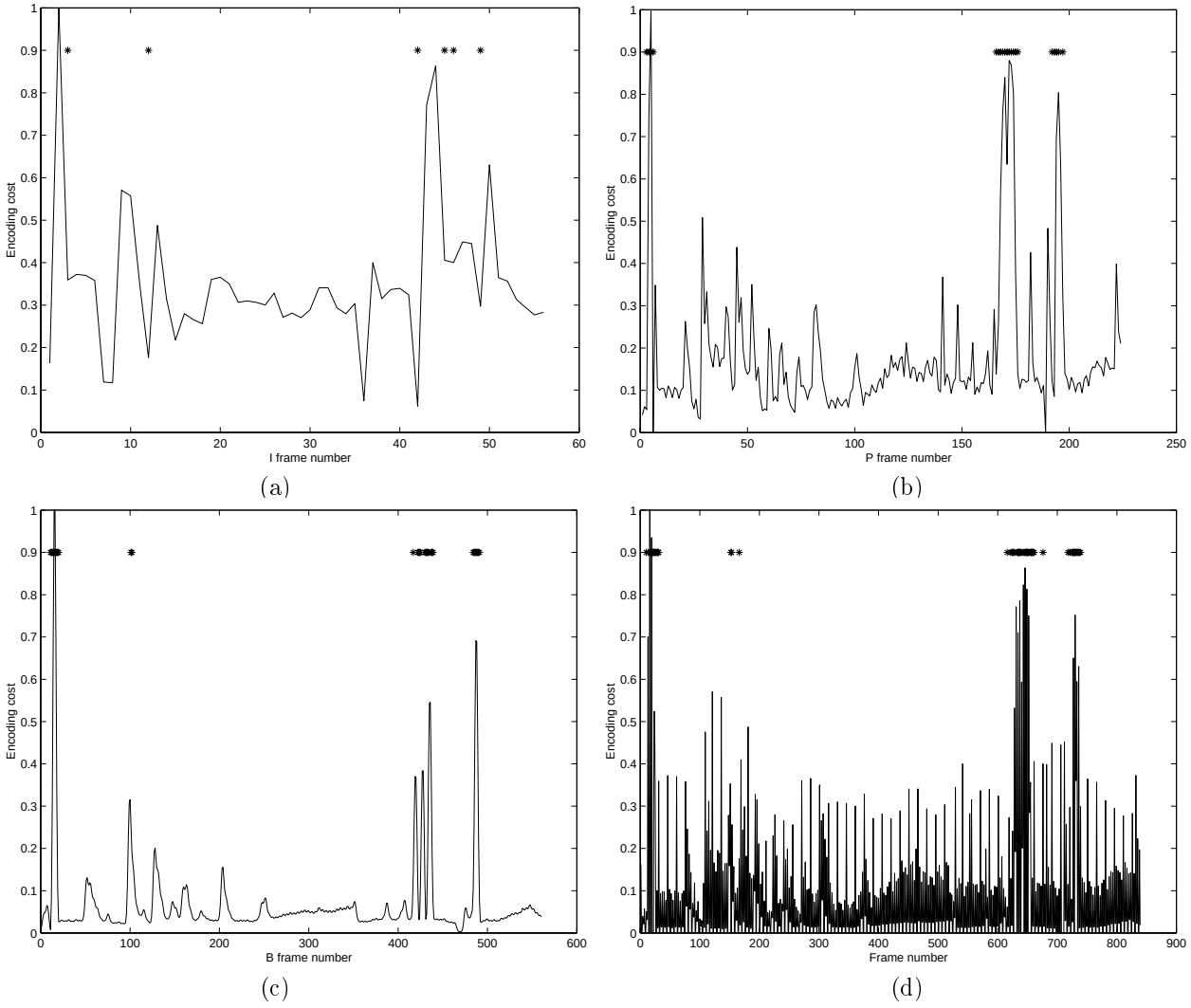

Figure 2. Detected scene cuts on a) I-frames, b) P-frames, c) B-frames, d) combined result for $Seq1$.

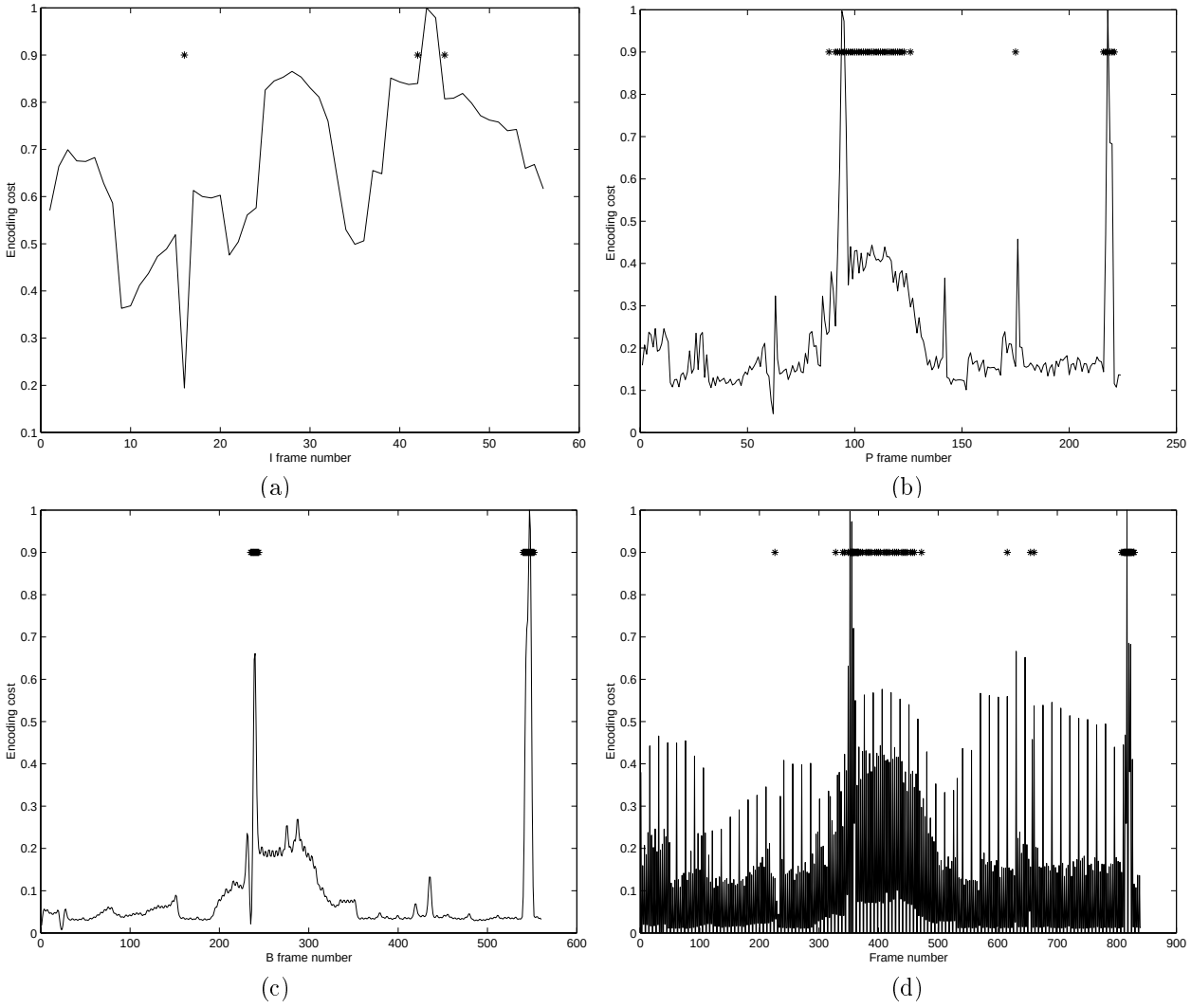

Figure 3. Detected scene cuts on a) I-frames, b) P-frames, c) B-frames, d) combined result for $Seq2$.

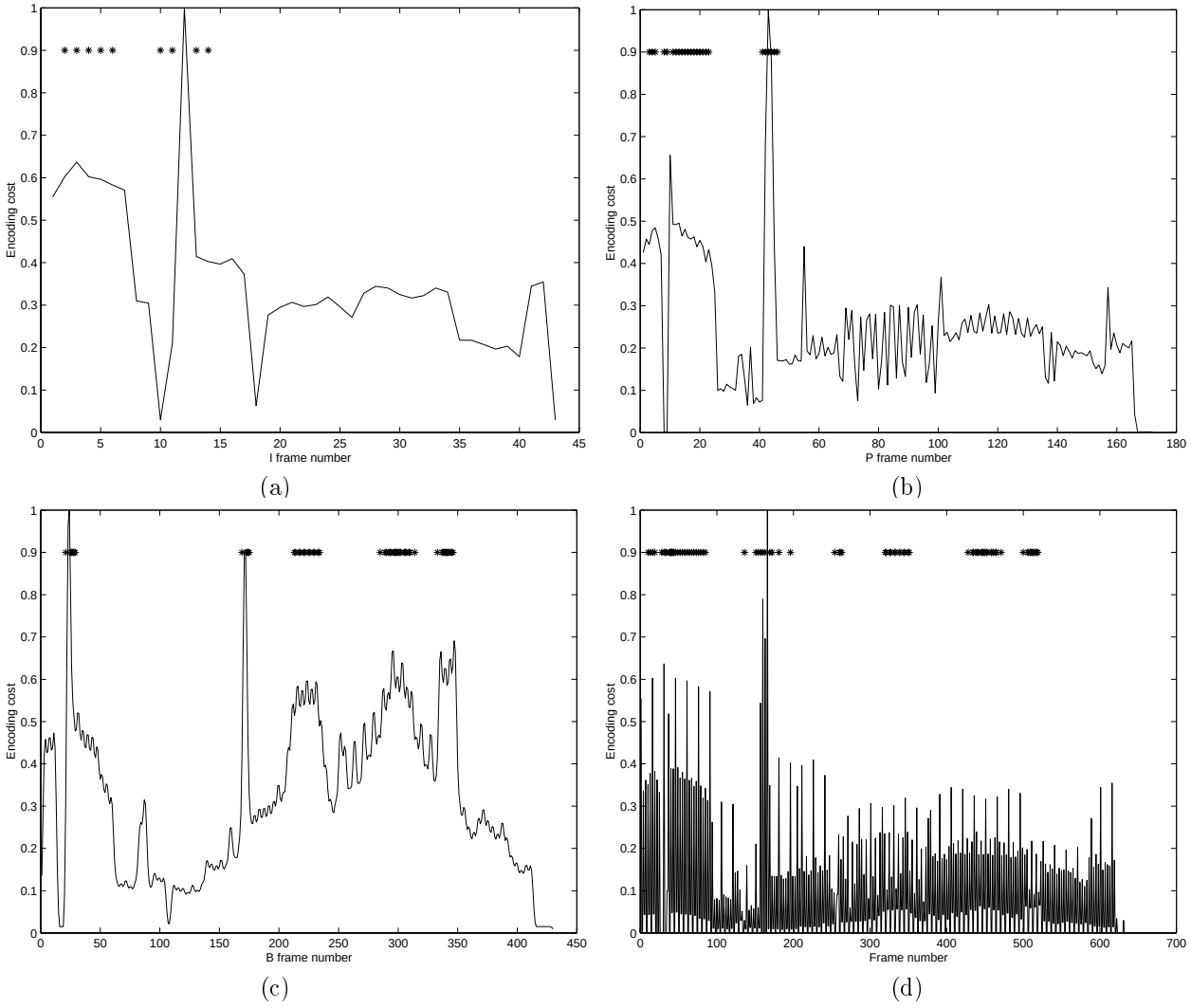

Figure 4. Detected scene cuts on a) I-frames, b) P-frames, c) B-frames, d) combined result for $Seq3$.

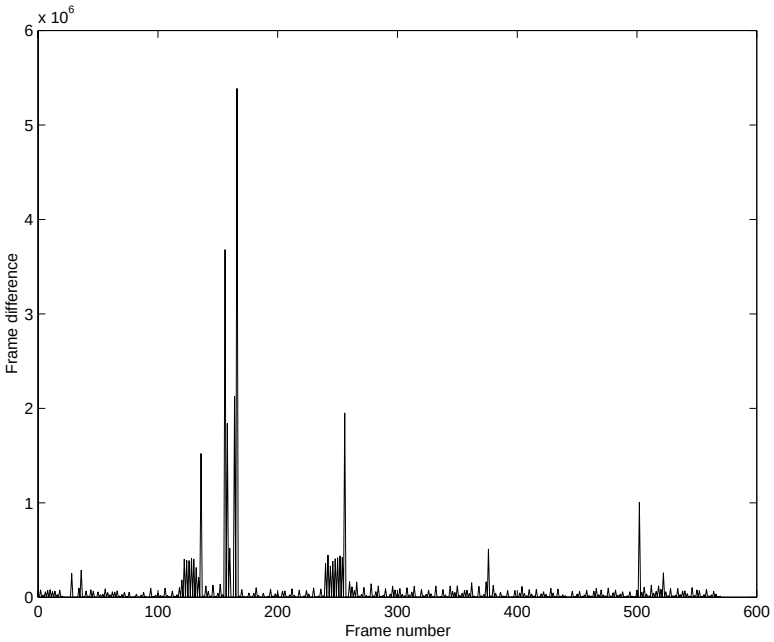

Figure 5. Frame difference for decompressed I-frames of $Seq3$.

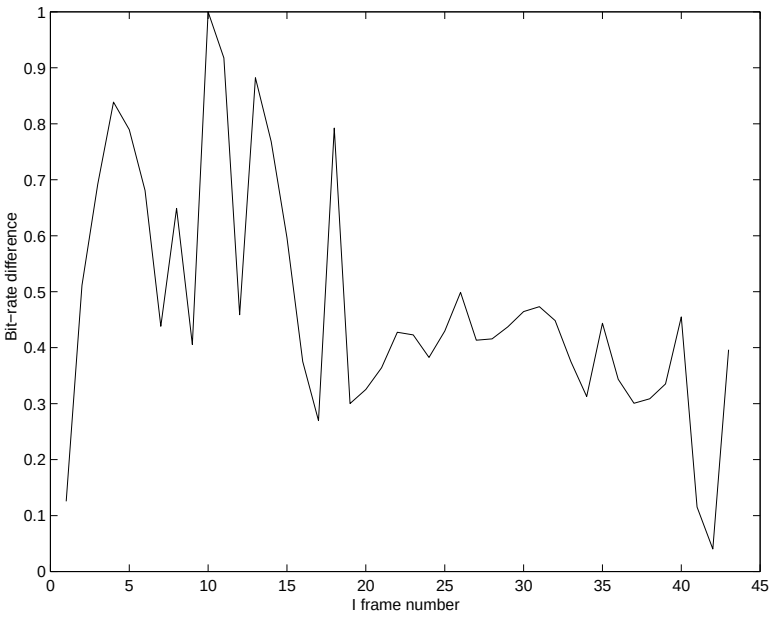

Figure 6. Bit-rate difference for I-frames of $Seq3$.