Proc. of IST Mobile Summit 2006, Mykonos, Greece, 4-8 June 2006

Improved Sequential Decoding of H.264 Video with VLC Resynchronization

Claudio Weidmann
ftw. Telecommunications Research Center Vienna
Donau-City-Strasse 1, A-1220 Vienna, Austria
Email: weidmann@ftw.at

Olívia Némethová
Vienna University of Technology
Gusshausstrasse 25/389, A-1040 Vienna, Austria
Email: onemeth@nt.tuwien.ac.at

Abstract— **Data partitioning in H.264 Extended Profile video coding enables unequal error protection, which is particularly interesting for wireless applications. It has been shown that sequential decoding of the prediction residuals encoded in low-priority packets can drastically improve the quality of the decoded video. We present a detailed analysis of such a sequential decoder and propose additional synchronization side information as a simple means to further increase its performance.**

I. Introduction

Mobile wireless video needs robust source decoders in order to overcome the inevitable variations in channel quality, which cannot always be mitigated by the lower layers of the protocol stack, especially in broadcast scenarios. Consequently, cross-layer methods for error resilient video transmission are receiving considerable attention, including even more closely integrated methods such as joint source-channel decoding (see e.g. [1] and references therein). In particular, sequential decoding has been proposed as a simple means to increase robustness of a H.264 Extended Profile video decoder [2], [3].

The Extended Profile of the H.264 video coding standard provides an error resilient mode which partitions data according to its importance [4]. Header data and motion vectors of a slice are labeled type A, so that they can be better protected. The *residuals* (prediction differences) of intra frames are labeled type B, while inter-predicted residuals are type C. The prediction residuals are encoded with a context-adaptive variable-length code (CAVLC), which has a simpler and potentially more robust structure than the arithmetic encoding available in the Main Profile. Both type B and C data can be less protected than type A. All data of a given type is separately encapsulated into network abstraction layer units (NALUs), which are put in RTP packets for transmission over an IP network. Data partitioning does not apply to instantaneous decoding refresh (IDR) pictures, which provide decoder restart anchors and should therefore be heavily protected.

In this paper we consider a scenario where all packets are equipped with a CRC to detect errors. Type A and IDR packets are protected by a stronger channel code than type B/C, so that one can assume that all A and IDR packets are received without error. However, the weaker (or absent) protection of B/C packets causes them to be received with random errors. Classic packet-loss receivers simply discard packets that fail their CRC and try to conceal the lost slice, while a soft-input sequential decoder will try to decode also packets containing errors. Therefore we assume that the physical layer provides the soft information (log-likelihood ratios, LLR) for the bits in these packets.

Section II details the decoder architecture, while Section III analyzes the sequential decoder in more detail. The use of additional synchronization information is proposed in Section IV and simulation results are presented in Section V.

This research has been conducted within the NEWCOM Network of Excellence in Wireless Communications funded though the EC 6th Framework Program. ftw. is supported by the Kplus competence center program of the Austrian Government.

II. Decoder Architecture

A. Sequential CAVLC Decoder

In H.264 all prediction differences (called *residuals*) are basically encoded with the same method, regardless of their origin. We describe the most common case, the 4×4 luma residuals.

A sub-macroblock (SMB) of 4×4 residual pixels is transformed and quantized. The quantization indices are scanned in zig-zag order and the resulting sequence of *coefficients* is encoded using the CAVLC. First, a VLC encodes the number of nonzero coefficients and the number of trailing ones (i.e. up to three coefficients of amplitude one at the end of the sequence, ignoring any zeros in between). The signs of the trailing ones are then coded using one bit per coefficient. The VLC table in this first step is chosen depending on the number of nonzero coefficients in the two SMBs left and above of the current SMB. Second, the values of the remaining nonzero coefficients are encoded, starting with the last coefficient preceding the trailing ones. Values are coded with an Elias-type code that consists of a variable-length prefix and a fixed-length suffix, whose length depends on the current prefix VLC table or the prefix itself (escape mechanism). The VLC table is switched based on the currently decoded coefficient value. Third and fourth, the number of zeros between the nonzero coefficients and the zero run-lengths are encoded. The code is again adaptive; the VLC table used depends on the number of zeros still left to code. These encoding operations are repeated for all the SMBs of a picture slice in order to form a type

B/C packet. In summary, the encoding of residual coefficients depends causally on previous data in the same B/C packet and on the relevant information from A packets.

This causal code structure is well matched to sequential decoding, which was originally proposed for convolutional channel codes by Wozencraft and Reiffen and later refined by Fano [5]. A list of partial decoding paths is kept in memory and each is labeled with a metric that allows comparing paths of different length. Since the list size shall be limited, the decoder needs to decide which paths to explore further based on the path metric. Several strategies exist, one of the simplest involves storing the paths in a stack which is sorted according to the metric. The top path (with the highest metric) is replaced by its extensions (a corresponding number of low-metric paths will be dropped from the stack) and the stack is sorted again. These steps are repeated until the top path has the required length and can be output as the decoded path.

The choice of metric determines the performance of sequential decoding. Massey [6] has shown that the heuristic metric introduced by Fano minimizes the error probability, provided the so-called "random tail assumption" holds. Consider a message w that is encoded with the binary variable-length codeword $x_{w,1}x_{w,2}\dots x_{w,\ell(w)}$ and transmitted over a binary-input memoryless channel with transition probabilities $P(y|x)$. The received vector $\boldsymbol{y}$ is assumed to be longer than the codeword $\boldsymbol{x}_w$. Then the random tail assumption states that the bits following the codeword (and belonging to the next codeword) are chosen i.i.d. with some distribution Q. For a good binary source code this is approximately satisfied with $Q=(\frac{1}{2},\frac{1}{2})$. Then the *a posteriori* probability that message w has been sent is

$$\Pr(w|\boldsymbol{y}) = P(w)\prod_{i=1}^{\ell(w)}\frac{P(y_i|x_{w,i})}{P_0(y_i)}, \tag{1}$$

where $P(w)$ is the *a priori* probability that w has been sent and $P_0(y_i)=\sum_x P(y_i|x)Q(x)$ is the marginal channel output distribution induced by Q. The metric is now simply the logarithm (usually base two) of $\Pr(w|\boldsymbol{y})$:

$$L(w,\boldsymbol{y}) = \log P(w) + \sum_{i=1}^{\ell(w)}\log\frac{P(y_i|x_{w,i})}{P_0(y_i)}. \tag{2}$$

Using $Q=(\frac{1}{2},\frac{1}{2})$, the argument of the right-hand "channel term" can be directly computed from the soft inputs, e.g. the LLRs $\log\frac{P(y_i|1)}{P(y_i|0)}$. Extending the metric to sequences $w_1^k = w_1w_2\dots w_k$ is straightforward: the *a priori* term $\log\Pr(w_1^k)$ can be decomposed into the sum $\sum_{i=1}^{k}\log\Pr(w_i|w_1^{i-1})$, which takes care of dependencies on past message symbols, e.g. due to syntax and/or semantics of the H.264 CAVLC. The channel term in $\log\Pr(w_1^k|\boldsymbol{y})$ is clearly additive; its summands will have to be conditioned on w_1^{i-1}, since the choice of VLC codebook for w_i may depend on past symbols.

The *a priori* probabilities $P(w)$ must be known in order to compute this MAP metric. If we assume that the compression is efficient, the probability of emitting a codeword will be exponentially related to its length:

$$P(w) = \frac{2^{-\ell(w)}}{\sum_w 2^{-\ell(w)}}. \tag{3}$$

It can be seen that by this assumption, (2) reduces to the maximum likelihood (ML) metric.

A key difference to sequential decoding of convolutional codes is the fact that not all syntactically valid paths correspond to valid decodings of a packet, since the header information imposes additional constraints. Only paths that have the correct length in bits *and* encode the correct number of SMBs (in the slice) are valid decoder outputs. This yields some error correction capability, since semantically invalid paths can be eliminated from the decoder stack.

B. *Artifact Detection*

The CAVLC for the residuals contains also fixed-length-coded (FLC) fields, for example to code sign and mantissa of coefficient values. The decoding metric assigns uniform probabilities to FLC fields, hence FLC errors go undetected and will cause some distortion. A more severe decoding error occurs when the sequential decoder outputs a wrong VLC path. We try to detect both types of decoder errors in order to request concealment of the affected macroblocks (MBs). For FLC errors, we use the soft (LLR) inputs to compute an estimate of the expected distortion, which is input to an multi-criterion artifact detection procedure [2]. The main detection mechanism, which targets also VLC errors, is based on thresholding a difference picture metric. The threshold is adapted to the instantaneous amount of motion to ease correct decisions for sequences with different temporal character.

C. *Error Concealment*

To keep the proposed method as generally applicable as possible, we did not want to use object-based error concealment methods. Our simulations focused on QCIF video (176×144 pixels or 11×9 MBs). At such low resolutions, a lot of visual information is contained in a single MB and therefore temporal interpolation provides in most cases a better and simpler basis for concealment than spatial or spatial frequency domain interpolation. We have chosen temporal error concealment with boundary matching [7] also for I frames. Factors that make the temporal error concealment more difficult, such as scene cuts, transitions and fast zooming in/out were not present in our simulation scenario.

III. Sequential Decoder Analysis

We may distinguish several kinds of redundancy that can be exploited by the sequential decoder:

- Mismatch between the actual source and its model in the encoder (e.g. using a memoryless model for a Markov source).
- Mismatch between ideal and actual codeword lengths (e.g. integer codeword lengths $\ell_i \neq -\log_2 p_i$, unused leaves/codewords in the VLC tree).
- Semantic side information about the encoded content.

The first two kinds of redundancy are of little importance in H.264: on the one hand, the CAVLC syntax has been closely matched to the correlation structure present in the residual data. Our experiments showed that the codewords are used with close to their ideal probability, in particular the short ones. As a consequence, having more precise *a priori* source probabilities, whether transmitted as side information or estimated online, yields only minor performance improvements, if at all. Efficient use of statistical model redundancy is also hampered by typical relatively small packet sizes. On the other hand, although there are a few unused codewords in some of the VLC tables, they generally differ in just one position from the longest codeword in a given table (examples include the often used tables `level_prefix` and `run_before`); thus they are unlikely to occur as result of a bit error, which would otherwise be detected. In summary, little redundancy is left in the data stream that could be easily exploited by the decoder; the statistical properties of the encoder output are close to those of a binary symmetric source. This finding is not very surprising given the excellent compression performance of H.264 and justifies choosing the *a priori* probabilities (3).

These observations suggest that the main source of redundancy must be the semantic side information contained in the A packets, that is, outside the actual residual data stream in the B/C packets. That information puts semantic constraints on the contents of the residual packets. Of these constraints, the simplest to exploit in a sequential decoder is the knowledge of the number of SMBs that are encoded in a given packet of length n. If a slice contains up to m MBs, this amounts to about $\log_2(16m)$ bits of side information at most, which for practical frame resolutions corresponds to 10–20 bits, independent of packet size. This fits well with the observed ability of the decoder to correct a small number of hard decision bit errors (typically less than five). However, this redundancy does not grow with data rate or packet size, thereby imposing limits on the practical operating range of a sequential decoder based on this kind of side information. The next section proposes additional synchronization information as means to alleviate this limitation.

IV. Additional synchronization information

Synchronization marks are a popular way to increase robustness of source coding schemes. They allow to restart the VLC decoder at known positions and thus confine the propagation of decoding errors, which are thereby also easier to detect and conceal. There is a vast literature on resynchronization in video coding, see e.g. [8] and references therein. In [9], the improvement achievable by having synchronization marks every m-th macroblock was analyzed together with the additional overhead needed. Synchronization marks combined with additional parity bits allow an even more reliable detection of areas with errors, making the concealment more efficient and the visually degraded area smaller.

In the present work, we use out-of-band side information to signal additional synchronization points to a sequential decoder. The boundaries of MBs in the code stream are natural candidates for resynchronization, since the decoder already checks for the correct number of MBs at the end of a packet. The decoder is informed of the current bit position every m-th MB in a slice, i.e. it knows the length n (in bits) and the position of a segment of the code stream encoding m MBs (or less, at the end of a slice). Note that this is not the same as having smaller packets of at most m macroblocks, due to the prediction mechanism that causally affects the encoding of MBs within a slice. The sequential decoder uses the extra side information to simply discard paths that do not line up correctly at the synchronization points.

Encoding the side information can be done with an Elias-γ code, which needs $\lfloor \log_2 n \rfloor + 2\lfloor \log_2 \lfloor \log_2 n \rfloor + 1 \rfloor + 1$ bits to encode the natural number n. To convey this information to the decoder without errors, it could be appended to the A packets; however, that is not standard-compliant and might break compatibility with other receivers. A better way is to use a reserved NALU type, which will be ignored by receivers not knowing how to handle it. The additional packetization overhead is negligible if the side information can be precomputed to generate packets of the maximal allowed size. The decoder needs only buffering capability for one additional packet.

Providing a synchronization point every m MBs has two shortcomings: first, the rate of side information is still limited by its granularity on the MB level (possibly sub-MB). This could be problematic for video encoded at high rates with a small quantization parameter, resulting in large intervals between (sub-)MB boundaries in the code stream. However, that is unlikely to be a problem in low to medium rate wireless applications. The other shortcoming is more serious: increasing the frequency of synchronization points does not necessarily increase the capability of correcting (or detecting) VLC errors, regardless of decoder complexity. For example, it cannot prevent confounding two codewords of the same length. Furthermore, errors in the FLC fields of the code stream still go completely unnoticed. Fortunately, both problems can be mitigated by proper artifact detection and error concealment.

V. Results and Discussion

All simulations are based on the Foreman QCIF sequence, encoded with H.264 (joint model encoder, 200 frames at 15 fps, $QP_I = QP_P = 30$, I9P GOP, maximal packet size 750 bytes) and then transmitted over a binary-input additive white Gaussian noise (BIAWGN) channel. The total file size was 169'509 Bytes (corresponding to about 100 kbit/s), of which 69'197 were in error-free A packets. The remaining 100'312 Bytes were in B/C packets and were decoded with the sequential decoder.

Synchronization points were added with a frequency of $m_I = 4$ in I frames and $m_P = 20$ in P frames, resulting in the same average side information rate of 2% in both frame types, or 15837 bits in total. To model error-free transmission of this information, we added 3×40 bytes of headers and assumed a channel code of rate 0.9, which in the practical decoder operating region above 6 dB E_b/N_0 is at more than 3 dB from capacity. All together, this corresponds to an increase

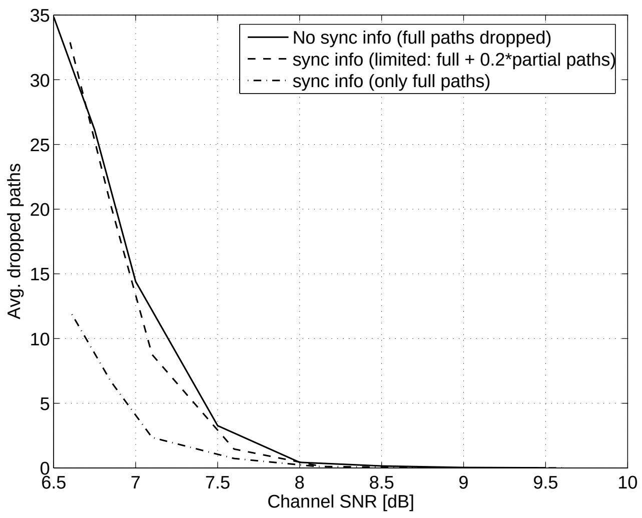

Fig. 1. Average number of dropped paths per slice.

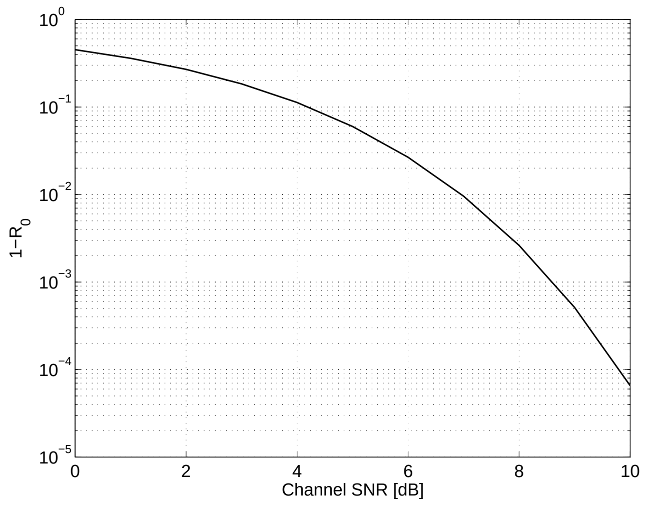

Fig. 3. Redundancy $1 - R_0$ at cut-off rate of the BIAWGN.

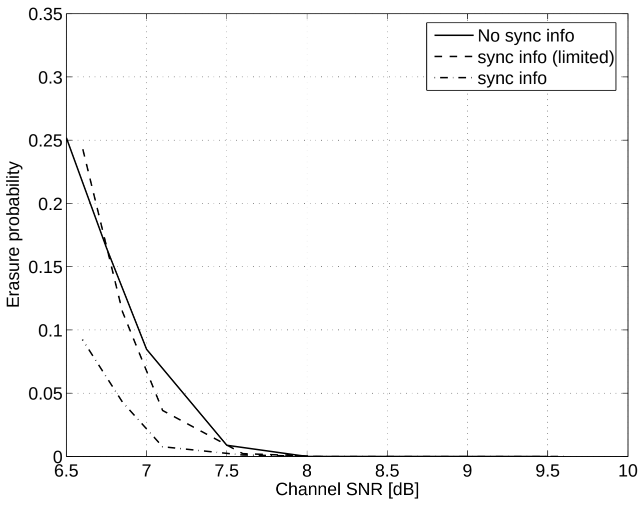

Fig. 2. Slice erasure probability.

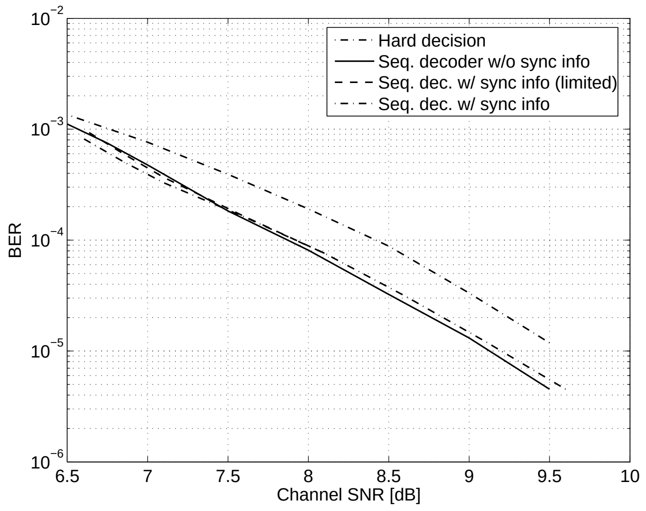

Fig. 4. Bit error rates vs. channel SNR

of 0.1 dB in E_b/N_0 compared to a reference system without additional synchronization information. To make comparisons possible, this shift is included in all plots shown.

The computational complexity of sequential decoding becomes exponential when operating above the cut-off rate R_0, which is a function of the modulation alphabet and channel used [10]. Therefore we limited the complexity by letting the decoder drop at most 100 invalid paths before declaring a slice *erasure*. Since the decoder with synchronization points drops many more shorter partial paths, we generated two sets of results: one were only full-length dropped paths were counted, and one were five dropped partial paths were counted as one full path (labeled “limited” in the plots). The actual complexity of the latter was smaller than the reference, while the former gets about twice as complex with decreasing channel SNR.

The average number of dropped paths is shown in Figure 1 and the slice erasure probability in Figure 2. The cut-off phenomenon is clearly visible around 7.5 dB. This value can be used to estimate the average redundancy that is actually exploitable by the decoder: Figure 3 shows the redundancy per channel input bit corresponding to the BIAWGN cut-off rate, which is about 0.005 at 7.5 dB. For a mean packet size of 330 bytes this turns out to 13 bits per packet, to be compared with the estimate of 10-20 bits in Section III. A bit surprisingly, additional synchronization information does not shift the cut-off point by much. This may be partly explained by error propagation beyond synchronization points due to the CAVLC prediction mechanisms.

Figure 4 shows the bit error rate (BER) performance of the different decoders. Over a wide range of channel SNR, there is a coding gain of about 0.5 dB compared to binary hard decision decoding. Additional synchronization information is penalizing above 7.5 dB, however BER is not relevant for decoded picture quality, as can be seen from the YUV-averaged PSNR in Figure 5. Results are shown for different decoders and different concealment scenarios. Lost or erased packets (slices) are either left as is or concealed with copy-paste from the previous frame. The best results are obtained

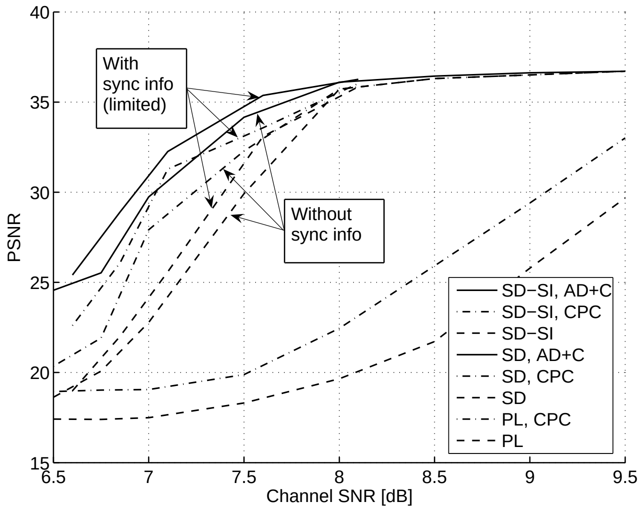

Fig. 5. Decoder comparison, best to worst: sequential decoder with sync info (SD-SI), sequential decoder alone (SD), packet loss (PL); combined with artifact detection and concealment (AD+C), simple copy-paste concealment of erased slices (CPC), no concealment.

when combining the sequential decoder with the artifact detector that generates concealment requests. The additional synchronization information results in about 0.1–0.2 dB gain in E_b/N_0 (after the 0.1 dB penalty). Figure 6 shows that the visual gains are not necessarily reflected in PSNR and can be quite impressive ($E_b/N_0 = 6.75$ dB).

VI. Conclusion

Recent years have seen growing interest in error-resilient source decoding methods, ranging from simple error detection with repeat request to elaborate iterative joint source-channel decoders. Many of these methods are unrealistically complex or cannot be used without breaking standards. Sequential decoding clearly fills a gap in this respect, as it allows for a reasonable complexity-performance trade-off. This work has shown that its performance can be further improved in a simple fashion. The presented scheme exploits the knowledge of macroblock boundaries in the code stream as means to eliminate incorrect paths in a sequential decoder. Section III mentioned that this is the simplest kind of semantic side information that can be exploited; other kinds would include e.g. knowledge about motion compensation or intra prediction. However, these are more difficult to handle, since they are likely best processed in the spatial domain, e.g. with a path metric that takes into account the distortion of the reconstructed slice compared to its prediction from side information. Besides representing a considerable complexity increase, this would require a much closer integration of the sequential CAVLC decoder with the actual H.264 decoder.

In summary, sending a small amount of synchronization side information is an easy way to boost sequential decoder performance while keeping the computational complexity constant or even reducing it.

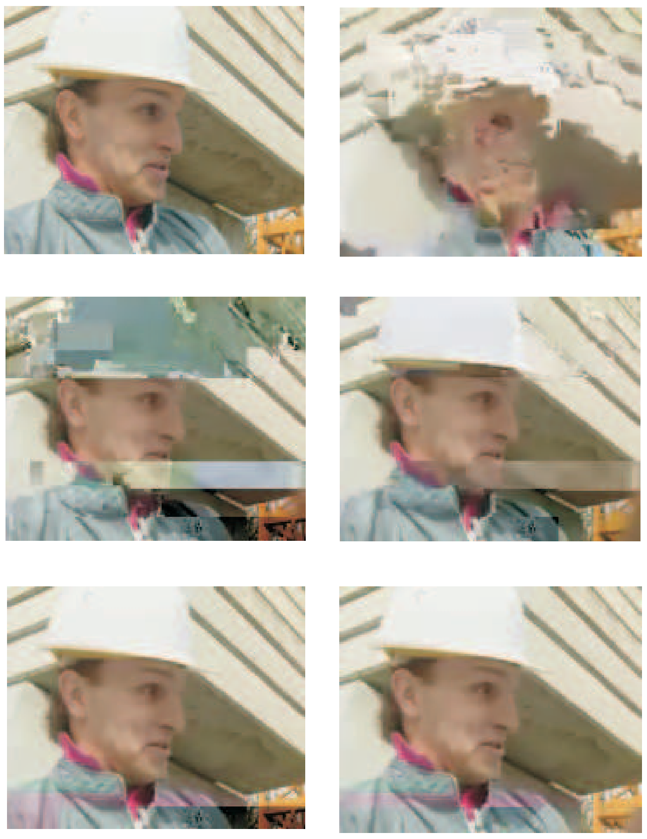

Fig. 6. Original picture, packet loss with copy-paste concealment, sequential decoding alone, sequential decoding and concealment, additional sync info alone, additional sync info and concealment (left to right, top to bottom).

Acknowledgment

The authors thank Prof. Markus Rupp from Vienna University of Technology for supporting their research collaboration.

References

[1] F. Zhai, Y. Eisenberg, and A. K. Katsaggelos, “Joint source-channel coding for video communications,” in *Handbook of Image and Video Processing*, 2nd ed., A. Bovik, Ed. Elsevier, June 2005.

[2] C. Weidmann, P. Kadlec, O. Némethová, and A. A. Moghrabi, “Combined sequential decoding and error concealment of H.264 video,” in *Proc. MMSP*, Siena, Italy, Sept. 2004.

[3] C. Bergeron and C. Lamy-Bergot, “Soft-input decoding of variable-length codes applied to the H.264 standard,” in *Proc. MMSP*, Siena, Italy, Sept. 2004.

[4] Joint Video Team of ITU-T and ISO/IEC JTC 1, “ITU-T rec. H.264 — ISO/IEC 14496-10 AVC,” Mar. 2003, document JVT-G050 available on http://bs.hhi.de/~wiegand/JVT.html.

[5] R. M. Fano, “A heuristic discussion of probabilistic decoding,” *IEEE Trans. Inform. Theory*, vol. IT-9, pp. 64–73, Apr. 1963.

[6] J. L. Massey, “Variable-length codes and the Fano metric,” *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 196–198, Jan. 1972.

[7] Y. Jung, Y. Kim, and Y. Choe, “Robust error concealment algorithm using iterative weighted boundary matching criterion,” in *Proc. of IEEE Int. Conf. on Image Processing*, vol. 3, Sept. 2000, pp. 384–387.

[8] G. Côté, S. Shirani, and F. Kossentini, “Optimal mode selection and synchronization for robust video communications over error-prone networks,” *IEEE J. Select. Areas Commun.*, pp. 952–965, June 2000.

[9] O. Nemethova, J. Rodriguez, and M. Rupp, “Improved detection for H.264 encoded video sequences over mobile networks,” in *Proc. 8th ISCTA*, Ambleside, UK, July 2005, pp. 343–348.

[10] J. L. Massey, “Coding and modulation in digital communications,” in *Proc. Int. Zurich Seminar on Digital Communications*, Switzerland, Mar. 1974, pp. E2(1)–E2(4).