Journal of Mobile Multimedia, Vol. 0, No. 0 (2005) 000–000

PERFORMANCE OF A H.264/AVC ERROR DETECTION ALGORITHM BASED ON SYNTAX ANALYSIS

LUCA SUPERIORI, OLIVIA NEMETHOVA, MARKUS RUPP
Institute of Communications and Radio-Frequency Engineering
Vienna University of Technology
Gusshausstrasse 25/389, A-1040 Vienna, Austria
{*lsuper, onemeth, mrupp*}*@nt.tuwien.ac.at*

Received (received date)
Revised (revised date)

In this work we investigate the possibility of detecting errors in H.264/AVC encoded video streams. We propose a method for the detection of errors exploiting the set of entropy coded words as well as range and significance of the H.264/AVC information elements. We evaluate the performance of such syntax analysis based error detection technique for different bit error probabilities and compare it to the typical packet discard approach. Particular focus is given on low rate video sequences.

Keywords: H.264/AVC, Syntax analysis, Error resilience
Communicated by: to be filled by the Editorial

1 Introduction

H.264/AVC (Advanced Video Coding) [1] is the recent video coding standard, defined by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) as the product of a collective partnership effort known as the Joint Video Team (JVT). This standard is especially suitable for low data rate applications as it provides substantially better video quality at the same data rates compared to previous standards (MPEG-2, MPEG-4, H.263), with only a moderate increase of the complexity. Moreover, H.264/AVC was designed to support a wide variety of applications and to operate over several types of networks and systems.

Video telephony and video streaming over Internet Protocol (IP) packet networks are quite challenging applications due to their requirements on delay and data rates. A video stream is encoded and encapsulated in Real Time Protocol (RTP) packets. These packets are tipically transported end-to-end within the User Datagram Protocol (UDP). Unlike the Transmission Control Protocol (TCP), UDP does not provide any retransmission control mechanisms. Nevertheless, it has been widely adopted for video streaming and video telephony, since end-to-end retransmissions would cause unacceptable delays. Thus, in such real-time applications, transmission errors cannot be completely avoided.

To allow for applications even in error-prone environments like mobile networks, apart from the improved compression performance, H.264/AVC provides several error resilience features. Therefore, the 3rd Generation Partnership Project (3GPP), standardizing the Universal Mobile Telecommunications Network (UMTS), has approved the inclusion of H.264/AVC as an

optional feature in Release 6 of its mobile multimedia telephony and streaming services specifications ([3], [2]).
To facilitate error detection at the receiving entity, each UDP datagram is provided with a simple 16 bits long checksum. The packets with detected errors are typically discarded [2, 4] and missing parts of the video are subsequently concealed. The reason for this handling is the variable length coding (VLC). The H.264/AVC standard supports a context adaptive VLC (CAVLC) in all its profiles. After a bit error, (CA)VLC may easily desynchronize, making the correct distinction between the following codewords impossible. Therefore, without any resynchronization mechanism and/or additional detection/decoding mechanism (e.g. [5], [6], [7]), the decoding of such stream may result in considerable visual impairments, or may become even impossible (due to the non-existing codewords, too many or too few bits left for decoding). The detection of errors allows to utilize the correctly received parts of the packet for the decoding. Since a packet usually contains a rather large picture area, it may considerably improve the quality of reconstruction at the receiver. The structure of the bit stream — the syntax of its information elements — may also provide some means to detect errors. For H.263 codecs, the performance of a simple syntax check method was evaluated in [8]. However, the structure of the H.264/AVC bitstream and the CAVLC differs considerably from the structure and VLC of the H.263 bitstream.

In this paper we investigate the possibility of detecting errors in H.264/AVC encoded video stream. We propose a method for error detection exploiting the codewords, as well as range and significance of the H.264/AVC information elements. We evaluate its performance and compare it to the typical packet discarding approach. The focus of this work is given on the baseline profile (targeting video conferencing, streaming and especially mobile applications) and thus, we work with CAVLC rather than with context adaptive binary arithmetic coding (CABAC), mainly designed for the storage applications. We do not take into account error detection within the RTP/UDP/IP header. Errors within the header could also be detected by other means, e.g. UDP-lite [9], or using the information from lower layers depending on the underlying system.
This paper is organized as follows. Section 2 introduces briefly the architecture of the H.264/AVC codec. The structure of the H.264/AVC RTP bitstream is described in Section 3. Section 4 analyzes individual information elements and presents the way in which their syntax may be used to detect errors. Results of the performance evaluation and comparison with alternative methods are provided in Section 5. Section 6 contains conclusions and some final remarks.

2 H.264/AVC Design Characteristics

H.264/AVC defines a hybrid block-based video codec. Despite of a significant increase of performance, compared to its predecessors of both the ISO and the ITU-T family, there is no single element of the coding process granting the majority of the improvement; the enhancements are rather gained using a plurality of single improved features.

Depending on the applications, H.264/AVC defines seven *profiles*: baseline (conversational services), main (broadcast and storage application), extended (internet streaming) and four high profiles (broadcast for High Definition TeleVision — HDTV). The 3GPP specification for transparent end to end packet switched streaming service [2] as well as the ITU-T specification

for 3G terminals (H.324) [10, 11] suggest the client to be compatible with the H.264/AVC baseline profile. Therefore, in the following, we will refer to the set of features supported by the baseline profile.

Similarly to its precursors, the H.264/AVC encoding process is characterized by a hierarchical structure. The video sequence consists of the succession of still pictures called *frames*. Each frame is segmented into *macroblocks* (MB) of 16×16 pixels. A macroblock could be further subdivided in smaller blocks up to 4×4 pixels.

H.264/AVC allows two frame encoding strategies: *intra* and *inter*. Intra (I) frames are encoded exploiting spatial correlation. The already encoded neighboring macroblocks can be used as a reference to predict the macroblock to be encoded. Inter predicted (P) frames exploit the temporal correlation by referencing the MBs from previous frames (*motion compensation*) contained in a buffer (*reference list*).

For I and P coding, given a macroblock to be encoded, the algorithm looks for its best prediction, in time and in space, respectively. This predicted block is then subtracted from the original one, obtaining the *residual* block. The residual block is transformed by means of a modified discrete cosine transformation (DCT) [12], and quantized using a certain *quantization parameter* (QP), obtaining the block of quantized coefficients called *levels*. The levels are then zig-zag scanned and entropy encoded.

I frames are used to refresh the sequence. They enable random access and, in case of error prone transmission channels, limit the propagation of errors in time. The set of frames from an I frame up to the P frame preceding the next I frame is defined as *group of picture* (GOP). Inter-coding requires much less associated information elements than intra-coding to encode a frame. Therefore, for a given sequence, the resulting data rate depends strongly on the GOP size and on the quantization parameter.

H.264/AVC is conceptually separated into a VCL (Video Coding Layer) and a NAL (Network Abstraction Layer), as depicted in Fig. 1.

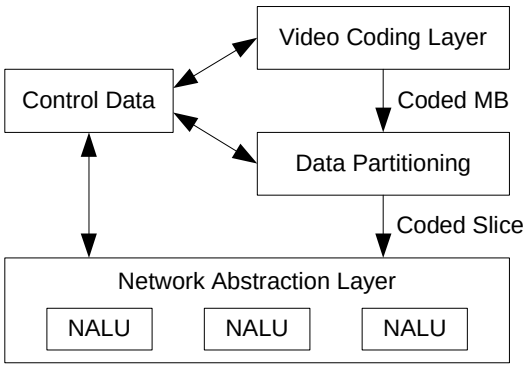


Fig. 1. H.264/AVC conceptual layers

The VCL is responsible for the core block-based hybrid coding functions, the NAL provides network friendliness by allowing the output bitstream to be transmitted over different transport layer protocols. The encoded video data produced by the VCL is segmented by the NAL in a stream of information units called NALU (Network Abstraction Layer Unit).

There are two types of NAL units: non-VCL and VCL NALUs. Non-VCL NALUs contain

information associated to the sequence characteristics. To this category belong Sequence Parameter Set (SPS) — defining profile, resolution and other information associated to the whole sequence — and Picture Parameter Set (PPS) — containing type of entropy coding, slice group and quantization properties. VCL NALUs contain the data associated to the video slice, each VLC NALU refers to a non-VCL NALU as shown in Fig. 2.

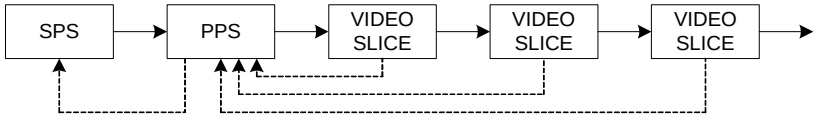


Fig. 2. NAL unit sequence

The NAL functionalities are responsible for the arrangement of encoded MBs into a NALU. Given a frame to be encoded, the NAL segments the frame in groups of MBs, called *video slices*. The number of MBs contained in a slice depends on the proper encoding settings.

In a packet oriented environment, each packet contains one NALU. Fig. 3 shows the encapsulation hierarchy of a video slice for a transmission over IP.

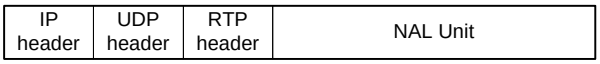


Fig. 3. NAL Unit encapsulation

Since the packets have fixed size, each NALU should fit the appropriate protocol packet payload. Therefore, the number of MBs stored in a video slice depends on frame content and on the encoding strategy. The MBs contained in a VCL NALU cannot contain references to macroblocks of the same frame but belonging to a different slice. This is intended to limit the impact of damaged or even not received packets on the quality of the decoded stream.

In this paper, we propose a syntax analysis performed on VCL NALUs, assuming that non-VCL NALUs are not transmitted within the RTP payload, but provided in the SDP (Session Description Protocol) ([2], [3]).

3 H.264/AVC VCL NALUs Bitstream Structure

This section offers a brief overview of the H.264/AVC syntax, as produced by the JM (Joint Model) reference software [13] ver. 10.2 in baseline profile and described in [1].

The standard [1] defines different binarization strategies for each of the information elements. Besides fixed length codes (FLC), several variable length coding strategies are used. The residuals are encoded in the baseline profile by means of CAVLC. A variable length coding reduces the entropy of the source by assigning shorter codewords to the most frequent symbols. This allows on one hand bit-rate saving but, on the other hand, it results to be sensitive to decoding desynchronization.

Fig. 3 shows the structure of the considered VCL NALU payload, composed by a Slice Header (SH) and the encoded macroblocks.

The slice header contains the basic characteristics of the slice, such as the slice index, the frame index and the macroblock prediction type. Since the information contained within the slice header determines the decoding of the contained macroblocks, an error in SH can make the entire slice undecodable.

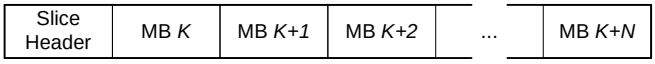


Fig. 4. Structure of a VCL NAL unit

After the slice header, the VCL-NALU contains the encoded macroblocks belonging to the slice. The VLC decoding is restarted at the beginning of each NALU since there is no resynchronization point within a NALU. The encoded information elements associated to the macroblock depend on the prediction type (inter or intra).

Without loss of information, we decided to subdivide the entropy coding strategies in the following four groups.

- Exp-Golomb coded codewords — *EG*: The exponential Golomb code [14] (or simply exp-Golomb code) is a parametric (k) universal code. H.264/AVC uses a special type of the exp-Golomb codes, the parameter k is set to 0, also known as Elias-γ encoding [15]. Exp-Golomb encoded words are characterized by a regular logical structure consisting of a predetermined code pattern without requirement of decoding tables. Each exp-Golomb codeword embodies the following elements:

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

  The first M zeros and the middle one are regarded as *prefix* while the following M bits represent the *info* field. In the prefix is unary encoded the value of the length M. The exp-Golomb field codeNum is obtained as

$$\text{codeNum} = 2^M + \text{info} - 1.$$

  The encoded value is derived from the codeNum depending on the chosen exp-Golomb coding style (unsigned, signed and truncated). An error affecting the leading zeros or the middle 'one' affects the decoding by modifying the value of M, therefore causing the misinterpretation of the codeword boundaries. desynchronization of the decoding process. An error in the *info* field causes deviation of the decoded parameter value and may affect the following elements, but does not cause desynchronization directly.

- CAVLC level codewords — *VL*: The context adaptive variable length coding style is characteristic for encoding the levels. The levels are zig-zag ordered from the highest to the lowest frequency, then they are encoded using a VLC-N procedure, where N is a parameter depending on the value of the previously encoded levels. The standard defines integer values of N in the range [0,6]. The first level of each macroblock is encoded using the VLC-0, the resulting codeword has the following structure:

$$\underbrace{0_1 \dots 0_M}_{M} 1,$$

  where the parameter M embodies both absolute value and sign of the level.

For increasing encoded residual values, the procedure is adapted in order to assign shorter codewords to the predicted level values. A codeword encoded with a VLC-N ($N > 0$) procedure has the form:

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

Similarly to exp-Golomb codes, the codeword starts with a sequence of M leading zeros followed by a one, an *info* field consisting of *N-1* bits and one explicit sign bit s. The encoded value is then obtained as

$$(-1)^s \cdot ((M+1) << (N-1) + 1 + \text{info}),$$

where $<<$ represents the left bit-wise shift operation.
The 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, errors lying in the *info* field or sign do not cause desynchronization but affect only the decoded level. Errors in the *info* field may also cause the use of a false VLC procedure for the next decoded items.

- Tabled codewords — *TE*: This category includes the VLC words to be found in a look-up table. H.264/AVC defines several VLC tables for different syntax elements (e.g. zero runs) and contexts (e.g. number of remaining zeros to be encoded). Errors may result in both deviation of the decoded value and decoding desynchronization.

4 Proposed Mechanism for Error Handling

In this section we discuss the effects of errors during the decoding of H.264/AVC encoded datastreams.

The consequences of an error are twofold. We observe *direct* effects, since errors cause the misinterpretation of the encoded parameter value and, therefore, affect the reconstruction of the considered macroblock. At the same time, bitstreams containing VLC-encoded informations are prone to *desynchronization*. Due to errors, the decoder segments the stream in an improper way, causing the following codewords to be misinterpreted aswell. The error affecting a parameter in a given macroblock is therefore not necessarily limited to the affected MB but can propagate until the end of the slice degrading a wide frame area (*spatial error propagation*).

The spatial propagation is particularly critical in I frames. Even if the code associated to a certain macroblock does not contain errors, each macroblock is usually spatially predicted, referencing to the surrounding macroblocks. If one of the referencing MBs is uncorrectly decoded, then the macroblock referencing to it will be flawed as well. In P frames the spatial error propagation is caused by decoding desynchronization and by wrong motion prediction.

Besides spatial propagation, in case of errors we also experience temporal propagation. In P frames each macroblock is predicted by means of motion compensation from a macroblock belonging to a previous frame. If the reference picture area is distorted, the reconstructed macroblock will result in distortion aswell. Such effect is called *temporal error propagation*. Errors in I frame can propagate all over the GOP.

In order to limit the error propagation in time and space, we propose a syntax check mechanism capable of spotting errors during the decoding of the stream. The syntax check limits in H.263, discussed in [8], are in H.264/AVC even more evident. H.264/AVC makes extensive use of variable length coding and, additionally, differs from H.263 since it does not provide *synchronization words* between group of blocks (GOB).

In the following we will discuss our proposed error detection mechanism and its application to the encoded information elements of I and P frames as well as the slice header.

4.1 Syntax Check Rules

In order to suite the structure of the JM reference software we subdivided the macroblocks decoding process into two main blocks, as depicted in Fig. 5. During the *READ* phase, the raw bitstream is read and partitioned in codewords. During the *DECODE* phase these codewords are interpreted as information elements (IE) and used to reconstruct the slice.

Since the length of the codewords is inferred by the bistream structure and by the expected parameter, possible desyncronizations occur during the READ phase.

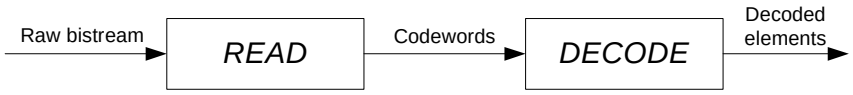


Fig. 5. Conceptual scheme of parameter decoding

We subdivide the possible decoding errors into three main categories, depending on their characteristics:

- **Illegal Codeword — IC**: Arises when the codeword does not find correspondence in the appropriate look-up table. IC occurs during the *READ* process for tabelled codewords.
- **Out of Range Codeword — OR**: Results when the decoded value lies outside the legal range. It appears during *READ* process for all types of codewords. If the decoded parameter can only take values between [-K,K], an error is produced if the absolute value of the read parameter is greater than K.
- **Contextual Error — CE**: Occurs when the decoded word leads the decoder to illegal actions. It arises during the *DECODE* phase for all types of encoded parameters.

The presented errors are not strictly related to current bitstream failures. They are rather referred to the detectable anomalies, possibly caused by propagation of previously undetected errors.

The three classes of errors are detected by our decoder implementation. Our error detection mechanism cannot lead to false detections. However, it may happen that an error is detected after its true occurrence.

4.2 Parameters Analysis

After describing the encoding techniques and detectable error categories, the characteristics of the elements contained in VCL NALUs will be introduced.

The parameters encoded in I slices, P slices and slice headers are presented as generated in the encoding trace file of the JM 10.2. Apart from the slightly different nomenclature, they correspond to the structure described in [1].

For each of the considered fields, the encoding style and the possible error category are outlined. Additionally, where useful, a brief investigation of the error characteristics is provided.

I Frames

Parameter Name	Enc.	Err.
`mb_type`	*EG*	***OR***
`intra4x4_pred_mode`	*TE*	***CE***

Since the spatial prediction uses reference to the surrounding macroblocks, if they are not available, not yet decoded or belonging to another slice, a contextual error is produced.

`intra_chroma_pred_mode`	*EG*	***OR***
`coded_block_pattern`	*EG*	***OR***
`mb_qp_delta`	*EG*	***OR***
`Luma(Chroma) # c & tr.1s`	*TE*	***IC***

The look-up table used to decode this value is not complete. The decoded codeword could not find reference to any legal value.

`Luma(Chroma) trailing ones sign`	*EG*	

The signs of the trailing ones are fixed length encoded and do not influence any of the following parameters. By means of syntax check it is not possible to detect such errors.

`Luma(Chroma) lev`	*VL*	***OR/CE***

Decoded macroblock pixels can only take values lying in the range [0,255]. During *READ* phase, values outside the bounds are immediately associated to errors. During *DECODE* phase, the residuals are added to the predicted values and the contextual check is performed. An extended range $[-\lambda, 255 + \lambda]$ is considered due to possible quantization offset. The value of λ depends on the quantization parameter used

`Luma(Chroma) totalrun`	*TE*	***IC***
`Luma(Chroma) run`	*TE*	***IC/OR***

Depending on the number of remaining zeros, a VLC look-up table is chosen. For more than six remaining zeros a single table covering the zero run range [0,14] is used. Therefore, the decoder is exposed to out of range errors.

P Frame

Many of the parameters encoded in P frames are equivalent to those used to describe an I frame. In the following only the parameters exclusive for P frames are discussed.

`mb_skip_run`	*EG*	***OR/CE***

The number of skipped macroblocks cannot be greater than the total number of MBs in frame minus the number of the already decoded MBs.

`sub_mb_type`	*EG*	***OR***
`ref_idx_l0`	*EG*	***OR/CE***

The index of the reference frame cannot be greater than the actual reference buffer size.

`mvd_l0` *EG* ***CE***

Slice Header

`first_mb_in_slice` *EG* ***OR***

`pic_parameter_set_id` *EG* ***OR/CE***

The VCL-NALU cannot refere to a PPS index greater than the number of available PPSs.

`slice_type` *EG* ***OR***

`frame_num` *EG* ***OR***

Depending on the GOP structure an out of range error can be detected

`pic_order_cnt_lsb` *EG* ***OR***

`slice_qp_delta` *EG* ***OR***

4.3 Error Handling

In this work we test two standard error handling strategies and compared them with our syntax check based method. Since the focus of this work was given on the comparison of error handling strategies and detection performance rather than on the concealment results, detected errors are concealed by means of a zero motion temporal error concealment. It simply replaces each corrupted macroblock in the current frame $\mathrm{MB}_f(i,j)$ with the spatially corresponding one in the previous frame $\mathrm{MB}_{f-1}(i,j)$.

4.3.1 Straight Decoding — SD

The straight decoding represents the plain decoding strategy where the errors are not detected and, therefore, not concealed. Since the reference software [13] cannot handle the error categories described in Section 4.1, we modified the JM letting damaged bitstream to be decoded. Each erroneous parameter value is replaced with the most similar legal one (for out of range and illegal codewords) or to the safest one (for contextual error).

Fig. 8(a) shows a corrupted frame decoded using the error handling strategy *SD*. The different slices are separated by means of the semitransparent dark lines. An error is inserted into macroblock 32 (red square). Since no concealment routine is called, the rest of the slice is decoded using desynchronized VLC codewords, causing wide artifacts until the end of the slice.

4.3.2 Slice Level Concealment — SLC

This strategy relies on the checksum information provided by the lower layer protocols. For wireless transmission the UDP protocol is used. The checksum information is calculated over the entire NALU and its RTP header. Each error, regardless of its position and effect on the decoding, results in the slice rejection and concealment. A block diagram of this approach is depicted in Fig. 6

The same error considered in the previous handling strategy, produces a concealed frame as shown in Fig. 8(b).

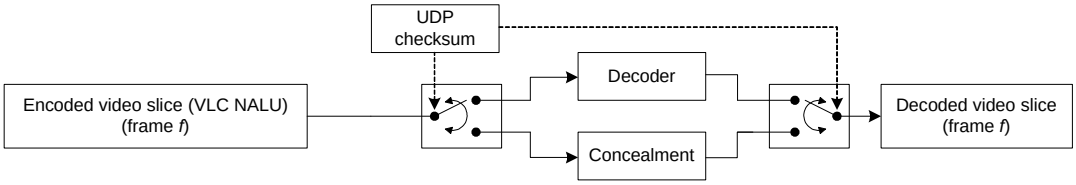


Fig. 6. Block scheme of the Slice Level Concealment handling

4.3.3 Macroblock Level Concealment — MBLC

Macroblock level concealment represents our proposal for an efficient error handling strategy.

The first presented strategy, SD, is not aware of errors and, therefore, an error propagates spatially until the end of the slice. On the other hand, the SLC approach appears to be exceedingly coarse. As shown in the previous discussion, some errors do not influence the decoding process radically. Moreover, the slice rejection mechanism causes the discarding of the error-free macroblocks preceding the error. These macroblocks can be correctly decoded. Therefore, in contrast to the slice level concealment, our error handling mechanism is performed at macroblock level.

The proposed approach consists of the implementation of the presented error detection mechanism based on the syntax analysis. We detect errors belonging to the classes defined in Section 4.1. Once, during the decoding, one of these errors arises, the affected macroblock and all the following (until the end of the slice) are concealed. This mechanism is depicted in Fig. 7.

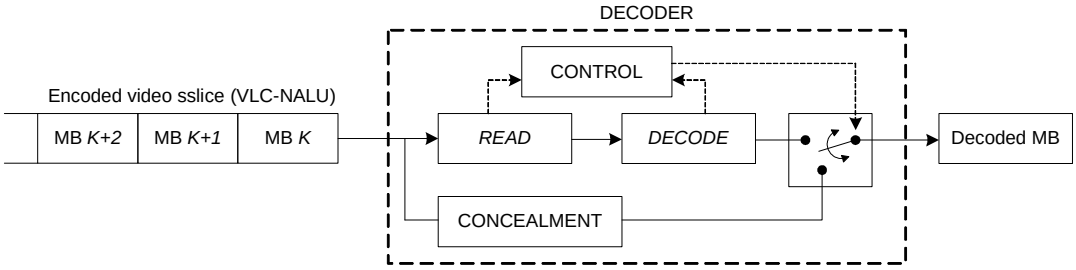


Fig. 7. Block scheme of the Macroblock Level Concealment handling

Fig. 8(c) displays the result of the MBLC applied to the frame affected by an error in MB 32 like before. Using MBLC, the error is detected in macroblock 33 (green square), the macroblocks from 33 up to the end of slice (MB 38) are concealed.

Further analysis shows that conceptual errors can arise also at the slice level. Due to desynchronization, the code length can be too small or too large. If the bitstream is too short to decode the whole slice, a concealment method is called for the remaining macroblocks. On the other hand, the number of decoded macroblocks can exceed the number of macroblocks originally belonging to the slice. The slice header contains a parameter named *first_mb_in_slice*, indicating the index of the first macroblock contained within the slice. Under the assumption of correctness of the slice header, if the decoded value is lower than the number of previously

decoded macroblocks, the exceeding macroblocks are overwritten with the correct MB.

(a) SD (b) SLC (c) MBLC

Fig. 8. Corrupted frame decoding

5 Performance Analysis

To evaluate the performance of the three presented error handling methods, we performed experiments with corrupted H.264/AVC encoded video sequences.

For our simulations we used the encoder and decoder of Joint Model H.264/AVC v.10.2 [13] adapted to our needs by introducing the following additions:

- The decoder is able to read external text files containing error patterns and to modify at NAL level the bits to be corrupted.
- If a read codeword or a decoded parameter assumes an illegal value, its value is restored to the most similar legal one (SD). Additionally, two error flags, one at macroblock level and one at slice level, are forced to one (and used for error handling at macroblock level).

The sequence used for the simulations is "Foreman" in QCIF (176×144 pixels) resolution. The total length of the sequence is 400 frames, played at 30 frames per second. The sequence was encoded in baseline profile, the selected GOP size was 10. The slicing mode was chosen fixing the maximum number of bytes per slice to 700 bytes. Flexible macroblock ordering, rate control and rate-distortion optimization were not used. The number of reference frames for motion estimation was set to five. In order to analyze the behavior of the error handling mechanism as a function of the compression rate, the sequences were encoded with different quantization parameters, namely 20, 22, 24, 26, 28, and 30.

For all quantization parameters, we considered various bit error rates (BER) in the range from 10^{-2} to 10^{-7}. We generated 75 random error patterns for each BER.

We performed two kinds of analysis. The first concerns the resulting end-to-end quality. The second investigation is performed over the detection capabilities of our proposed method.

5.1 End-to-End Quality Results

We simulated transmission over a binary symmetrical channel by inserting errors in the positions indicated by the error patterns. The degraded streams were decoded using the three approaches described in Section 4.3.

The quality was expressed as the peak signal-to-noise ratio of the luminance component (Y-PSNR). Given the luminance component $\mathbf{DY}_f$ of a degraded sequence (at frame f) and

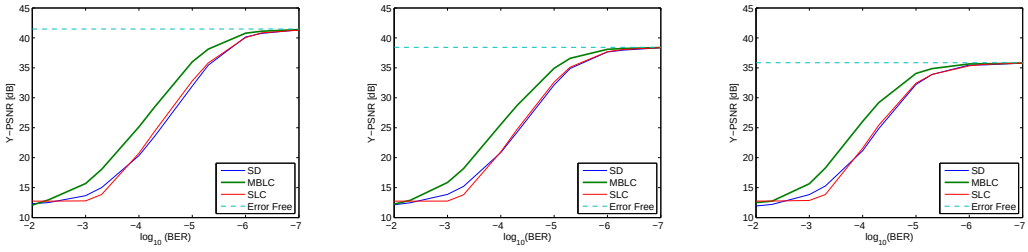
the luminance component $\mathbf{OY}_f$ of the original non compressed (non degraded) sequence, then we calculate the mean square error (MSE) and Y-PSNR as

$$\mathrm{MSE}(f) = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} [\mathbf{OY}_f(i,j) - \mathbf{DY}_f(i,j)]^2, \tag{1}$$

$$\text{Y-PSNR}(f) = 10 \cdot \log_{10} \frac{255^2}{\mathrm{MSE}(f)}, \tag{2}$$

The resolution of the frame is $N \times M$, indexes i and j address particular luminance values within the frames. For a given BER, the quality in Y-PSNR was averaged over the 400 frames and over the 75 decoded sequences.

The comparison is performed over the quality performance of the three approaches together with the error-free decoded sequence, used as reference. The results of the simulations are plotted in Fig. 9.



(a) QP = 20 (b) QP = 24 (c) QP = 28

Fig. 9. Performance of the different error handling strategies

The results confirm the assumption made in Section 4.3. The proposed approach clearly outperforms both classical approaches, with quality improvement up to 4 dB in the middle of the considered BER range. For high BER ($[10^{-2}, 10^{-4}]$) the slice rejection mechanism performs even worse than the straight decoding, since, statistically, one error is inserted in each slice. For decreasing BER, SLC slightly outperforms the SD.

Remaining undetected errors are not concealed and, therefore, result in artifacts. These, however, remain local if the error is detected in some successive information elements. A graphical interpretation of the results, referred to the scenario described in Section 4.3, is provided in the following.

Using the proposed approach, the decoding of a slice can be described by subdividing it into three intervals as shown in Fig. 10:

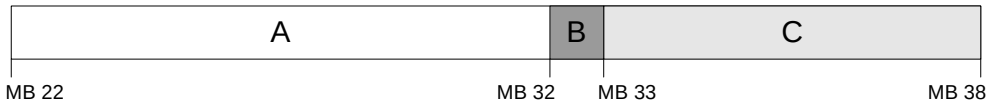


Fig. 10. Origination of macroblock level concealment delay

The figure considers only the slice affected by the error. The characteristics of each interval are described in the following.

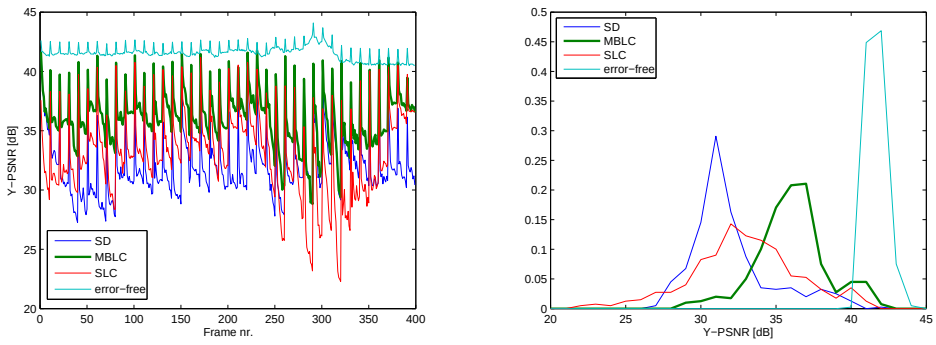
- ***Interval A :*** MB [22,32), the slice is correctly decoded from its beginning (MB 22) up to the error appearance (MB 32). These macroblocks were not affected by errors and are thus correctly decoded.

- ***Interval B :*** MB [32,33), the error (appeared in MB 32) remains undetected until MB 33. These macroblocks were decoded incorrectly and result in artifacts in the decoded frame.

- ***Interval C :*** MB [33,38], the macroblocks starting from MB 33, where the error was detected, until the end of the slice (MB 38) are concealed.

The efficiency of the proposed approach in comparison to SLC and SD, lies in the interval [22,32) and [33,38], respectively. SD, by discarding the whole slice, does not exploit the macroblocks that were not affected by errors. This fraction of the slice can be significant. For the considered error concealment method, the improvement is even higher in the sequences with faster motion, resp. in the sequences with reduced frame rate, since in such cases the performance of error concealment decreases. The macroblocks decoded in the interval B cause errors with higher magnitude. However, assuming a small delay between error appearance and error detection (cf. Section 5.2), the effect of the artifacts is limited. Using the proposed method, only the macroblocks following the error detection are concealed. This clearly prevails the straight decoding, where the macroblocks in the interval C are uncorrectly decoded, resulting in the wide artifact shown in Fig. 8(a).

As final results, the average Y-PSNR over time (frame number) is shown in Fig. 11(a) for the three investigated methods compared to the error-free decoded sequence. Averaging was performed over sequences encoded with QP 28 and decoded with BER of 10^{-5}. For the same BER, Fig. 11(b) shows the histogram of the decoded frame quality for the considered method. In both graphs the improvement brought by the utilization of the proposed method can be observed.



(a) Y-PSNR over time (b) Empirical PDF

Fig. 11. Y-PSNR over frame number for SD, SLC, and MBLC.

The frames in the range [250,350] of the Foreman sequence are characterized by the fast camera movement. In Fig. 11(a) we can distinguish the degradation of SLC performance in this interval.

5.2 Detection performance

Furthermore, we investigated the detection capabilities of the proposed method. The tests were performed choosing a quantization parameter of 28 and inserting one error per slice during the decoding.

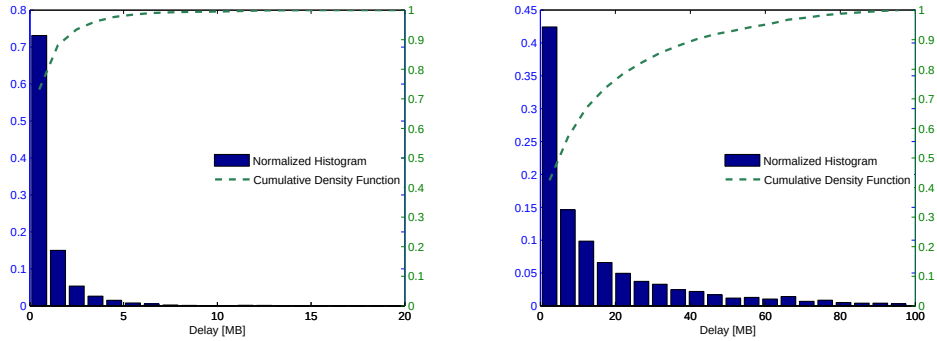
The performance was tested separately for I and P frames. The I frames syntax presents significant differences to the P frames.

The first main difference is the size of the code associated to a frame. Considering a quantization parameter of 28 and slice dimension of 700 bytes, the majority of the P frames are contained within one NALU, whereas an I frame usually consists of more than 4 slices. This reflects on different effects of errors propagation and concealment, since in P frames the end of the slice usually corresponds to the end of the frame. This also explains why the SLC approach performs poorly for P frames.

Additionally, the parameters encoded in I frames are usually more sensitive to errors, improving the performance of our detection mechanism. Bit errors in P frames, the parameters of which consist mainly of motion vectors, yield less frequently to desynchronization.

For both frame types we first calculate the detection probability. More than 60% of the errors inserted in I frame are detected, for P frames the percentage is 47 %. The following discussion will help the understanding of these values.

For the detected errors we calculate the detection delay, i.e. the distance, expressed in number of macroblocks between the error appearance and the error detection. Fig. 12 shows the histogram for I and P frames.



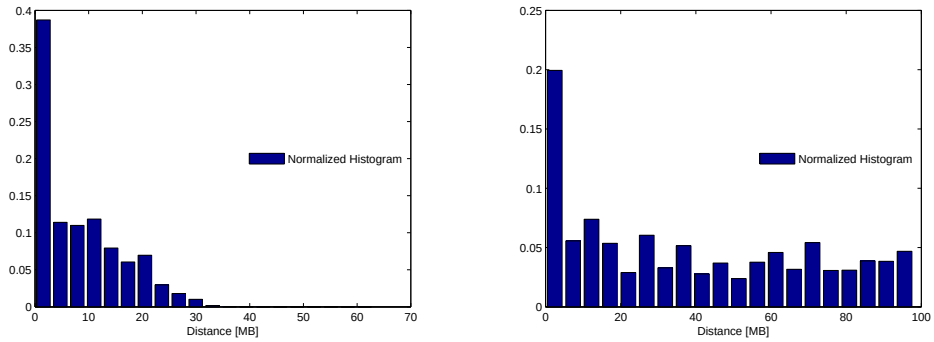
(a) I frames (b) P frames

Fig. 12. Detected errors: detection distance

As expected, the range of detection distance for I frames is much smaller than the one for P frames. For I frames we obtained excellent results. The average detection delay is 1.39 MB and over 85% of the errors are detected within 2 MBs. The interval B in Fig. 7, where the macroblocks are incorrectly decoded, is therefore extremely narrow. For P frames, the average detection delay is bigger: 15 MBs.

For the undetected errors we performed a different analysis. We measured the distance, expressed in macroblocks, between the error appearance and the end of the slice. Besides *undetectable errors*, errors that cannot be detected at all by our detection approach, we assumed that missed error detections also occur if the errors arise near to the end of the slice

and, therefore, the decoder reaches the end of the slice before the error could be detected. Fig. 13 shows the results of these measurements.



(a) I frames (b) P frames

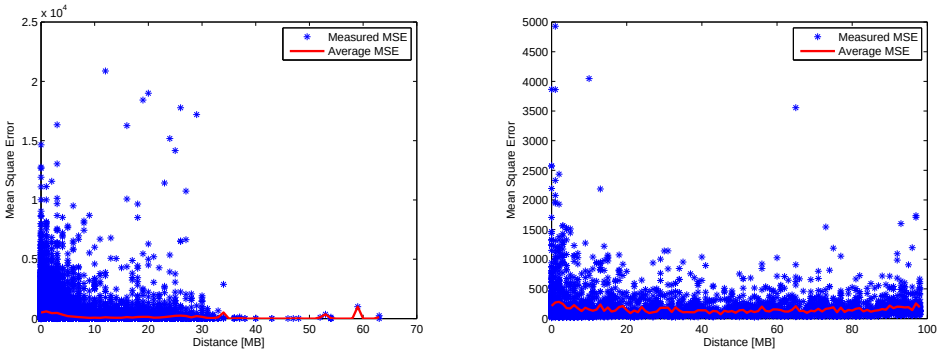
Fig. 13. Undetected errors: distance between error appearance and end of slice

The obtained results appear to be fully compatible with our assumption. For both I and P frames we observe peaks of the histogram for distances smaller than the average detection delays. These missed detections can be reasonably attributed to the errors arising near the end of the slice. In the remaining range the histogram is uniformly distributed, representing the undetectable errors. For I frame we observe the decreasing of the histogram for increasing distance. This effect could be justified considering that in the simulations scenario an I frame (99 MBs) consists, in average, of five slices. Bigger I slices occur rarely, therefore the number of occurrence of I slices decrease with increasing slice dimension (in MB), as well as the number of undetected errors.

We conclude the performance analysis by examining the effects of undetected errors. Since undetected errors result in artifacts, we observed their impact on the quality measured in terms of MSE of such missed detection. The mean square error is calculated with respect to the same frame index belonging to a error-free decoded stream. In Fig. 14 we plotted the resulting MSE as a function of the distance between the error appearance and the end of the slice.

The obtained results are significant. The graphs does not show any direct proportionality between the distance and the resulting MSE. For increasing distance between the error occurrence and the end of the slice the resulting mean square error remains roughly constant. This effect is clearly notable for P frames (Fig. 14(b)). For I frames we explain the decreasing MSE density for increasing distance as for the behavior in Fig. 13(a). We can also measure the highest MSE peaks for undetected errors appearing near the end of the slice. These are caused by errors occurring near the end of the slice and, therefore, not detectable because of the detection distance as discussed previously.

We can therefore conclude that the errors that cannot be detected by the proposed algorithm are the one that does not cause decoding desynchronization. Moreover, we can infer that they do not lead to significant quality degradation.



(a) I frames (b) P frames

Fig. 14. Undetected errors: resulting MSE

6 Conclusions

Three possible error handling strategies are compared in this paper: common packet discard approach, strategy based on straight decoding of H.264/AVC and the proposed syntax analysis based error detection. The proposed syntax analysis based detection method performs better than the packet discard strategy. Since the syntax analysis does not require much complexity, we can conclude that it is beneficial to implement as an alternative to the widely adopted packet discard method.

Acknowledgments

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.

1. ITU-T Recommendation H.264 and ISO/IEC 11496-10 (MPEG-4), "AVC: Advanced Video Coding for Generic Audiovisual Services", version 3, 2005.
2. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects,"Transparent end-to-end Packet-switched Streaming Service (PSS); Protocol and Codecs," 3GPP TSG TS 26.234, ver. 5.7.0, available in `http://www.3gpp.org/.`
3. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "Packet switched conversational multimedia applications; Default codecs (Release 6)," ver. 6.4.0, available in `http://www.3gpp.org/.`
4. S. Wenger, "H.264/AVC Over IP," IEEE Transactions On Circuits And Systems for Video Technology, 13, no. 7, pp. 645–656, Jul. 2003.
5. O. Nemethova, J. Canadas, M. Rupp, "Improved Detection for H.264 Encoded Video Sequences over Mobile Networks," Proc. of Int. Symp. on Com. Theory and Appl., Ambleside, UK, Jul. 2005.
6. M. Chen, Y. He, R.L. Lagendijk, "A Fragile Watermark Error Detection Scheme for Wireless Video Communications," IEEE Trans. on Multimedia, vol. 7, no. 2, pp. 201-211, Apr. 2005.
7. C. Weidmann, O. Nemethova, "Improved Sequential Decoding of H.264 Video with VLC Resynchronization," in Proc. of IST Mobile Summit 2006, Myconos, Greece, Jun. 2006.
8. M. Barni, F. Bartolini, and P. Bianco, "On the performance of syntax-based error detection in h.263 video coding: A quantitative analysis," Image and Video Com., Proc. of SPIE, vol. 3974, pp. 949–956, Jan 2000.
9. IETF RFC 3828 "The Lightweight User Datagram Protocol (UDP-Lite)," Jul. 2004.

10. D. Lindberg, "The H.324 multimedia communication standard," IEEE Communication Magazine, vol. 34, pp. 46–51, Dec. 1996.
11. ITU-T Rec. H.324, "Terminal for low bit-rate multimedia communication," Feb. 1998
12. Joint Video Team Document JVT-C028, G. Bjontegaard and K. Lillevold, "Context-adaptive VLC (CVLC) coding of coefficients," Fairfax, VA, May 2002.
13. H.264/AVC Software Coordination, "Joint Model Software," ver. 10.2, available in `http://iphome.hhi.de/suehring/tml/`.
14. S. W. Golomb, "Run-length encodings," IEEE Transaction on Information Theory, vol. 12, pp. 399–401, Jul 1996.
15. P. Elias, "Universal Codeword Sets and Representations of the Integers," IEEE Transaction on Information Theory, vol. 21, no. 2, Mar. 1975.