Cost-effective scene change detection algorithm for real-time H.264 rate control

Changhyun Lee,[a] Yunho Jung,[b] Seongjoo Lee,[c] Yunje Oh,[d] and Jaeseok Kim[a]
[a]Yonsei University, Department of Electrical and Electronic Engineering, Seoul, South Korea
[b]Korea Aerospace University, School of Electronics, Telecommunication and Computer, Seoul, South Korea
[c]Sejong University, Department of Information and Communication Engineering, Seoul, South Korea
[d]Samsung Electronics Company, Limited, Department of TN Research and Development Center, Suwon, South Korea
E-mail: hyun2@yonsei.ac.kr

Abstract. For frames involving an abrupt scene change, almost all macroblocks are encoded using intramode in H.264/AVC. Before encoding such frames, one must first determine whether a scene has changed to perform the appropriate rate control. Since the computation load of H.264/AVC is heavy, a more cost-effective algorithm for detecting scene changes is necessary for real-time operations. We propose such an algorithm for high-motion video, and the results show that the proposed algorithm reduces the computational load by 37 to 92% from existing algorithms.
[DOI: 10.1117/1.2890162]

Subject terms: abrupt scene change; computational load; intracoding; H.264; rate control.

Paper 070583LR received Jul. 9, 2007; revised manuscript received Dec. 14, 2007; accepted for publication Jan. 16, 2008; published online Mar. 11, 2008.

1 Introduction

H.264/AVC is a state of the art video coding standard with a heavy computational load (CL), which affects encoder performance in many real-time applications. Several papers reducing the computational load have been published, and it is one of the important issues on H.264/AVC.[1]

The reference software for H.264/AVC includes a function that encodes an interframe in intramode. This feature is especially useful for frames involving an abrupt scene change (ASC), because the ASC frame is difficult to correlate with the previous frames. To achieve higher video quality and lower fluctuation in video quality, an appropriate rate control (RC) is necessary for an ASC frame.[2] This requires the detection of an ASC before the ASC frame is encoded.

If a distinct difference between consecutive frames can be detected by a suitable dissimilarity metric (DM), an ASC can be declared whenever the DM exceeds a given threshold. Various such DMs have been published and can largely be categorized into two groups: those designed for use with compressed video[3] and those designed for use with uncompressed video.[4,5] Since an ASC detection for RC cannot be performed before encoding the current frame, DMs for compressed video are not suitable. Therefore, only DMs for uncompressed video are considered in this work, which are further categorized into two subgroups: pixel-based and histogram-based DMs. Pixel-based DMs are simpler, but are inferior when analyzing high-motion video or when dealing with partial changes within a whole frame, such as flashes and objects that appear abruptly. To detect ASCs robustly, the well-known mean square error (MSE) between two consecutive frames is suggested as a DM.[4] This DM is defined as follows:

$$\mathrm{DM}_{\mathrm{MSE},i} = (\mu_i - \mu_{i-1})^2 + |\sigma_i^2 - \sigma_{i-1}^2|, \tag{1}$$

where μ_i and σ_i represent the mean intensity value and corresponding standard deviation for frame i, respectively. Since the resulting performance of pixel-based DM is still poor for high-motion video, both pixel-based and histogram-based DMs are successfully used in Ref. 5. However, since a histogram-based DM necessitates conditional and loop statements, the algorithm of Ref. 5 yields a very high CL, especially when applied to high-motion video, requiring more than 15 times that of the frame-layer H.264 RC from our experiments. The resulting CL is so heavy that the real-time operation of H.264 RC becomes burdensome when working with ASC frames. Since the existing ASC detection algorithms either perform poorly or result in heavy CL for high-motion video, we propose a cost-effective ASC detection algorithm for real-time H.264 RC and compares its performances with Refs. 4 and 5 in high-motion video.

2 Proposed Detection Algorithm

Most DMs for uncompressed video use the pixels of the previous frame to verify how strongly the consecutive frames correlate. This process demands an increased CL to save the previous frame. If instead we can obtain information about correlation in the process of H.264 encoding, the CL can be reduced. In Ref. 2, the predicted peak signal-to-noise ratio (PPSNR) is presented for use in the H.264 RC. The PPSNR is calculated using the current picture and the previously reconstructed frame. Since this reconstructed frame has already been saved in the process of H.264 encoding, PPSNR can be computed without increasing the CL.

As discussed in Sec. 1, pixel-based DMs are defective when applied to high-motion video and partial frame changes. To fix these problems while minimizing the CL, we divide a whole frame into several parts. Figure 1 illustrates how to divide a whole frame into 12 parts. Based on these ideas, we propose a new DM for frame i and part x as follows:

$$\mathrm{DM}_{\mathrm{pro},i}^{x} = \frac{\mathrm{PPSNR}_{i,i-1}^{x}}{\left(\dfrac{1}{i-s_j}\right)\displaystyle\sum_{k=s_j+1}^{i}\mathrm{PPSNR}_{k,k-1}^{x}}, \tag{2}$$

where $\mathrm{PPSNR}_{m,n}^{x}$ denotes the PPSNR between picture m and reconstructed frame n for part x of a whole frame, and s_j is the frame number corresponding to scene change j. Even though PPSNR has the correlation between two consecutive frames to itself, taking various encoding condi-

0091-3286/2008/$25.00 © 2008 SPIE

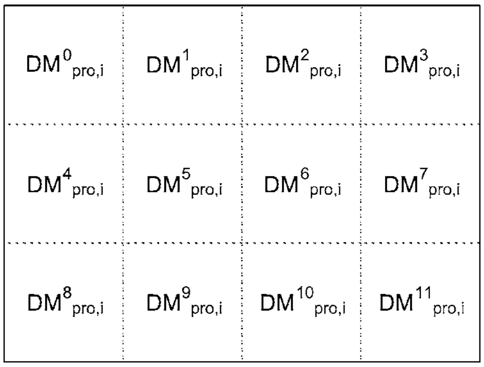

Fig. 1 Frame divided into 12 parts to improve tolerance of partial changes.

tions such as target bit rates into account, $\mathrm{DM}^{x}_{\mathrm{pro},i}$ is used as a DM because the PPSNR^x averaged over a bunch of frames without an ASC can be the representative PPSNR^x for a given condition. An ASC is declared for frame i whenever the following is satisfied:

$$\sum_{x=0}^{N_f-1} C^x \geq \alpha \cdot N_f, \tag{3}$$

where α and N_f denote the counting threshold and the number of divided parts for a whole frame, respectively. C^x is given by

$$C^x = \begin{cases} 1; & \mathrm{DM}^{x}_{\mathrm{pro},i} < \beta \\ 0; & \text{else} \end{cases}, \tag{4}$$

where β indicates the decision threshold for the divided parts. Since DM is calculated in each divided part, it prevents a partial frame change from influencing the performance of ASC detection. For example, if a frame is partially changed in the third part, $\mathrm{DM}^{3}_{pro,i}$ is obtained within the third part. The partial change in the third part does not influence DMs for the other parts. Therefore, this method mitigates the defect of pixel-based DM.

3 Experimental Results

We evaluated the proposed algorithm along with Refs. 4 and 5 for three high-motion sequences of QCIF size: "Goal of the Tournament for 2006 Worldcup" (Worldcup), "60 Greatest Playoff Moments for NBA" (NBA), and "Final Fantasy X-2" (FF-X2). Further information about these test sequences is tabulated in Table 1. The proposed algorithm was implemented on JM9.8[6] to use the previously saved reconstructed frame in the H.264 encoding process. The other algorithms were also implemented on JM9.8 to allow CL comparisons within the same platform. The parameters for all tests were set as follows: only the first frame within a group of pictures was encoded in intramode; there were no B frames and no frame skip; the search range was 16; the number of reference frames was one; rate distortion optimization was enabled; the entropy coding was context adaptive binary arithmetic coding; the buffer size was set to half of the target bit rates and the frame rate was 30 fps. The target bit rates were set to 57.6, 115.2, and 230.4 kbps to verify their effect on the proposed algorithm.

Table 1 Video sequences for performance evaluation.

Sequence	Sequence Comments	Number of Frames	Number of ASCs
Worldcup	Soccer highlight	6843	13
NBA	Basketball highlight	12303	48
FF-X2	Animation highlight	7138	159

Table 2 Comparison results of False and Miss detections between proposed and existing algorithms. See Refs. [4,5].

Sequence	Algorithm		Number of False	Number of Miss	$DP_{FalseMiss}$ (%)
Worldcup	Ref. 4		2	2	30.8
	Ref. 5		0	1	7.7
	Proposed (kbps)	57.6	0	2	10.3
		115.2	0	1	
		230.4	0	1	
NBA	Ref. 4		89	20	227.1
	Ref. 5		14	8	45.8
	Proposed (kbps)	57.6	36	9	99.3
		115.2	39	8	
		230.4	43	8	
FF-X2	Ref. 4		97	60	98.7
	Ref. 5		11	50	38.4
	Proposed (kbps)	57.6	30	53	49.9
		115.2	31	48	
		230.4	31	45	

Table 2 summarizes the detection performance of the tested algorithms as measured by the number of False and Miss detections. A False detection indicates that an ASC is declared, but does not occur. Contrarily, a Miss detection indicates that an ASC occurs, but is not declared. The thresholds of the proposed algorithm were set after intensive experiments: $\alpha=0.75$, $N_f=12$, and $\beta=0.70$. The thresholds of the other tested algorithms were modified to improve performance for the test sequences as follows: the threshold of Eq. (1) in Ref. 4 was set to 1450, and the threshold of absolute difference frame variance (ADFV) after histogram equalization with normalization (HEN) in Ref. 5 was decreased from 50 to 30. As shown in Table 2, the proposed algorithm executed fewer Miss detections than the existing algorithms. Regarding False detections, the proposed algorithm was superior to Ref. 4 in all test sequences and equivalent to Ref. 5 in Worldcup. In NBA and FF-X2, the proposed algorithm executed more False detections than Ref. 5. This occurred by virtue of HEN because the two sequences, especially NBA, contain many frames involving flashes. However, HEN requires a very high CL, which seriously obstructs the real-time H.264 RC.

When a False detection occurs, bits are allocated unnecessarily. This causes an increase in the subsequent quantization parameters and a decrease in video quality. When a Miss detection occurs, bits are allocated inappropriately. In this case, the distortion propagates to the subsequent frames.[7] Based on these observations, the sum of False and Miss detections is a better parameter for evaluating an ASC detection algorithm in viewpoint of H.264 RC. Using the sum of the False and Miss detections, we define a parameter indicating the detection performance numerically as follows:

$$DP_{\text{FalseMiss}} = \frac{N_{\text{FalseMiss}}}{N_{\text{ASC}}}, \tag{5}$$

where $N_{\text{FalseMiss}}$ and N_{ASC} denote the number of False and Miss detections and the number of ASC frames in a sequence, respectively. In the proposed algorithm, $DP_{\text{FalseMiss}}$ was computed after averaging $N_{\text{FalseMiss}}$ for three distinct target bit rates. $DP_{\text{FalseMiss}}$ of the proposed algorithm decreased by 65.7% in comparison to Ref. 4, but increased by 22.5% in comparison with Ref. 5. The detection performance of the proposed algorithm was almost not affected by the target bit rates.

To compare the CL of the tested algorithms, we used Intel® VTune™ Performance Analyzer 8.0 after optimizing the CL of Refs. 4 and 5 in C code level. When computing a variance of Eq. (1) in Ref. 4, we obtained a squared value from memory without multiplication after rounding a mean value to the nearest integer. In Ref. 5, although a noncandidate ASC frame can be filtered using the mean absolute frame difference (MAFD), we cannot predict whether an ASC will occur in the next frame. Therefore, HEN must be executed in every frame and the equalized pixels must be saved. Since this process requires a high CL, we instead saved the original pixels of frames (i-1) and (i-2) to perform HEN if necessary for frame i. To further reduce the CL, all pixels of a frame were copied with a width of one frame when saved to memory.

Figure 2 shows the CLs of the three tested algorithms. For comparison, the CL of the frame-layer H.264 RC is also included. We set the Intel® VTune™ to the time-based mode using the operating system (OS) timer. The sampling interval was one millisecond. For corroboration, all tests were performed on three computers: Intel® Core™2 CPU 6400 at 2.13 GHz, Intel® Pentium® D CPU 3.00 GHz, and Intel® Pentium® D CPU 3.40 GHz. The OS for all three computers was Microsoft® Windows® XP. The timer samples were obtained by each computer and for each sequence. The timer samples shown in Fig. 2 are the sum of the samples measured by each computer. On summing timer samples over three computers and three sequences, the proposed algorithm achieved a CL 37.1% lower than that of Ref. 4 and 92.1% lower than that of Ref. 5.

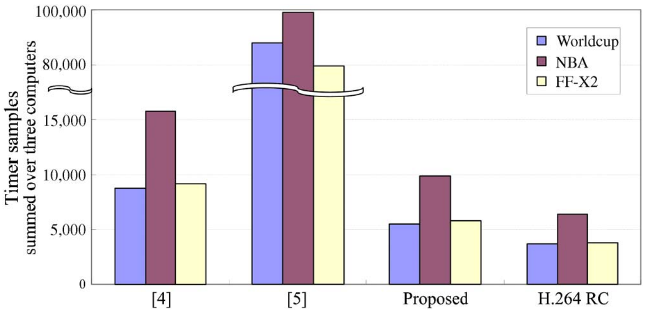

Fig. 2 Timer samples presenting the computational load of each abrupt scene change detection algorithm.

4 Conclusion

We propose a cost-effective ASC detection algorithm, because the existing ASC detection algorithms either perform poorly or result in heavy CL that seriously obstruct the real-time H.264 RC for very high-motion video. Compared with a pixel-based algorithm, the detection performance of the proposed algorithm increases by 66%, while the CL is reduced by 37%. Compared to an algorithm using both pixel-based and histogram-based DMs, the detection performance of the proposed algorithm is 23% lower. However, the CL is dramatically reduced by 92%. Taking both the detection performance and CL into consideration, the proposed algorithm is more suitable than the existing algorithms for detecting ASCs in real-time H.264 RC.

Acknowledgments

This research was supported by University ITRC Project and partly by the TN R and D Center in Samsung Electronics Company, Limited.

References

1. I. Werda, F. Kossentini, M.-A. Ben Ayed, and N. Massmoudi, “Analysis and optimization of UB video’s H.264 baseline encoder implementation on Texas Instruments’ TMS320DM642 DSP,” *IEEE Intl. Conf. Image Process.*, pp. 3277–3280 (2006).
2. X. Yi and N. Ling, “Improved H.264 rate control by enhanced MAD-based frame complexity prediction,” *J. Visual Commun. Image Represent* **17**(2), 407–424 (2006).
3. I. K. Sethi and N. Patel, “A statistical approach to scene change detection,” *Proc. SPIE* **2420**, 26–37 (1995).
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 Trans. Consum. Electron.* **46**(3), 769–779 (2000).
5. X. Yi and N. Ling, “Fast pixel-based video scene change detection,” *IEEE Intl. Sym. Circuits Sys.* **4**, 3443–3446 (2005).
6. See http://ftp3.itu.ch/av-arch/jvt-site/reference_software/jm98.zip.
7. C. H. Lee, Y. H. Jung, S. J. Lee, Y. J. Oh, and J. S. Kim, “Real-time frame-layer H.264 rate control for scene-transition video at low bit rate,” *IEEE Trans. Consum. Electron.* **53**(3), 1084–1092 (2007).