Rate Control of H.264 Encoded Sequences by Dropping Frames in the Compressed Domain

Spyridon K. Kapotas and Athanassios N. Skodras, *Senior Member*, IEEE
School of Science and Technology, Hellenic Open University Patras, Greece
{s.kapotas,skodras}@eap.gr

Abstract

Abstract—A new technique for controlling the bit-rate of H.264 encoded sequences is presented. Bit-rate control is achieved by dropping frames directly in the compressed domain. The dropped frames are carefully selected so as to either eliminate or cause non perceptible drift errors in the decoder. The technique suits well H.264 encoded sequences such as movies and tv news, which are transmitted over wireless networks.

1. Introduction

The H.264/AVC standard [1] is becoming the dominant standard in video coding. Moreover, due to the nature of the H.264 standard (high compression, low bitrate, etc.), the H.264 encoded sequences suit very well applications such as Video On Demand and video streaming over the Internet or other networks. On the other hand, rate control is an important issue in video streaming applications for both wired and wireless networks. Rate control techniques fall into the video transcoding category when they do not take place during the encoding of the original sequence. A typical scenario is delivering a high quality media source to various receivers (PCs, cell phones, PDAs, etc) on wireless and wireline networks. The rate controller, hereafter the *transcoder*, must generate appropriate bitstreams directly from the original bitstream in order to accommodate different network bandwidths. Another scenario is delivering the media source to a receiver, which supports only a lower frame rate. In that case the transcoder must reduce the frame rate. In our case the target application is a bit-rate controller of low complexity and with low memory requirements, which can control the bit-rate of H.264 encoded movies in real time.

Basically, there are two ways of controlling the bit rate in the compressed domain: the temporal transcoding, e.g. dropping frames and the bit rate transcoding in a per frame basis. Several bit-rate and temporal transcoding techniques have been proposed in the past. An overview of various techniques is given in [2, 3]. Lefol et al [4] evaluate the performance of some known bit rate transcoders when these are applied to an H.264 bitstream. Most of them were presented many years ago and although they could be applicable in H.264 video sequences, they do not take into account the special characteristics of the H.264 standard. In this paper we propose a new bit rate transcoder of low complexity, which works directly in the compressed domain in real time and either eliminates or causes non-perceptible drift errors.

The rest of the paper is organized as follows. In Section 2 we present the problem. In Section 3 we describe the main concepts that the proposed technique is based on. The technique itself is presented in Section 4. The simulation results are given in Section 5. Finally, in Section 6 conclusions are drawn.

2. Problem Formulation

In this Section we describe the target application and the issues that we need to address. Video sequences are stored in a media streaming server. The server is connected to a gateway through an error free high speed channel, e.g. Ethernet. The video is transmitted to the gateway and then to various devices with wireless capabilities, such as smart phones, PDAs, Tablet PCs and Laptops, which may belong to the same or different networks or subnets.

Apparently, the gateway must perform some bit rate control in order to accommodate either the bandwidth of the different networks or the network congestion. However, the gateway may be a standalone device,

© 2010 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

even embedded, with limited CPU and memory capacities. Therefore, with regard to CPU and buffering requirements, complicated (high demand) transcoders cannot be implemented. On the other hand low complexity transcoders, such as the open-loop transcoders, generate drift errors. Real time operation of the transcoder is of course mandatory. The proposed technique addresses all of these issues.

3. Main concepts

In this Section the main concepts, which the proposed technique is based on, are described.

Frame (and Slice) Types

I-Frame: The macroblocks in an I frame can be predicted only using the Intra prediction model [5].

P-Frame: The macroblocks in a P frame are predicted using the Inter prediction model. The macroblocks are predicted from one or more (usually five) reference frames before the current frame.

B-Frame: B frame is essentially the same as a P frame, with the exception that each inter predicted macroblock in a B frame may be predicted from one or two reference frames, before or after the current frame in temporal order.

IDR-Frame: The H.264 standard introduces the Instantaneous Decoder Refresh (IDR) frame. The IDR is the same with an I frame. However, the subsequent P or B frames of an IDR frame are not allowed to use frames, prior to the IDR, as references.

D-Frame: The H.264 standard also introduces the Disposable (*D*) frame. The *D* frame cannot be used as a reference for other frames. *D* frames play a key role to the proposed technique.

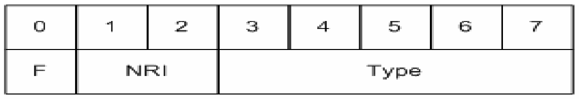
Network Abstraction Layer (NAL)

H.264 makes a distinction between a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL) [5]. A coded video sequence is represented by a sequence of NAL units. Each NAL Unit (NALU) contains a header and a set of data corresponding to coded video data (Raw Byte Sequence Payload, RBSP). In the context of this work many times the term *frame* implies a NALU, which contains a frame and the opposite, i.e. the term *NALU* implies a frame. Fig. 1 shows the first octet of the NALU header. What is interesting is that the NALU header contains useful information about the video data of the NALU. Specifically:

The *F* or *forbidden_zero_bit* was included to support gateways. The H.264 specification declares a value of 1 as a syntax violation.

The *NRI*, or *nal_ref_idc* signals the relative importance of that NALU. A value of 00 in binary format, indicates that the content of the NALU is not used to reconstruct reference frames for inter frame prediction.

The *Type* or *nal_unit_type* specifies the NALU payload type as defined in Table 7-1 of [1].



0	1	2	3	4	5	6	7
F	NRI		Type				

Fig. 1 First octet of the NAL Unit (NALU) Header.

4. Bit Rate Transcoder

The proposed technique is mainly based on the disposable *(D)* frame concept. A *D* frame can be dropped without generating drift error because it is not used as a reference for other frames. The problem is that the H.264 encoder does not generate *D* frames by default. Even if it does, the frequency that the *D* frames appear in the bitstream may not be the desirable one. However, many other frames could be regarded as *D* or "almost" *D* frames, meaning that only a small number of macroblocks within these frames are used as references for inter predicted macroblocks in other frames. These frames could also be dropped causing a non-perceptible drift error. There are also other frames, which could be dropped under certain conditions. All of the aforementioned frames will be referred to as droppable altogether.

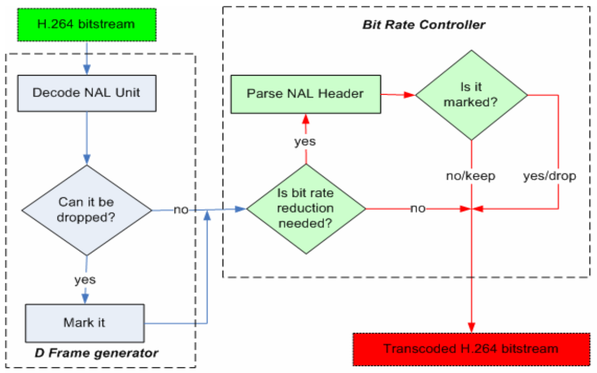


Fig. 2 The block diagram of the proposed bit rate transcoder.

The purpose of the proposed transcoder is two-folded. First it must discover these droppable frames and signal them in the NALU header (Fig. 1). Then it must drop the marked NALUs according to the bit rate requirements of the channel. Fig. 2 shows the block diagram of the proposed technique. There are two distinct components, namely the *Droppable Frame Generator (DFG)* and the *Bit Rate Controller (BRC).*

The main advantage of separating the two components is that they can be implemented in different devices. The *BRC* can be implemented in a gateway even of limited capabilities, while the *DFG* can be implemented in a powerful stream server. In this way one can correct or implement more advanced techniques for generating droppable frames while maintaining the same simple software in the gateway. After all, gateways cannot be easily upgraded.

Droppable Frame Generator (DFG)

The *DFG* takes into account the special encoding characteristics of the H.264 encoder in order to choose those frames that are candidates for being dropped. The *DFG* applies several rules in order to increase the number of candidate frames.

Rule 1: If a frame is an *IDR,* then the previous frame is surely a non reference *D* frame; this can be safely dropped without affecting the visual quality of the decoded video.

Rule 2: If a frame is *I* and the number of the reference frames is one, then again the previous frame is a *D* frame.

Rule 3: The $X(k-r)$ is a *D-like* frame if $X(k-r+1)$ is an *I* frame and the following conditions are met for the $X(k-n)$ frame.

$X(k-n) \rightarrow I$ or

$$T_{ref} \leq \sum_{i=0}^{r-(n+1)} \sum_{j=0}^{N_{k-n}} idx(i,j) \leq N_{k-n} \qquad (1)$$

where $0 \leq n \leq r+2$, I denotes an *I* frame, r is the number of the reference frames that have been used during the encoding, N_{k-n} is the total number of the luma blocks and sub-locks in frame $X(k-n)$, $idx(i,j)$ is the reference index of the reference list 0 of the block j that equals to i and T_{ref} is a threshold that is used to denote a *D-like* frame. In this work $T_{ref} = N_{k-n}$.

Rule 4: Frame $X(k)$ can be dropped if the following condition is met:

$$T_m \leq \sum_{i=0}^{P} MB_P(i) + \sum_{j=0}^{M} MB_{16x16}(j) \leq N \qquad (2)$$

where N is the total number of the inter-predicted MBs in frame $X(k)$, MB_P denotes a skipped MB, MB_{16x16} denotes a 16×16 MB with (0, 0) motion vector and T_m is a threshold, which is used to denote a droppable frame; in this work $T_m = 3N/4$.

The *DFG* must signal the droppable frames in the NALU header without violating its syntax. The *DFG* will do so by modifying the NALU header (Fig. 1) as follows:

```
- If the NALU contains a Droppable Frame
- Set NRI = 0
- Set F = 1 (Rules 1,2)
- Set F = 0 (Rules 3)
```

where *NRI = 0* means that the current NALU contains a droppable frame and *F = 1* indicates that none of the frames, which precede the current frame, is used as a reference by the frames that follow. For droppable frames, which have been detected by rule 4 we set *F = 0* in order to protect the previous frames since rule 4 detects one droppable frame at a time.

Bit Rate Controller (BRC)

The *BRC* will drop the frames based on the bit rate constraints of the transmission channel. The *BRC* is as simple as a parser of the NALU header. When NRI=0 and F=0 the *BCR* must drop k frames in order to meet the bit rate constraint. Variable k is crucial for the bit rate reduction as well as for the visual quality of the decoded video. The larger the k is, the greater the bit rate reduction achieved. However, this is achieved at the expense of the video quality. The value of k clearly depends on the number of frames between two successive droppable frames. In other words, if there is a sufficient number of droppable frames within a sequence, the value of k is small and close to one. In general k is defined as in eq. (3)

$$1 \leq k < N_D \qquad (3)$$

where N_D is the number of frames between two successive droppable frames. This is the reason why BRC needs to buffer up to $N_D + 1$ frames.

5. Simulation Results

The simulation tests were performed on a Windows XP system with an Intel CPU T2400 at 1.83 GHz and 1.50 GB of RAM. Both of the *DFG* and the *BRC* components were implemented in C++ and were integrated in the same software. In order to evaluate the performance of the proposed technique we measured the impact of each rule on the bit rate and the video quality separately. Some well-known video sequences in CIF and QCIF format were used. In order to objectively evaluate the performance of the proposed technique, two metrics were used, namely the bit rate variation and the average Peak Signal to Noise Ratio (APSNR) for the whole sequence.

The bit rate variation is calculated as

$$\mathrm{VAR}_{\mathrm{rate}} = \frac{\mathrm{R}_{\mathrm{mod}} - \mathrm{R}_{\mathrm{ref}}}{\mathrm{R}_{\mathrm{ref}}} \times 100(\%) \qquad (4)$$

where R_{mod} is the bit rate of the transcoded bitstream and R_{ref} is the original bit rate.

The APSNR is calculated as

$$\mathrm{APSNR} = \sum_{n=1}^{N} \frac{PSNR(n)}{N} \qquad (5)$$

where $\mathrm{PSNR}(n)$ is the PSNR of frame n and N is the total number of the frames in the video sequence.

We used JM16.2 reference encoder and decoder [6] for our experiments. The baseline profile of the encoder was used and the configuration parameters retained their default values apart from the ones, which are shown in Table I. The configuration parameters of the decoder retained the default values with the exception of the error concealment parameter, which was set to one, i.e. the *Frame Copy* error concealment method [7] was used.

TABLE I
CONFIGURATION PARAMETERS OF THE ENCODER

Rule	Sequence	Parameters
1	akiyo qcif/150f	IDRPeriod=5 ReferenceFrames=5
2	bridge-close qcif/150f	IntraPeriod=8 ReferenceFrames=1
3	grandma qcif/150f	IntraPeriod=10 ReferenceFrames=5
4	bridge-far cif/89f	IntraPeriod=10 ReferenceFrames=5

The Bit Rate Variation and the average PSNR of each rule are shown in Table II. The minus sign in the bit rate variation denotes a bit rate reduction. Moreover, the larger the APSNR is, the better the visual quality is. Fig. 3 shows the PSNR per frame between the original and the transcoded sequence *grandma*.

6. Conclusions

In this paper we presented a new bit rate transcoding technique, suitably adapted for the H.264 encoded sequences. It works directly in the compressed domain, it is simple and very effective under certain conditions. The technique may be further improved by discovering more rules which detect droppable frames.

TABLE II
BIT RATE VARIATION AND APSNR

Rule/k		Bit Rate Var. (Kb/s)	APSNR (dB)
1	1	-6.475694	88.13477
	2	-7.040593	67.87395
	3	-8.377952	65.93387
2	1	-6.255422	92.32405
	2	-10.296116	42.00108
	3	-14.074904	41.31070
3	1	-3.641593	95.07396
	2	-7.493855	70.67783
	3	-11.168888	66.59341
4	1	-26.828371	62.02823

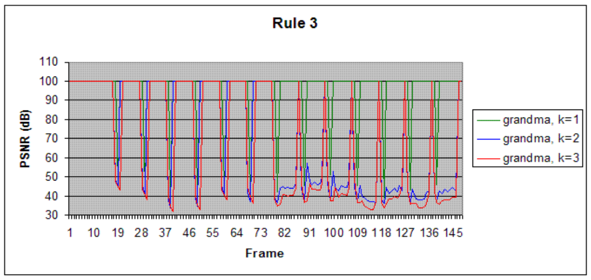


Fig. 3. PSNR values per frame for the grandma sequence.

Acknowledgements

The support by PENED (contract no 03EΔ832) is gratefully acknowledged.

References

[1] ITU-T Rec. (05/2003) "Advanced Video Coding For Generic Audiovisual Services", available online at http://www.itu.int/rec/T-REC-H.264-200903-I/en.

[2] J. Xin, C.W. Lin and M.T. Sun, "Digital Video Transcoding", IEEE Proceedings, vol. 7, issue 1, pp. 84-97, January 2005.

[3] A. Vetro, C. Christopoulos and H. Sun, "Video Transcoding Architectures and Techniques: An overview," IEEE Signal Process. Mag., vol. 20, no 2, pp. 18-29, March 2003.

[4] D. Lefol, D. Bull and N. Canagarajah "Performance Evaluation of Transcoding Algorithms for H.264", IEEE Trans. Consumer Electronics, vol. 52, issue 1, pp. 215-222, February 2006.

[5] I.E.G. Richardson, "*H.264 and MPEG-4 Video Compression*", John Wiley & Sons Ltd, 2003.

[6] JVT Reference Software version JM 16.2, available online at http://iphome.hhi.de/suehring/tml/

[7] S.K. Bandyopadhyay, Z. Wu, P. Pandit and J.M. Boyce "*An Error Concealment Scheme for Entire Frame Losses for H.264/AVC*", Proc. IEEE Sarnoff Symposium, March 2006.