PERFORMANCE OF WATERMARKING AS AN ERROR DETECTION MECHANISM FOR CORRUPTED H.264/AVC VIDEO SEQUENCES

Eva Rodriguez Rodriguez, Luca Superiori, Olivia Nemethova and Markus Rupp

Institute of Communications and Radio-Frequency Engineering,
Vienna University of Technology, Austria
Gusshausstrasse 25/389, A-1040 Vienna, Austria
Email: {erodr, lsuper, onemeth, mrupp}@nt.tuwien.ac.at

ABSTRACT

In this work we investigate the performance of watermarking as an error detection method for H.264/AVC encoded videos. The efficiency of a previously proposed forced even watermarking is evaluated in a more realistic error-prone transmission scenario. A less invasive watermarking scheme, the force odd watermarking, is proposed as alternative. In order to handle possible decoding desynchronization at the receiver, we implement a syntax check error detection mechanism together with watermarking and evaluate its performance.

1. INTRODUCTION

The 3rd generation of mobile systems allows multimedia services such as video streaming, video call and conferencing. The Universal Mobile Telecommunications System (UMTS), standard developed by the 3rd Generation Partnership Project (3GPP) in Europe, includes the baseline profile of H.264/AVC [1] in its specifications for the Packet-switched Streaming Service (PSS) [2].

In mobile communications, the encoded H.264/AVC video content is encapsulated as Real-time Transport Protocol (RTP) payload, and transmitted over the User Datagram Protocol (UDP) /Internet Protocol (IP). As the wireless channel is an error prone channel due to its strong time variation, the packets at the receiver might be corrupted. Typically, if the UDP checksum at the decoder is not correct the whole packet is discarded and the missing information is concealed, i.e. by motion compensation using information from previous packets.

However, corrupted packets may still contain some correct information. The localization of the errors inside a packet is not a trivial task, which calls for refined error detection mechanisms. The aim of error detection methods at bitstream level is to locate the errors within the packet and recover the correct information. Because of the H.264/AVC Variable Length Coding (VLC), a single bit inversion can make the decoder unable to distinguish the correct boundaries of codewords. We will refer to this effect as decoding *desynchronization*. When such desynchronization occurs, it propagates until the end of the packet. In this case, and without any resynchronization mechanism or additional detection/decoding mechanisms, the decoding of the stream may even be impossible. However, the information elements stored before the error occurrence are still correct and can be exploited by the decoder.

Several error detection methods have already been proposed, some of the most relevant can be found in [3]. In this work, we consider the use of watermarking to detect errors in a videostream. The invisible watermarking technique consists in modifying data content, so that the receiver is able to recover the watermark and thus verify the correctness of the information. This was introduced in [4] as an error detection method, based on the work proposed for H.263 [5] in [6], by watermarking the DCT coefficients or residuals in H.264/AVC videos. Under the assumption that the residuals represent the majority of the encoded video information, both in [4] and [6], errors modifying the value of the reconstructed residuals are considered and tested, achieving good results in the performance.

However, in a more realistic scenario, errors can affect any part of the video sequence. For this reason, in this work we analyse the effect of errors over the whole H.264/AVC video content, thus evaluating the performance of watermarking in this new scenario. Specifically, two watermarking schemes are tested. We performed simulations using the Force Even Watermarking (FEW) presented in [4]. It will be shown that the usage of FEW causes a high initial degradation at the encoder. Taking this into consideration, we also introduce and test a Force Odd Watermarking (FOW) approach. When errors occur over the whole video stream, desynchronization can happen, and with watermarking alone the decoder would not be able to deal with the decoding of some information elements. In order to handle desynchronization, we implement the syntax check error detection method presented in [7] together with watermarking, and perform simulations under the new error conditions.

The rest of the paper is structured as follows: Section 2 explains the basis of H.264/AVC and syntax check method. Section 3 analyzes previous watermarking schemes and proposes force odd watermarking as an improvement of force even watermarking. The evaluation through simulation of the two watermarking schemes, under more realistic error conditions, is presented in Section 4, focusing on distortion, error detection probability and error detection delay at the decoder. Section 5 analyzes the influence of errors on critical H.264/AVC parameters. Section 6 contains conclusions and final remarks.

2. H.264/AVC STRUCTURE AND SYNTAX CHECK

The H.264/AVC encoding mechanism uses the correlation between basic frame units called macroblocks (MB) to generate Intra (I) encoded frames, exploiting the spatial correlation of the frame, or Inter predicted (P) frames, using as reference the previous frames. The encoded macroblocks are stored into the Network Abstraction Layer Unit (NALU). In the case of mobile networks transmission, the NALUs containing the video information are further encapsulated as RTP payload, over UDP/IP. Since the size of an IP packet is limited by the network's Maximum Transfer Unit (MTU), the number of MBs stored in a NALU is bounded as well. We refer to the macroblocks contained into an IP packet as a picture slice. Figure 1 shows the encapsulation process.

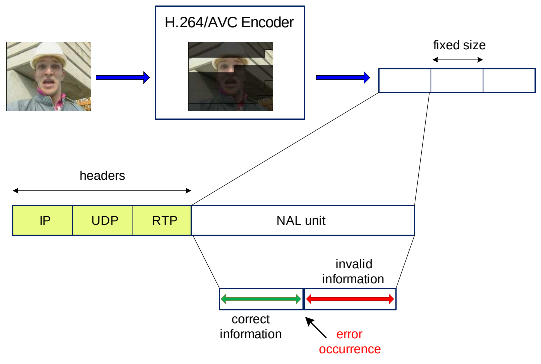


Figure 1: H.264/AVC stream encoding and encapsulation for network transmission.

The H.264/AVC bitstream comprises the information elements associated to each MB, sequentially stored in a fixed structure. As discussed in the introduction, bit inversions might make the decoder unable to reconstruct the information, from that point until the end of the packet. The impaired stream can lead the decoder to illegal actions that cannot be handled by the standard developer's decoder JM [8]. In [7], a decoder able to detect errors within the received video sequence by means of *syntax check* has been presented. The detection is performed by exploiting the codewords, range and significance of the H.264/AVC information elements. Most of the codewords in H.264/AVC are entropy coded and can be decoded without the need of a look-up table, but rather with a contextual analysis of each information element. Entropy coded words can be divided into Exp-Golomb coded codewords and VLC level codewords.

- **Exp-Golomb coded codewords:** these codewords make use of the following structure:

$$\underbrace{0_1 \ldots 0_M}_{M} 1 \underbrace{b_1 \ldots b_M}_{M}.$$

  The first M zeros and the first one form the *prefix*, whereas the last M bits represent the *info* field. From these fields, the encoded value can be reconstructed. Errors affecting the *prefix* modify the value of M, thus causing desynchronization at the decoder, while errors in the *info* field cause deviations of the decoded parameters and possibly affect the following elements.

- **VLC level codewords:** correspond to the context adaptive residual coding. The levels or residuals of the prediction are coded in this way. The encoding uses one out of seven VLC-N procedures, where N is an integer parameter in the range [0,6] and depends on the value of the previous decoded levels. Generically, the VLC structure is represented as follows:

$$\underbrace{0_1 \ldots 0_M}_{M} 1 \underbrace{i_1 \ldots i_{N-1}}_{N-1} s.$$

  In particular, VLC-0 codewords are just defined by the first M zeros and the following one, whereas for higher N the whole structure is considered. VLC-0 codewords are highly susceptible to errors since the whole information is contained in the leading zeros. For VLC-N the first $M+1$ bits are critical and can cause desynchronization, whereas errors lying in the info field could just imply the use of a false VLC procedure for the following decoded items. Errors in the sign field are neither detectable, nor harming.

The H.264/AVC decoder is differentiated in two steps; a reading phase, that reads and partitions the raw bitstream into codewords, and the decoding phase, that transforms the codewords into information elements, used to reconstruct the slice. Syntax check is able to detect a limited subset of errors (illegal codewords, that do not have correspondence in the look-up table, codewords out of range, or contextual errors, that cause the decoder to perform illegal actions), and suffers of a *detection delay*, meaning that usually errors are not detected "on the fly" but rather after a certain amount of MBs, when one of the previous types arises. The macroblocks decoded between the error occurrence and the error detection might be impaired by severe visual artifacts.

Nevertheless, the method is able to detect and handle desynchronization, meaning that the decoder is able to detect than an error occurred and to apply a concealing method to the MBs from the point the error is located until the end of the packet. This reason, together with the fact that it has no additional cost in the implementation, makes syntax check a perfect candidate to be combined with watermarking when considering a more realist error scenario.

3. WATERMARKING

Watermarking (WM) consists in modifying the encoded information before transmission by using a specific pattern or rule also known at the decoder. After transmission, the correctness of the pattern is verified, in order to determine whether transmission errors have occurred. The FEW scheme was first introduced for H.263 in [6], in [4] watermarking schemes for H.264/AVC are proposed, also including a FEW approach. A fragile watermark is forced onto quantized DCT coefficients at the encoder; when reconstructing the macroblocks, the decoder checks the correctness of this mark and detects errors at the MB level. The main problem of this approach though, is that even when invisible watermarking is used, the modification of the residuals introduces an initial distortion into the video sequence.

FEW consists in changing the values of one or more AC coefficients, starting at a given position p inside a *sub-macroblock* (part of a MB), and following the zig-zag scan until the end of the given sub-macroblock. Considering a sub-macroblock of size $N \times N$ (4×4 in our case) with DCT coefficients a_n, where a_1 corresponds to the DC value (not watermarked) and $n \in [2, N^2]$; each a_i with $i \in [p, N^2]$, the resulting ${a_i}^{(w)}$ watermarked coefficient follows:

$$ {a_i}^{(w)} = \begin{cases} a_i & ; \quad |a_i| \text{mod } 2 = 0 \\ a_i - \text{sign}(a_i) & ; \quad |a_i| \text{mod } 2 = 1; \end{cases} $$

where mod 2 stands for the operation modulo 2 and $\text{sign}(a_i)$ is the *sign* function.

Figure 2 shows how the coefficients are watermarked following the zig-zag scan for $p = 2$. In this case, all coefficients but the DC value are watermarked.

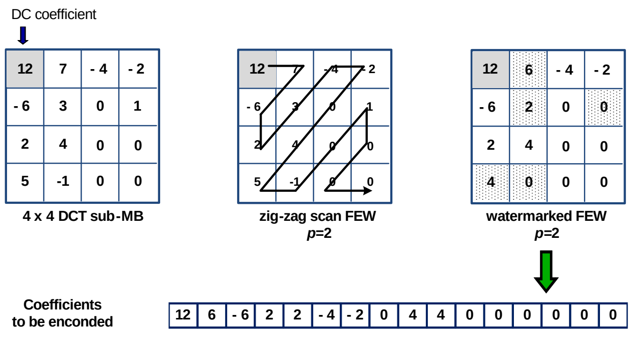


Figure 2: WM process

One of the main drawbacks of watermarking is that it causes an initial degradation of the video quality at the encoder. Low values of p imply more watermarked coefficients, and thus a bigger loss in quality. Taking this into consideration, we propose and test a Force Odd Watermarking (FOW) following the same idea of FEW. In FOW, even values are turned to odd depending on the value of p (it is important to notice that coefficients with value "0" are not modified). FOW achieves a considerable improvement compared to FEW in terms of initial degradation, as shown in Figure 3.

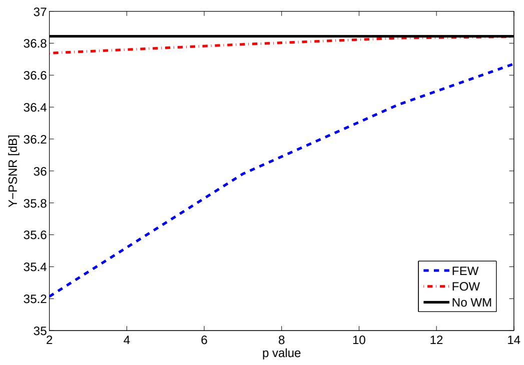


Figure 3: Distortion at the encoder depending on p value. QP = 26, 30 f/s.

We analysed the coefficient values in P frames and found out that around 90% of the non-zero coefficients of a MB have an absolute value of "1" (Figure 4), meaning that in the majority of cases FEW is turning the so-called *trailing ones* to zero. The encoding of the residuals or levels (DCT coefficients of a sub-macroblock) is performed by encoding the non-zero value levels, the trailing ones, and signalising the zero values appearing between two non-zero levels. The N zero values appearing after the last non-zero coefficient are not encoded. For this reason, the use of FEW results in higher initial degradation, whereas FOW (performing changes in even coefficients) has no effect on the trailing ones and performs a less aggressive watermarking.

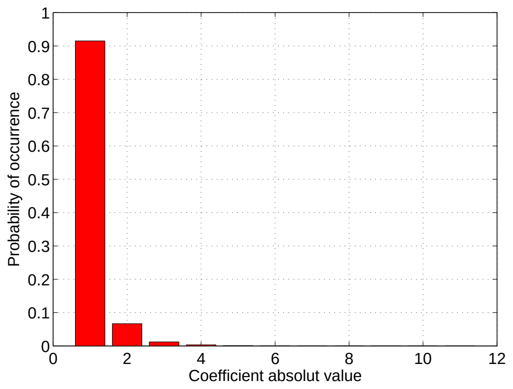


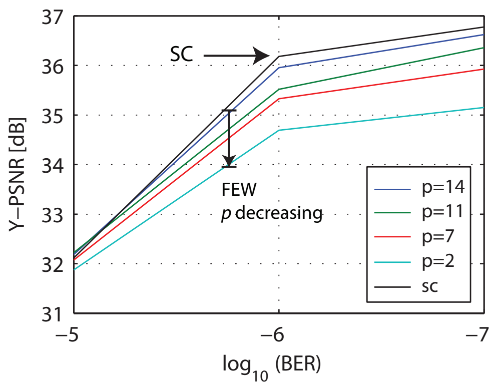
Figure 4: Normalized distribution of the coefficient values per MB in P frames using the "foreman" video sequence. QP=26, 30 frames/s.

4. PERFORMANCE EVALUATION

The watermarking schemes (FEW/FOW) and syntax check have been programmed by modifying the Joint Model (JM) reference software [8].

In this work, we use the "foreman" test video sequence of 400 frames, encoded at 30 f/s and with packet size limited to 800 bytes. Considering a mobile communications environment, the applicable resolution to mobile phones screen QCIF (Quarter Common Intermediate Format)(176×144) is used. We use Quantization Parameter (QP) of 20, 26 and 30. The considered frames are I and P frames, with a Group of Pictures (GoP) of 10, meaning an I frame every 10 frames. In this way, the temporal propagation stops with the reception of a new I frame. A binary symmetrical channel (BSC) affecting the whole H.264/AVC stream is generated, simulating different Bit Error Rates (BERs) ($10^{-5}, 10^{-6}, 10^{-7}$). We use the "copy-paste" method for error concealment, consisting in concealing the corrupted MBs by copy-pasting the correspondent spatial MB of the previous decoded frame. Watermarking is applied to all the MBs of a frame, and only to P frames. In fact, in P frames, the residuals of a 4×4 subblock do not influence the encoding nor the reconstruction of the neighboring blocks. Moreover, we decided not to impair the quality of the I frames in order to keep an unaltered source of temporal prediction. We consider watermarking starting positions $p = 2, 4, 7, 11, 14$, and only watermark the luminance (*luma*).

To evaluate the performance of the method we analyse the distortion at the receiver, error detection probability and error detection delay. The distortion is evaluated by using the Peak Signal to Noise Ratio (PSNR). More specifically, just luminance distortion is compared in this work, which is denoted as Y-PSNR.



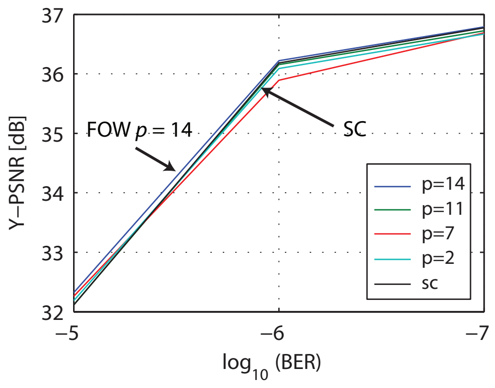


Figure 5: FEW and FOW degradation at the decoder depending on the BER. "SC" denotes syntax check.

The effect of the initial degradation, shown in Figure 3, prevails after the transmission with errors. Figure 5 shows the quality at the decoder depending on the BER. FEW is not able to overcome the initial degradation, whereas FOW improves the quality compared to syntax check alone when $p = 14$. This effect is also shown when a more realistic scenario, with 10f/s and different QPs, is simulated (Figure 6). In this case, FOW achieves better results for all p values, compared to syntax check and FEW. In our scenario, higher bit error rates are not considered because for those the received video stream is extremely degraded, i.e. a BER of 10^{-3} would cause every packet to be erroneous.

As the amount of coefficients watermarked in FEW is higher than in FOW, the error detection probability for FEW is also higher and directly related to the value of p, as shown in Table 1.

Due to desynchronization, to compute the number of error detections it is important to have a correspondence between the inserted error and the actual detection. To achieve this, we perform simulations introducing one error per slice. We define *error detection delay* as the number of MBs between the MB where the error actually occurs and the MB in which the error is detected. Once the error is detected, the concealment of MBs is applied until the end of the slice.

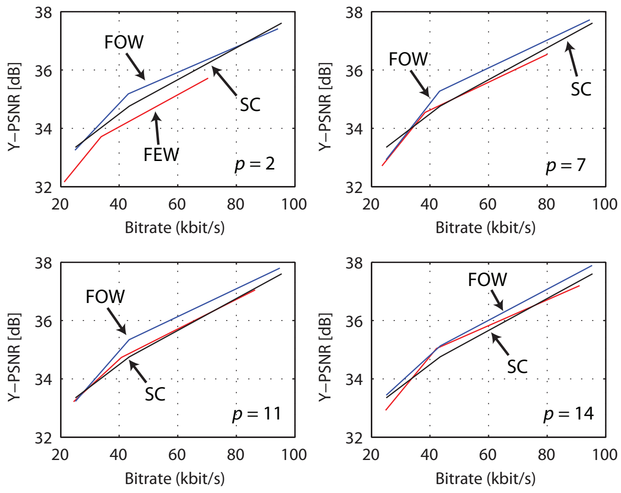


Figure 6: Y-PSNR versus the bitrate of the encoded video sequence.

We evaluate error the detection delay by using an Empirical Cumulative Distribution Function (ECDF), as shown is Figure 7. In terms of error detection delay, lower values of p significantly improve the detection distance, and the decrease of the performance is directly related to the value of p.

Table 1: Error detection probabilities (QP = 26, 30 f/s)

p	FEW	FOW
2	65.1%	59.8%
7	62.5%	55.2%
11	58.1%	52.6%
14	54.4%	51.4%

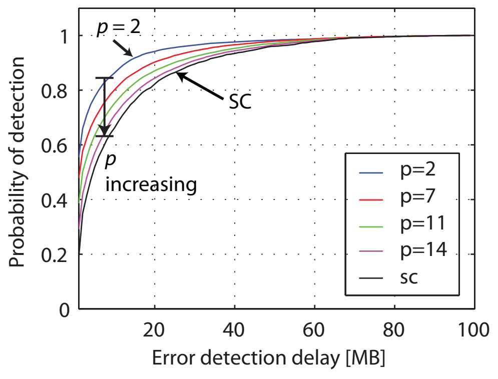
5. H.264/AVC CRITICAL PARAMETERS

The results obtained for watermarking in presence of more realistic errors differ appreciably from those presented in [4]. In order to justify this discrepancy, in the following a critical analysis of the results is provided.

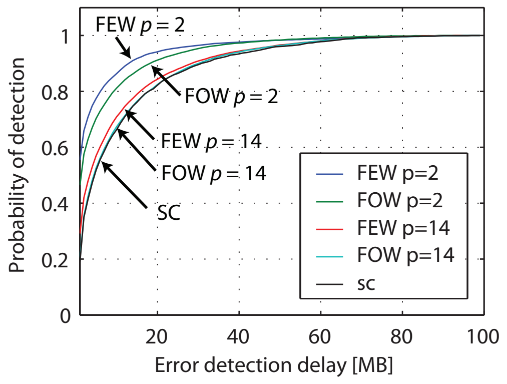
Watermarking is able to recognize errors, when any detectable change on the Luma coefficients happens. Nevertheless, if during the reconstruction of a macroblock, the residuals are not read, watermarking cannot be used to detect errors in that macroblock.

After analyzing the effect of desynchronization in H.264/AVC, it has been noted how two specific parameters strongly affect the correct reconstruction of the residuals: `mb_skip_run` and `coded_block_pattern`.

In case the temporal prediction performed using the predicted motion vectors does not require further correction by means of residuals, a macroblock is encoded as "skipped". The amount of consecutive skipped macroblocks is signalised in the code. The parameter representing this information is called `mb_skip_run` and is encoded as an exp-Golomb codeword. For zero skipped MBs, the



(a) FEW



(b) FOW

Figure 7: Detection delay distribution for FEW and FOW

codeword associated to the `mb_skip_run` field consists in one single bit with value 1. Because of desynchronization, under the assumption that the ones and the zeros are equally distributed in the code, the probability of the first bit being "one" is 50%. An exp-Golomb codeword, whose first bit is a one, represents an encoded value bigger than zero. This would lead the decoder to skip a certain number of MBs, delaying the possible detection by means of watermarking.

The field `coded_block_pattern` (also known as "cbp") determines which of the four 8×8 submacroblocks contains residuals. Similarly to the `mb_skip_run` the `coded_block_pattern` is encoded using a variable length Exp-Golomb entropy coding. The codeword signalising no residuals is the bit 1. However, most of the encoded macroblocks contain residual information, meaning their cbp is therefore starting with the bit 0. Following the previously discussed reasoning, it is highly probable that the first bit of the cbp turns to be a 0, signalising no residuals to be decoded.

The presented discussion helps understanding why the presented results reduce the performance of watermarking as an error detection mechanism when applying real errors in the whole bitstream. However, further deep investigation of the code under different error assumptions is required.

6. CONCLUSIONS

The performance of watermarking combined with syntax check as an error detection method for H.264/AVC videos is tested in this work. Two different schemes have been proposed in this paper: after observing the significant distortion introduced by force even watermarking, force odd watermarking is proposed as a less invasive implementation. Differently from previous works, in this article we considered errors occurring in the whole stream. In this more realistic scenario, the improvements brought by watermarking schemes are reduced. A detailed analysis of the corrupted video has shown that the corruption of specific parameters, due to errors, strongly affects the performance of watermarking.

7. ACKNOWLEDGEMENT

The authors thank mobilkom austria AG for technical and financial support of this work. The views expressed in this paper are those of the authors and do not necessarily reflect the views within mobilkom austria AG.

REFERENCES

[1] ITU-T "H.264. Series H: Audiovisual and multimedia systems, Infrastructure of audiovisual services - coding of moving video. Advanced video coding for generic audiovisual services", November 2007. Available at: `http://www.itu.int/rec/T-REC-H.264`.

[2] 3GPP TSG TR 26.937 "Transparent end-to-end packet switched streaming service (PSS); Real-time Transport Protocol (RTP) usage model", ver 5.0.0, 2003.

[3] O. Nemethova, *Error Resilient Transmission of Video Streaming over Wireless Mobile Networks*. PhD thesis, Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology, 2007. Available at: `http://publik.tuwien.ac.at/files/pub-et_12661.pdf`

[4] O. Nemethova, G. Calvar Forte, and M.Rupp, "Robust Error Detection for H.264/AVC Using Relation Based Fragile Watermarkin," in *Proc. of 13th Int. Conf. on Systems, Signals and Image Processing (IWSSIP)*, Budapest, Hungary, September 2006.

[5] ITU-T "H.263. Series H: Audiovisual and multimedia systems, Infrastructure of audiovisual services - coding of moving video. Video coding for low bit rate communications", January 2005.

[6] M. Chen, Y. He, and R.L. Lagendijk, "A Fragile Watermark Error Detection Scheme for Wireless Video Communications," *IEEE Transactions on Multimedia, 7(2)*, pp. 201–211, April 2005.

[7] L. Superiori, O. Nemethova, and M. Rupp, "Performance of a H.264/AVC Error Detection Algorithm Based on Syntax analysis," in *4th Int. Conf. on Mobile Computing and Multimedia (MoMM)*, Yogiakarta, Indonesia, pp. 49–58, December 2006.

[8] H.264/AVC Software Coordination, "Joint Model Software," ver. 13.0. Available at: http://iphome.hhi.de/suehring/tml/.