Improved Detection for H.264 Encoded Video Sequences over Mobile Networks

Olivia Nemethova*, Jacob Canadas Rodriguez** and Markus Rupp*

*TU Wien, Institute of Communications and RF Engineering
Gusshausstrasse 25/389, 1040 Wien, Austria
{onemeth, mrupp}@nt.tuwien.ac.at
**Universitat Politecnica Catalunya, Technical School of Castelldefels (EPSC)
Avinguda del Canal Olimpic, s/n, 08860 Casteldefels, Spain
jacob.canadas@estudiant.upc.es

Abstract

Real-time video conferencing or streaming over mobile network introduces a challenge to the end-to-end design of the whole system. Due to the real-time character of these applications, there are no retransmissions possible at the transport layer and therefore any errors result in packet loss at the receiver. This article investigates error resilience methods, especially improvements of the detection granularity to minimize perceptual visual quality degradation at the receiver. Costs of the additional error resilience is evaluated and a combination of H.264 resilience features, improved error detection and error concealment is finally tested.

1 Introduction

Emerging 3rd generation of mobile communication systems brought new multimedia services as video conferencing and streaming. These services are delay sensitive and therefore they are usually transported via the unreliable User Datagram Protocol (UDP) rather than via the Transmission Control Protocol (TCP), the latter providing the possibility of transport layer retransmissions. UDP usage leads to possible packet losses at the receiver, further degrading the end-user quality. To match the screen of common mobile terminals, a resolution QCIF (144×176 pixels) is used. The loss of a UDP packet for such small resolutions may result in a loss of significant part of the picture. H.264 [1] is the newest high compression digital video codec standard jointly written 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 also known as AVC, for Advanced Video Coding. This codec performs temporal and spatial prediction to achieve high compression gain. The usage of prediction results in a temporal and spatial propagation of errors within the video stream. The final visual perceptual distortion depends also on the used error resilience methods. Error resilience methods can be performed either at the sender (i.e. the way of encoding, packetization) or at the receiver (error concealment).

In case of video transport over wireless networks, the receiver can be a power and size limited mobile terminal; the typical capacity of a Release 99 Universal Mobile Telecommunications System (UMTS) cell is 2 Mbit/s. These are the limits for complexity and overhead that can be spent to use additional error resilience methods.

The intention of this article is to propose a hybrid of existing error resilience methods and additional new detection methods for H.264 video over mobile networks, so that the perceptual visual quality degradation is reduced as far as possible while keeping the costs reasonably low. In Section 2 the character of the errors in the H.264 is described. Section 3 discusses error resilience features defined in H.264, while Section 4 introduces some additional error resilience methods designed for H.264 video over wireless networks and analyzes their costs. In Section 5 the results are presented and interpreted. Section 6 contains conclusions and some final remarks.

2 Errors

For a transmission over a mobile network, raw video stream is sub-sampled, segmented into slices, compressed, and multiplexed into the Real Time Protocol (RTP) packets that are further encapsulated into UDP packets. UDP packets contain two Cyclic Redundancy Check (CRC) bytes allowing detection of errors within the packet.

Even if there is a single bit error only, the CRC will fail and the whole packet is usually discarded. Such lost packet may result in a loss of significant part of the picture content. The decoder will furthermore fail to decode all other frames using the erroneous one as a reference for temporal prediction - the error will propagate in time. In the following we will investigate possible ways to use the (maybe) correct information from an erroneous UDP packet at the application layer.

There is a difficulty caused by the fact that H.264 does use variable length coding. After the first bit error the code gets desynchronized and the remaining information cannot be directly used. Decoding of a desynchronized VLC stream may lead to strong visual quality degradation. In [2] a method was presented for joint sequential decoding of VLC code and error concealment that allows for correction and detection of errors within the erroneous RTP packet. As the method is still rather complex to run on a power limited mobile terminal in real time, we will concentrate on other ways how to limit the error size and propagation. Some useful error resilience features are also defined within the H.264 itself as will be described in the following section.

3 Error resilience mechanisms in H.264

One of the most important error resilience features of H.264 is slicing. To limit the size of erroneous area and spatial error propagation each video frame is segmented into parts, called slices. One slice does not reference other slices within the same frame. There are four possible slicing modes: without slicing (slice = frame), slices with constant number of macroblocks, slices with constant number of bytes or interleaved slices. There is also possibility to define the slicing arbitrary by means of Flexible Macroblock Ordering (FMO). If no slicing is used, the loss of a packet results in loss of the whole frame. Using the same number of macroblocks per slice results in packets with different length in bytes. Therefore there is a possibility to use the slices with the same number of bytes. One of the possibilities to ease the error concealment is the interleaved slicing. In this case if a slice gets lost, it can be concealed from the diagonally neighboring macroblocks belonging to other slices. However, this method provides lower compression gain due to the inefficiency of the spatial prediction.

Another important error resilience feature to stop temporal error propagation is inserting the refreshment frames, slices or macroblocks. Inserting frames or slices causes significant peaks in the resulting bit rate of the stream. For transmission over mobile systems simple profiles generating a sequence of P frames is usually used, with I frame inserted every 2-5 seconds. In Table 1 the overhead introduced by inserting I frames with different frequency can be seen for several chosen I frame frequencies.

I frame frequency	Data rate increase [%]
1	470.16
2	211.61
4	97.16
10	34.93
15	23.41
50	2.96

Table 1: Data rate increase caused by inserting I frames with different frequency into the video stream. The increase is calculated relative to the data rate of sequence encoded with I frame only at the beginning.

We obtained these Table using H.264 codec [5], applying it to the foreman sequence with QCIF resolution. We used quantization parameters QP=28, did not use B frames, encoded into RTP stream with slicing mode 0. According to our further experiments with various typical test sequences (foreman, akiyo, football, container, news, flower garden) the data rate increase O_1 caused by inserting the I frame every k frames can be estimated as follows:

$$O_1 \approx \left(-0.078 + \frac{4.673}{k}\right) \cdot 100\%, \qquad (1)$$

for $k > 0$. Please, note that the data rate increase caused by inserting I frames get higher if the sequence is smooth in time (i.e. does not contain fast scene changes) since otherwise the temporal prediction cannot be efficient. The foreman sequence is a video sequence containing both - slow (at the beginning) and fast movement (at the end) and therefore values listed in the Table 1 represent almost the average overhead. Inserting I frames, slices and macroblocks were studied in more details in [4].

To enable unequal error protection and/or differentiated prioritization in the underlying network, H.264 allows data partitioning. Data partitioning sorts the information into the packets according to its importance.

4 Additional detection mechanisms

The effects of the built in resilience features and their usage (slice sizes, I frames insertion, etc.) have been already studied, for example in [3, 4]. Our intention is to investigate additional possibilities of improvements and their costs.

4.1 Resynchronization of VLC

For resynchronization of the Context Adaptive Variable Length Code (CAVLC) in H.264 additional synchronization marks would be needed to be inserted into the video stream. A tradeoff has to be found between the overhead and the frequency of the marks determining the granularity of detection. The marks can be inserted for instance after every M macroblocks. Then the smallest detectable error area would be exactly M macroblocks large. Let m be the number of bits per synchronization mark, then we will need to add

$$O_2 = \frac{\left\lfloor \frac{\text{No\_of\_MBs}}{M} \right\rfloor \cdot m}{\text{Slice\_size}} \cdot 100\% \qquad (2)$$

of additional overhead to each slice, where No_of_MBs is number of macroblocks per slice (99 for 16×16 pixel large macroblocks in QCIF resolution frame) and Slice_size is the size of the slice in bits. Minimum size of the synchronization mark m needed will depend on the CAVLC codewords set and thus on used quantization parameters (for marking the start of the slice 24 bit long prohibited sequence is used). In Figure 1 the distribution of P and I frame sizes in the sequence foreman can be seen for slicing mode 0. The mean for P frames distribution is 690 bytes, for I frames it is 3310 bytes.

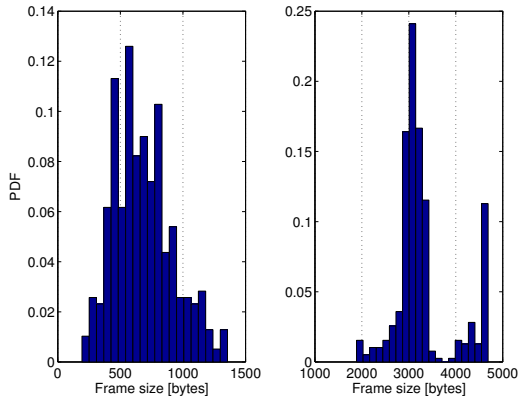

Figure 1: Distribution of frame sizes for P and I frames in the whole sequence foreman encoded by H.264.

Using the mean of the I and P frame size and assuming slicing mode 0, we obtain the relative overhead needed for synchronization marks that can be seen in Figure 2.

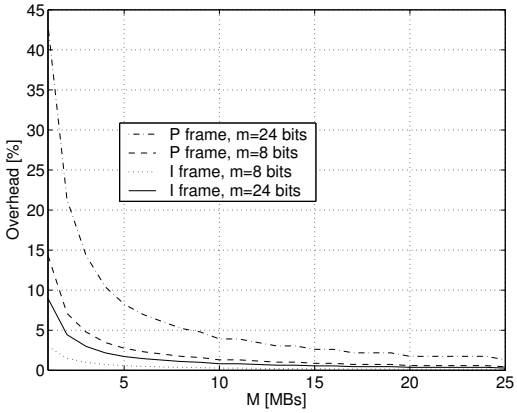

Figure 2: Overhead caused by 'in-stream' synchronization marks.

While the overhead becomes too high for P frames, for I frames this method could still be used. However, signaling the synchronization marks 'in-stream' has a disadvantage in bursty error environment, because the synchronization mark can be erroneous as well as the data. If there is an error in the synchronization mark, the VLC may desynchronize anyhow making the whole detection mechanism unreliable. Making synchronization marks more robust requires increasing of m, resulting in even more overhead. The 'out-of-stream' signaling solves this problem. The points of resynchronization can be sent via an additional 'channel' out of the actual VLC stream. We can signal the position of the resynchronization within the bitstream. Let N be the size of a slice in bits. Then the minimum position indicator length m_p can be written as:

$$m_p = \lceil \log_2 \text{Slice\_size} \rceil \text{[bits]}. \qquad (3)$$

To save the bits, we can better signal the length of the bitstream between two synchronization points within a slice. This is not as robust as the previous method, but reduces the overhead to on average

$$m_l = \lceil \log_2 N_s \rceil = \left\lceil \log_2 \frac{\text{Slice\_size} \cdot M}{\text{No\_of\_MBs}} \right\rceil \text{[bits]}, \qquad (4)$$

where N_s is the maximum length of a bit stream segment between two position indicators. The length N_s depends on QP and efficiency of the compression and therefore also from the sequence content. The above formula only provides an estimation of the synchronization mark size. The

total overhead to the bitstream is given by:

$$O_3 = \frac{\left\lfloor \frac{\text{No\_of\_MBs}}{M} \right\rfloor \cdot m + \text{Header}}{\text{Slice\_size}} \cdot 100\%, \quad (5)$$

where m can be m_p or m_l and Header is the length of the header of the 'out-of-stream' packet containing the position/length indicators. This header can introduce more overhead than the payload if we apply again the (RTP)/UDP/IP protocol. This could be solved by using the data partitioning and adding the position/length indicators to the data partition A. Such solution would require a small addition to H.264 standard - for instance using the five reserved bits to signalize the length of the position/length indicator bit sequence added at the end of the packet. A comparison of the example overhead for discussed methods can be seen in Figure 3.

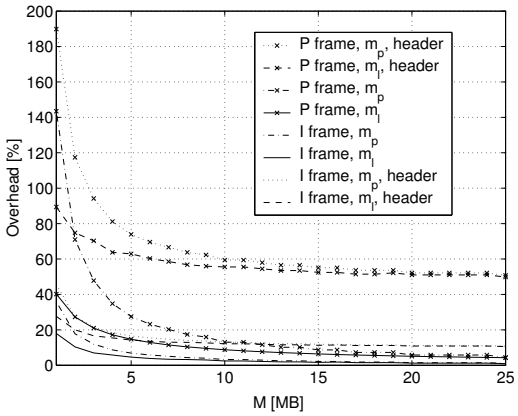

Figure 3: Overhead caused by 'out-of-stream' synchronization marks.

For the results presented in the figure, slicing mode 0 was assumed as well as the mean I and P frame sizes. The overhead caused by header of RTP/UDP/IP packet is not acceptable especially for the P frames. According to the Figure 2 and 3, a granularity of three and more macroblocks seems to be feasible from an overhead point of view as from this number the curves start to decay slowly.

4.2 Parity bits for detection

The approach discussed in the previous sections results in isolated corrupted areas of size of M macroblocks. However, we would like to additionally know what area contains an error to be able to call the concealment procedure. Therefore still a reliable detection mechanism is needed. There are more possibilities how to perform the detection. In [2] an artifact detection was proposed using the knowledge about the rectangular shape of the error area. This could be used in our case as well, the reliability is limited by the amount of distortion caused by the VLC desynchronization. A more reliable way to detect the error is to use parity bits over either defined number of macroblocks or bytes. Using the parity in the same way as the synchronization marks - over macroblocks and not over particular number of bytes - has the advantage of constant granularity. Using the synchronization and parity over bytes could cause artifacts of different size according to the compression efficiency. In Figure 4 the average overhead added to the bit stream for different number of parity bits is shown (RTP/UDP/IP was not considered, $M = 3$).

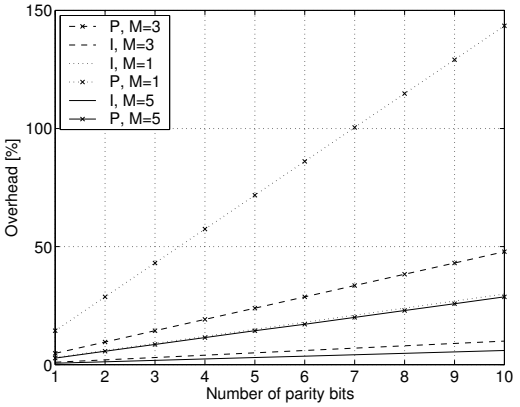

Figure 4: Average overhead caused by adding m parity bits after every $M = 3$ macroblocks.

The number of added parity bits determines the reliability of the detection. According to the presented graph one could choose M determining the granularity and m determining the reliability for I and P frames.

Please, note that we could also use the parity bits without resynchronizing the VLC, but in such case we could only find the first error and the rest of the VLC stream would not be possible to detect anyhow.

5 Results and discussion

5.1 Experimental setup

To experiment with H.264 we used the Joined Model (JM) software [5], version 7.3. We modified the source code and added following simple error concealment methods [6]:

- boundary matching temporal error concealment (used for I frames if there was no scene cut),
- motion compensated temporal error concealment (used for P frames),

- weighted averaging spatial error concealment (used for I frames in case of a scene cut).

Slices with the same number of macroblocks (33) were used. We did not use B frames to ease and speed up the decoding process. An I frame was inserted every 15 frames, quantization parameter was set to 28. We wanted to compare the quality degradation caused by missing areas of different sizes and overhead needed for such granularity.

5.2 Results

In Figure 5 the snapshots of the foreman sequence after the detection with various granularities are shown: one slice, three macroblocks and one macroblock. On the left side the errors in the I frame and their spatial propagation can be seen, on the right side these errors are concealed.

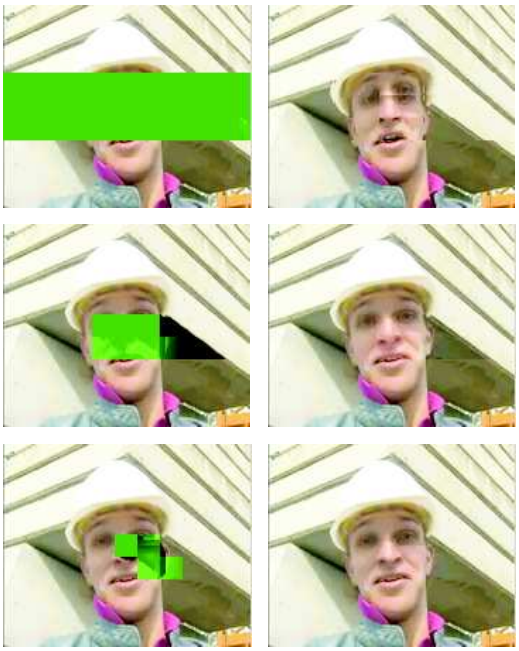

Figure 5: Screenshots of foreman test sequence decoded without (left) and with (right) error concealment for (top to bottom) missing slice, missing 3 macroblocks, missing 1 macroblock.

The PSNR of the luminance (Y-PSNR) for concealed and not concealed I frames from previous snapshots are shown in Table 2.
Simulating with various typical sequences, we obtained Figure 6, showing the average luminance PSNR over M for P and I frames with and without the error concealment.

We used rather efficient error concealment methods, so that P frames especially in case of low-movement sequences could be fully recovered. One can see that an appropriate error detection

M	Y-PSNR [dB]	Y-PSNR$_c$ [dB]
0	9.34	27.46
3	18.05	37.12
1	21.84	44.67

Table 2: Picture improvement for detection granularity $M = 1$ and $M = 3$ with and without error concealment.

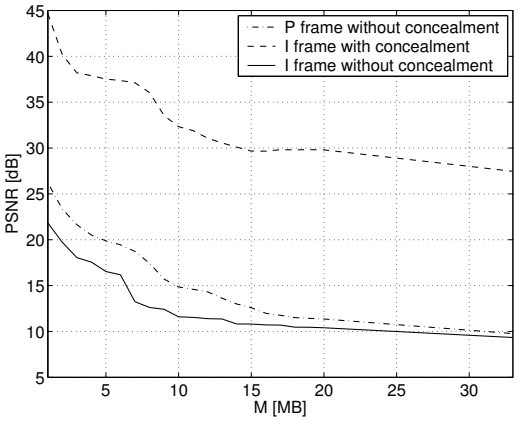

Figure 6: PSNR of luminance over M for I and P frames.

can result in an increase of PSNR by 15 dB. This is especially important for I frames, since the errors from an I frame propagate in time until the next I frame arrives.

5.3 Discussion

But still, we have to solve the problem of the optimal choice between the granularity and the overhead. An approach can be defining the efficiency ratio R for the granularity M as follows:

$$R(M) = \frac{\Delta\text{PSNR}(M)}{O(M)} = \frac{\text{PSNR}(M) - \text{PSNR}_s}{O(M)}, \quad (6)$$

where PSNR$_s$ is the PSNR obtained for the whole slice missing. By finding the maxima of this efficiency ratio we obtain M optimal in sense of the PSNR improvement to overhead relation. For our experiment results we obtained the maxima for I frames without concealment for $M = 5$, for I frames with concealment for $M = 6$, for P frames without concealment for $M = 6$ and for P frames with concealment for $M = 8$. Please, note that the case without concealment corresponds to the synchronization marks only and the case of concealment also requires the error detection by parity bits. Therefore also the efficiency ratio can have different values.

The intention of our simple experiments was to show the influence of detection granularity on

the needed overhead and picture quality improvement. We simplified the task. To design an exact method, more assumptions would be needed: to take into account the character of the errors in an underlaying system, to design the method depending on the video sequence character (faster sequences need fined granularity to achieve the same results), to consider the errors in signaled information etc. Please, note that in praxis, often simpler error concealment methods are used. In that cases an appropriate error detection is even more important.

6 Conclusions

In this paper we analyzed the possibility of improving the error resilience of the real time video transmitted over error prone networks as for instance mobile networks. We proposed two additional methods and investigated their qualities and costs by means of simple experiments with modified H.264 codec. Both - inserting the additional VLC synchronization marks and parity bits - have configurable overhead and provide configurable granularity of detection. This could be also set in an adaptive manner by means of a feedback mechanism according to the channel conditions or sequence character if necessary.

References

[1] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, "Overview of the H.264/AVC Video Coding Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003.

[2] C. Weidmann, P. Kadlec, O. Nemethova, A. Al-Moghrabi, "Combined Sequential Decoding and Error Concealment of H.264 Video," Proc. of IEEE MMSP, 28 Sep. - 1 Oct. 2004, Siena, Italy.

[3] B. Jung, Y. Hwang, B. Jeon, M. Kim and S. Choi, "Error Resilient Performance Evaluation of MPEG-4 and H.264," Proc. of SPIE, Visual Communications and Image Processing (VCIP), July 2003.

[4] T. Halbach, S. Olsen, "Error robustness evaluation of H.264/MPEG-4 AVC," Proc. of Visual Communications and Image Processing 2004, pp. 617-627, San Jose, California, Jan. 2004.

[5] H.264/AVC Software Coordination, "JM Software," ver.7.3, available in http://iphome.hhi.de/suehring/tml/.

[6] M.T. Sun, A.R. Reibman, "Compressed Video over Networks," Signal Processing and Communications Series, Marcel Dekker Inc., New York, 2001.