Accepted in the 6th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2012), 2012

A Maximum Likelihood Approach to Video Error Correction Applied to H.264 Decoding

François Caron
Department of Software and IT Engineering
École de technologie supérieure, Université du Québec
1100 Notre Dame St. West, Montreal, H3C 1K1, Canada
cc-francois.caron@etsmtl.ca

Stéphane Coulombe
Department of Software and IT Engineering
École de technologie supérieure, Université du Québec
1100 Notre Dame St. West, Montreal, H3C 1K1, Canada
Stephane.Coulombe@etsmtl.ca

***Abstract*—In real-time video applications, where unreliable networks are commonplace, corrupted video packets can adversely affect visual quality. In this paper, we present a novel maximum likelihood approach to performing video error correction. Rather than discarding corrupted video packets, the method estimates the likeliest syntactically valid video slice content based on these packets. First, we present the mathematical foundations that enable solution of the problem at the slice level. Then, we present a simplified solution operating at the syntax element level. The method's performance is evaluated using the H.264 baseline profile. Unlike error concealment methods, we correct the errors in the bitstream, instead of reconstructing missing pixels. Simulation results show that the method yields improved visual quality. Furthermore, the proposed approach is computationally simpler than state-of-the-art error concealment methods.**

Keywords-maximum likelihood; video error correction; H.264

I. INTRODUCTION

Real-time video transmission is a challenging task, especially when typical error handling mechanisms, such as retransmission, cannot be used. The H.264 standard [1], with its network friendly approach, introduced new coding tools to deal with the challenging task of sending video from one device to another. Flexible Macroblock Ordering (FMO) [3] was introduced to break the traditional raster scan, ordering allowing packets to hold non consecutive macroblocks (MBs). As a result, packet loss has less impact, and error concealment mechanisms are offered more information (boundaries). It was hopped that error concealment, under FMO, would produce better visual results.

Arbitrary Slice Ordering (ASO) [2] was introduced to break up the relationship between packets, making every slice/packet independently decodable. ASO also enhances the robustness of the data to packet loss. Data partitioning [3] – available with the Extended Profile – took the process one step further, splitting the prediction information (MB types, motion vectors, etc.) from the residual information (luminance and chrominance values), based on the argument that the prediction information was more important than the residual information. Researchers verified this assumption by applying Unequal Error Protection (UEP) schemes where data partitions A – those carrying all the syntax elements belonging to Category 2 – are better protected. Intra placement, although not a new feature, was enhanced, to allow Intra MBs to use information from neighboring Inter MBs. These tools all share the same two goals: 1) enhance robustness to data loss; and 2) assist error concealment/resynchronization. They also share the same drawback, as the added robustness comes at the cost of reduced compression efficiency (lower visual quality compared to a non error resilient scheme when there is no error).

Error concealment, by contrast, does not require additional bandwidth (or doesn't sacrifice bandwidth for error protection). Using the correctly received information, as well as information from the previous pictures, it estimates the value of the missing pixels to reconstruct the pictures when errors occur. Starting with Sun [18] and Kwok's [17] initial work, researchers have published spatial, temporal, and hybrid approaches to interpolate the lost pixels. The common denominator throughout the error concealment literature is that transmission errors only arise in the form of packet loss – corrupted packets are always discarded. However, video data fall into the class of applications that benefit from having damaged data delivered, rather than discarded [4]. Wenger conveniently illustrates the use of the *forbidden_zero_bit* in a scenario where a smart node forwards a corrupted Network Abstraction Layer Unit (NALU) to its destination [3]. Assuming that corrupted packets do reach the decoder, Weidmann [5] uses Joint Source-Channel Decoding (JSCD) to correct the CAVLC prediction residual coefficients found in data partitions B and C, using the number of MBs extracted from data partitions A, which are always intact, to impose additional constraints on the solution. Wang and Yu [6] apply JSCD to correct the motion vectors. Their experiment does not comply with the H.264 standard, however, as partitions B and C carry the horizontal and vertical motion vectors respectively. Sabeva [7] applies JSCD to CABAC encoded bitstreams, on the assumption that each packet carries an entire picture, and so the number of MBs in a packet is known a priori. As both the picture resolution and the visual quality increase, the solution becomes increasingly complex computationally. Lee [8] proposes to use Fuzzy Logic to feed information back to the channel decoder, although they provide very few details about their Fuzzy Logic engine.

Levine [9] and Nguyen [10] both apply iterative JSCD to a CABAC coded stream. A Slice Candidate Generator produces a list of hypothetical slices by flipping one or more bits in the

corrupted slices received. Each candidate is then studied at the semantic level. The bits that seem to have been correctly fixed are fed back into the channel decoder between iterations, until the likeliest bitstream is selected. Farrugia [11] uses a list decoding approach, where the M likeliest feasible bitstreams are reconstructed and evaluated in the pixel domain. Using a value of M=5, the approach produces very good visual results. However, its high computational complexity makes it prohibitively costly for most applications. Trudeau [12] proposes a two-step solution: decoding the corrupted stream without a list of candidates, and concealing the potentially lost MBs. The fit of the decoded and concealed MBs – the way they connect to the correctly received MBs in the pixel domain – are then compared, and the best fitting MBs are selected for display. He does not assume the use of any error resilience or data portioning method.

In this paper, we present a novel method for video error correction based on maximum likelihood decoding at the syntax element level. The proposed approach does not require additional overhead for forward error correction, can be applied in conjunction with error resiliency and requires fewer computations than error concealment. The correction performance translates into increased visual quality compared to state-of-the-art error concealment. Fig. 1 shows the proposed system's architecture. A video encoder first compresses and packages video slices. After channel encoding, the slices are sent to their destination via an unreliable channel. Upon reception, the channel decoder forwards hard and/or soft information (bits and/or information indicating the reliability of each bit) to the communication protocol stack, where protocol headers, typically IP headers, are checked for transmission errors. Assuming that the headers are intact, the information is then sent to the video application layer, where the headers of other protocols, such as RTP, are used. Depending on the results of the UDP checksum, the video information is sent either directly to the video decoder or to the proposed video error correction layer, to produce the likeliest video slice based on the corrupted information received. If, after decoding, MBs are still missing because they could not be repaired, they are concealed before being displayed.

The rest of the paper is organized as follows. Section II presents the slice-level maximum likelihood solution to correcting transmission errors. Section III then presents a less complex approach, in which the maximum likelihood approach is applied to each syntax element individually. A solution is then derived in section IV, specifically for four syntax elements present in an H.264 slice header. The experimental results are given in section V, and our concluding remarks are presented in section VI.

II. Slice-Level Maximum Likelihood Decoding

Let $S=\{s_1,s_2,\cdots,s_N\}$ be the series of syntax elements (SE) in a transmitted slice. For example, the H.264 standard uses the SE *first_mb_in_slice* to indicate the raster index of the MB coded first in the slice and the SE *mb_type* to indicate the MB coding type. Let $\tilde{S}=\{\tilde{s}_1,\tilde{s}_2,\cdots,\tilde{s}_{\tilde{N}}\}$ be the series of SEs in a received corrupted slice.

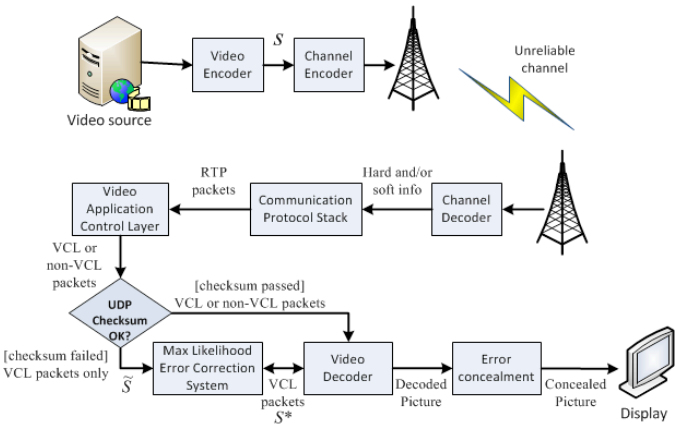

Figure 1 Proposed system architecture

Although both S and $\tilde{S}$ contain the same number of bits, the number of SEs in each slice may differ, due to transmission errors affecting variable length codewords (VLC). For clarity, let $L_S(\cdot)$ represent the number of syntax elements in a slice, and let $L_B(\cdot)$ represent the number of bits in a slice or a codeword. Since we know that $\tilde{S}$ contains at least one erroneous bit, let $H=\{\hat{S}_j \mid 0\le j<K\}$ be the set of all hypothetical syntactically valid slices of length $L_B(\tilde{S})$ that could have been sent (that is, $\forall j: L_B(\hat{S}_j)=L_B(\tilde{S})$), where $\hat{S}_j=\{\hat{s}_1,\hat{s}_2,\cdots,\hat{s}_{\hat{N}_j}\}$ is the series of SEs of the j[th] hypothetical slice. A syntactically valid slice meets all the requirements of a specific video standard (e.g. the ranges and restrictions associated with each H.264 SE defined in subclause 7.4 and Annex A respectively [1]). Note that $\tilde{S}$ is not always an element of H.

Let $S^*=\{s_1^*,s_2^*,\cdots,s_{N^*}^*\}$ be the likeliest series of syntactically valid SEs given that $\tilde{S}$ was received. Equation (1) gives the proposed slice-level maximum likelihood decoding approach for finding S^*.

$$S^*=\arg\max_{\hat{S}_j\in H}\left\{P\left(\hat{S}_j\middle|\tilde{S}\right)\right\} \tag{1}$$

Using Baye's theorem, we can express (1) as follows:

$$S^*=\arg\max_{\hat{S}_j\in H}\left\{\frac{P\left(\tilde{S}\middle|\hat{S}_j\right)\times P\left(\hat{S}_j\right)}{P(\tilde{S})}\right\}=\arg\max_{\hat{S}_j\in H}\left\{P\left(\tilde{S}\middle|\hat{S}_j\right)\times P\left(\hat{S}_j\right)\right\} \tag{2}$$

The denominator in (2) has been factored out, as it is constant and maximizing the numerator is its equivalent. The likelihood $P\left(\tilde{S}\middle|\hat{S}_j\right)$ can be modeled as $L_B(\tilde{S})$ independent Bernoulli trials with a fixed success rate, represented by the bit error rate, where the Hamming distance d_j, the number of different bits between the two slices, represents the number of successes:

$$P\left(\tilde{S}\middle|\hat{S}_j\right)=\rho^{d_j}\times(1-\rho)^{L_B(\tilde{S})-d_j} \tag{3}$$

The bit error rate ρ can either be estimated from the observed received slices, or it could be a known value guaranteed by the channel's quality of service.

The unknown number of MBs carried in a slice, combined with the sequential dependencies between SEs, makes evaluating $P(\hat{S}_j)$ for a whole slice difficult and impractical. Using the Chain Rule, we can conveniently account for the sequential dependencies between SEs and write the probability $P(\hat{S}_j)$ as follows:

$$\begin{aligned} P(\hat{S}_j) &= \prod_{i=1}^{\hat{N}_j} P\left(\hat{s}_{j,i} \middle| \hat{s}_{j,i-1}, \hat{s}_{j,i-2}, \cdots, \hat{s}_{j,1}\right) \\ &= P(\hat{s}_{j,1}) \times P\left(\hat{s}_{j,2} \middle| \hat{s}_{j,1}\right) \times \cdots \times P\left(\hat{s}_{j,\hat{N}_j} \middle| \hat{s}_{j,\hat{N}_j-1}, \hat{s}_{j,\hat{N}_j-1}, \cdots, \hat{s}_{j,1}\right) \end{aligned} \tag{4}$$

Decomposing the Hamming distance d_j for each SE (i.e. $d_j = d_{j,1} + d_{j2} + \cdots + d_{j,L_S(\hat{S}_j)}$, where $d_{j,i}$ represents the number of different bits between $\hat{s}_{j,i}$ and the bits in $\tilde{S}$ at the same positions), we can rewrite (3) as:

$$P\left(\tilde{S} \middle| \hat{S}_j\right) = \prod_{i=1}^{L_S(\hat{S}_j)} \rho^{d_{j,i}} \times (1-\rho)^{L_B(\hat{s}_{j,i}) - d_{j,i}} \tag{5}$$

Substituting (4) and (5) into (2), we obtain:

$$S^* = \arg\max_{\hat{S}_j \in H} \left\{ \begin{array}{l} \prod_{i=1}^{L_S(\hat{S}_j)} \rho^{d_{j,i}} (1-\rho)^{L_B(\hat{s}_{j,i}) - d_{j,i}} \times \\ P\left(\hat{s}_{j,i} \middle| \hat{s}_{j,i-1}, \hat{s}_{j,i-2}, \cdots, \hat{s}_{j,1}\right) \end{array} \right\} \tag{6}$$

Finding the likeliest slice S^* is computationally expensive, however. Let H^+ be a set composed of all possible slices of size $L_B(\tilde{S})$ (it includes syntactically valid as well as invalid slices). The cardinality, or number of elements, of H^+, denoted $Card(H^+)$, is $2^{L_B(\tilde{S})}$. For an ideal video standard in terms of compression efficiency, all the elements of H^+ would be syntactically valid and $H = H^+$. However, for existing video standards, such as H.264, $Card(H)$ is significantly smaller than $Card(H^+)$ (i.e. $H \subset H^+$), and even with additional constraints, $Card(H)$ will still be extremely large. It is clear that a method operating on a smaller solution space, and following the sequential behavior of existing decoders would be highly desirable for real-time video applications.

III. SE-Level Maximum Likelihood Decoding

To alleviate the problems discussed in the previous section, we propose a breadth-first approach based solely on the previously decoded SE. By maximizing the likelihood of individual SEs, or groups of SEs, rather than the whole slice, we are reducing the cardinality of the solution space, since each maximization step eliminates all but one outcome.

Let $C_i = \{\hat{c}_{i,j} \mid 0 \le j < M_i\}$ be the codebook containing all the valid codewords the ith SE can use, and let $C^* = \{c_1^*, c_2^*, \cdots, c_M^*\}$ be the series containing the likeliest codewords creating a syntactically valid slice using our proposed method. This method progressively decodes each SE by maximizing the codeword likelihood without considering the SEs in the slice that remain to be decoded. Using a similar development leading to (2), we can derive the SE-level maximum likelihood decoding solution:

$$c_i^* = \arg\max_{\hat{c}_{i,j} \in C_i} \left\{ P\left(\tilde{S} \middle| \hat{c}_{i,j}\right) \times P(\hat{c}_{i,j}) \right\} \tag{7}$$

where the likelihood $P\left(\tilde{S} \middle| \hat{c}_{i,j}\right)$ represents the ith term in (5), and $P(\hat{c}_{i,j})$ represents the probability of a codeword being selected based on the previously decoded SEs.

$$P\left(\tilde{S} \middle| \hat{c}_{i,j}\right) = \rho^{d_{i,j}} (1-\rho)^{L_B(\hat{c}_{i,j}) - d_{i,j}} \tag{8}$$

$$P(\hat{c}_{i,j}) = P\left(\hat{c}_{i,j} \middle| c_{i-1}^*, c_{i-2}^*, \cdots, c_1^*\right) \tag{9}$$

Substituting (8) and (9) into (7), we obtain:

$$c_i^* = \arg\max_{\hat{c}_{i,j} \in C_i} \left\{ \begin{array}{l} \rho^{d_{i,j}} (1-\rho)^{L_B(\hat{c}_{i,j}) - d_{i,j}} \times \\ P\left(\hat{c}_{i,j} \middle| c_{i-1}^*, c_{i-2}^*, \cdots, c_1^*\right) \end{array} \right\} \tag{10}$$

A small adjustment to (10) is required to account for the use of VLC. The maximization step should consider the same number of bits, as a codeword's exposure to transmission errors increases with its length. Since we are not using the subsequent SEs to maximize the likelihood, we can use the so-called Random Tail Assumption [5] to model the bits beyond the codeword under study as random events to account for the codeword length differences.

Let $\max(L_B(C_i))$ represent the number of bits in the longest codeword in the codebook C_i. Assuming that the coded video information is generated by a good binary source (i.e. zeros and ones are equally likely), the uninterpreted bits can be seen as random information:

$$c_i^* = \arg\max_{\hat{c}_{i,j} \in C_i} \left\{ \begin{array}{l} \rho^{d_{i,j}} (1-\rho)^{L_B(\hat{c}_{i,j}) - d_{i,j}} \times \\ \frac{1}{2}^{\max(L_B(C_i)) - L_B(\hat{c}_{i,j})} \times \\ P\left(\hat{c}_{i,j} \middle| c_{i-1}^*, c_{i-2}^*, \cdots, c_1^*\right) \end{array} \right\} \tag{11}$$

IV. Slice Header Correction Based on SE-Level ML Decoding Applied to H.264

Consider the scenario where an H.264 Baseline profile encoder sends video coding layer (VCL) packets using unreliable means and non VCL packets using reliable means. In addition, consider that the MBs are coded following the raster scan order, and that the transmitted slices are limited to a fixed number of bytes and arrive in the order in which they were sent. Finally, consider that all the packets sent reach their destination. To test our approach, we model the following syntax elements: *first_mb_in_slice, slice_type*, *frame_num* and

pic_order_cnt_lsb, since the last three SEs all depend on the SE *first_mb_in_slice*.

The SE *first_mb_in_slice* represents the raster scan index of the first coded MB carried in the slice. Under our current assumptions, the number of MBs carried in a slice can be expressed as the difference between the values used in consecutive slices associated with the same picture. For clarity, we will use the notation $c_i^{(k-1)}$ to represent the value of a reconstructed codeword from the previous slice, and $\hat{c}_{1,j}$ to represent the j[th] valid value of *first_mb_in_slice* in the k^{th} slice. In addition, let X, a discrete random variable, represent the difference between $\hat{c}_{1,j}$ and $c_1^{(k-1)}$, corresponding to the number of MBs in the previous slice. Since we know that transmission errors can affect the number of MBs extracted from a corrupted slice, and because X represents a count, let us assume that X follows a Poisson distribution. Then, the probability of an outcome $\hat{c}_{1,j}$ can be expressed using the previously reconstructed value as follows:

$$P\left(\hat{c}_{1,j}\right)=\frac{e^{-E(X)}E(X)^{\left(\hat{c}_{1,j}-c_1^{(k-1)}\right)}}{\left(\hat{c}_{1,j}-c_1^{(k-1)}\right)} \tag{12}$$

where $E(X)$ is the average number of MBs in a slice and can be estimated using the intact slices previously received. It is worth mentioning that the last slice associated with a picture is not considered in estimating $E(X)$ in our scenario. Limiting the maximum number of bytes a packet may carry introduces the possibility that the last slice contains significantly fewer MBs than the other slices since MBs associated with different pictures cannot be transported together.

The SE *slice_type* indicates the coding type employed in the current slice. Under our current assumptions, the valid outcomes are 0 or 5 for Inter coding, and 2 or 7 for Intra coding. Values above 4, corresponding to the higher range, are used to indicate that all the slices associated with the current picture share the same coding type. We assume that the encoder does not mix values from the lower and higher ranges within a picture, because if it does, the only slice using information in the higher range could be lost during transmission. This behavior has been observed in the H.264 reference software JM 18.2 [15]. We can model the SE *slice_type* as two pairwise independent Bernoulli trials. The first experiment checks for the range used, where a value in the range above 4 indicates success. The second experiment checks for the coding type, where the use of Intra coding indicates success. Furthermore, the value of the SE *first_mb_in_slice* (c_1^*) must be considered, since the effect of using a *slice_type* value above 4 is limited by the picture boundaries. The conditional probability distribution of $\hat{c}_{2,j}$ varies, based on the reconstructed value $c_2^{(k-1)}$. This means that, for each combination of $c_2^{(k-1)}$ and c_1^*, we obtain a different probability distribution:

$$P\left(\hat{c}_{2,j}\middle|\, c_1^*\right)=\begin{cases}(1-\alpha)\cdot\left(\delta\left(\hat{c}_{2,j}\right)\cdot(1-\beta)+\delta\left(\hat{c}_{2,j}-2\right)\cdot\beta\right)+ & c_1^*=0\\ \alpha\cdot\left(\delta\left(\hat{c}_{2,j}-5\right)\cdot(1-\beta)+\delta\left(\hat{c}_{2,j}-7\right)\cdot\beta\right) & \\ \delta\left(\hat{c}_{2,j}\right)\cdot(1-\beta)+\delta\left(\hat{c}_{2,j}-2\right)\cdot\beta & c_1^*\neq 0, c_2^{(k-1)}\le 4\\ \delta\left(\hat{c}_{2,j}-c_2^{(k-1)}\right) & c_1^*\neq 0, c_2^{(k-1)}>4\end{cases} \tag{13}$$

where $\delta(\cdot)$ is the discrete Dirac function ($\delta(w)=1$ if $w=0$; $\delta(w)=0$ otherwise), α represents the probability that a *slice_type* value above 4 is used, and β represents the probability that the *slice_type* value maps to Intra coding. Both probabilities are estimated from the previously reconstructed *slice_type* values.

The SEs *frame_num* and *pic_order_cnt_lsb* are used to identify pictures. They both represent the least significant bits of monotonically increasing sequences, where the use of a new value is triggered when the value of *first_mb_in_slice* (c_1^*) equals 0 (the start of a new picture). Subclause 7.4.3 [1] specifically indicates that all slices belonging to the same picture shall use the same values of *frame_num* and *pic_order_cnt_lsb*. The only difference is that *pic_order_cnt_lsb*'s increment is typically 2 instead of 1 [16]. Assuming that slices may be damaged but never lost, this behavior indicates that we only need to consider two outcomes: either the same values present in the previous slice are used, or the least significant bits of the next value in the monotonically increasing sequence are used. As in the case of $\hat{c}_{2,j}$, the conditional probability distribution of $\hat{c}_{3,j}$ and $\hat{c}_{4,j}$ varies based on previously reconstructed values:

$$P\left(\hat{c}_{3,j}\middle|\, c_2^*,c_1^*\right)=\begin{cases}\delta\left(\hat{c}_{3,j}-c_3^{(k-1)}\right) & c_1^*\neq 0\\ \delta\left(\hat{c}_{3,j}-lsb\left(c_3^{(k-1)}+1\right)\right) & c_1^*=0\end{cases} \tag{14}$$

$$P\left(\hat{c}_{4,j}\middle|\, c_3^*,c_2^*,c_1^*\right)=\begin{cases}\delta\left(\hat{c}_{4,j}-c_4^{(k-1)}\right) & c_1^*\neq 0\\ \delta\left(\hat{c}_{4,j}-lsb\left(c_4^{(k-1)}+2\right)\right) & c_1^*=0\end{cases} \tag{15}$$

where *lsb*(·) is the modulo operator. The divisors are derived from the SEs *log2_max_frame_num_minus4* and *log2_max_pic_order_cnt_lsb_minus4*, found in the active Sequence Parameter Set.

V. Experimental Results

The DVD-NTSC sequences *Driving*, *Opening ceremony*, and *Whale show* were coded using the JM 18.2 software [15]. The first 60 frames of each sequence were selected, where the first picture was coded as an IDR picture and the 31[st] picture was coded using only Intra slices. The other 58 pictures were coded using Inter slices. The packet size was limited to 100 bytes, so as to obtain slices with a variable number of MBs.

The decoder corrects the four SEs as described above, and uses the rest of the slice if it can. If an invalid codeword is encountered during the decoding of the remaining SEs (i.e.

motion vectors, residual coefficients, MB coding types, etc.), the MB containing the syntax error, as well as the remaining MBs in the slice, are discarded. The MBs successfully decoded (i.e. those containing only valid SEs) are reconstructed. When all the slices associated with a picture have been decoded, the missing MBs are concealed using the state-of-the-art error concealment method described in [13] with the following parameters: $\alpha = 0.5$, $\lambda = 0.1$, $\sigma_1^2 = 0.5$, $\sigma_2^2 = 4$, $T = 40$, $T_1 = 0.01$, and the diffusion process is limited to 30 iterations. Furthermore, the algorithm described in [14] is used to select the order in which the missing MBs are concealed.

To evaluate the performance of the proposed method, we used three different quantization parameters (QP), as these will affect the number of MBs per slice. Errors were introduced using a Gilbert-Elliott channel with a fixed bit error rate of 10^{-5} and three different average burst lengths (ABL), 2, 4 and 9, to make consecutive slices more or less likely to contain errors. The locations of the erroneous bits were selected with a Uniform Distribution, as we assumed that a bit interleaver was used to combat burst errors. For each combination of QP and ABL, 10 noisy sequences were generated. A total of 270 corrupted sequences were studied.

Table I presents both the average number of MBs and the standard deviation per slice type (Intra and Inter) for each combination of sequence and QP. The values increase significantly from QP 28 to QP 40, especially in the case of the *Opening ceremony* sequence. The effects of such a large standard deviation are apparent in Tables II, III, and IV. As expected, the average number of MBs per slice increases with increasing QP, reaching values between 18 and 88 for QP=40 in the case of Inter slices, which makes error correction very challenging.

Tables II, III, and IV list the statistical errors committed by the proposed approach applied at the SE level. Type I errors refer to the cases where the received value of *first_mb_in_slice* was actually correct, and the reconstructed value, after our correction method, was incorrect (i.e. the correction should not have changed the value). Type II errors refer to the cases where the received value was incorrect and remained unchanged after the correction step (i.e. the correction should have changed the value, but it didn't).

Table I. Statistics on MBs per slice

Sequence	QP	Intra average	Intra standard deviation	Inter average	Inter standard deviation
Driving	20	1.1	0.58	1.5	1.03
	28	2.0	1.55	5.1	4.76
	40	9.2	5.30	38.6	15.3
Opening ceremony	20	1.0	0.21	1.7	1.51
	28	1.3	1.06	8.0	14.2
	40	4.0	3.39	88.3	112.9
Whale show	20	1.1	0.31	1.1	0.43
	28	1.6	1.00	2.3	1.75
	40	6.7	4.16	18.0	14.28

The effect of a very large average and standard deviation (Table I) when a QP of 40 is used to compress the *Opening ceremony* sequence is apparent in Table IV. As the ABL increases, the probability of committing a type I error increases to nearly 50%. However, the number of type I statistical errors committed with the other sequences is very low, no matter what the conditions were.

Table II. Statistical errors for *first_mb_in_slice* (ABL = 2)

Sequence	QP	Corrupted slices	Type I errors	Type II errors
Driving	20	387	1	0
	28	171	6	1
	40	37	0	1
Opening ceremony	20	241	0	0
	28	90	2	2
	40	25	7	1
Whale show	20	365	0	1
	28	223	0	0
	40	42	4	0

Table III. Statistical errors for *first_mb_in_slice* (ABL = 4)

Sequence	QP	Corrupted slices	Type I errors	Type II errors
Driving	20	142	0	1
	28	148	10	2
	40	38	2	0
Opening ceremony	20	182	2	2
	28	144	9	0
	40	45	22	1
Whale show	20	89	0	0
	28	164	2	0
	40	42	5	1

Table IV. Statistical errors for *first_mb_in_slice* (ABL = 9)

Sequences	QP	Corrupted slices	Type I errors	Type II errors
Driving	20	113	0	1
	28	109	14	3
	40	102	4	0
Opening ceremony	20	198	25	2
	28	89	21	0
	40	77	36	1
Whale show	20	108	0	0
	28	135	0	0
	40	157	5	1

Fig. 2 presents the PSNR distribution of the first picture affected by transmission errors. The box plots appear in pairs. The first box plot represents the PSNR distribution when a state-of-the-art concealment method is used, combined with an optimal concealment order selection. The second box plot represents the PSNR distribution when our proposed method is used. Each row of box plots is associated with an ABL, while each column is associated with a video sequence.

The results show that a higher PSNR is expected with our method. Although there are statistical errors, the vast majority of the observations show improvements when error correction is applied first, as this reduces the area where error concealment is applied. Indeed, the error concealment method performs better when MBs are successfully repaired from corrupted slices. Over the three sequences tested, for an ABL value of 2 and a QP of 28, the average expected PSNR gains range from 0.75 dB to 2.22 dB with peaks at 3.67 dB. The worst loss observed (concealment performed better than our method) was of -0.32 dB.

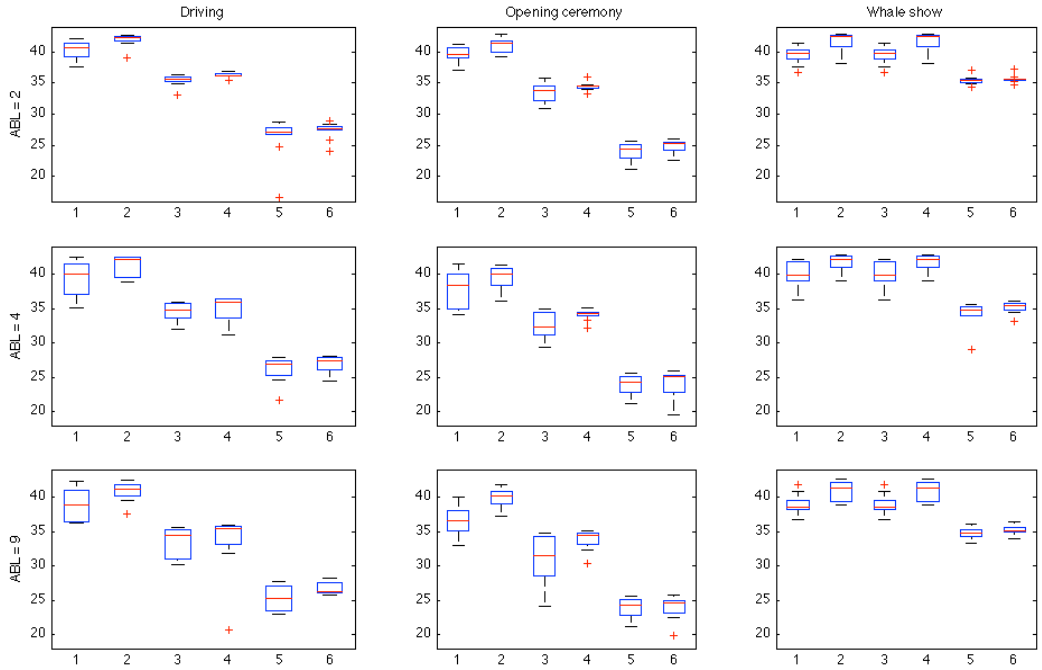

Figure 2 PSNR distributions using state-of-the-art error concealment [13][14] and our proposed method. Boxes 1, 3, and 5 correspond to state-of-the-art error concealment with QP=20, 28, and 40 respectively. Boxes 2, 4, and 6 correspond to the proposed error correction method followed by the same state-of-the-art error concealment method with QP=20, 28, and 40 respectively. The rows correspond to ABL values of 2, 4, and 9 respectively. The columns correspond to the video sequences *Driving*, *Opening ceremony*, and *Whale show* respectively.

VI. Conclusion

In this paper, we presented a novel maximum likelihood method for performing video error correction, both at the slice level and at the SE level. We have demonstrated that a breadth-first approach at the SE level using only four H.264 slice header SEs performed better than a state-of-the-art error concealment method at a BER of 10^{-5}. Not only were the PSNR results better, but they were obtained using significantly fewer computations. Future work will be aimed at modeling more SEs, as well as applying the method to the upcoming HEVC standard.

Acknowledgment

The authors thank Dr. Yan Chen for validating our spatio-temporal cost function implementation of [13].

References

[1] "Advanced Video Coding for Generic Audiovisual Services," ISO/IEC 14496-10 and ITU-T Recommendation H.264, Nov. 2007.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra. "Overview of the H.264/AVC video coding standard," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, p. 560-576, July 2003.

[3] S. Wenger. "H.264/AVC over IP," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no 7, p. 645-656, July 2003.

[4] L. A. Larzon, M. Degemark and S. Pink. "UDP lite for real time multimedia," IEEE Int. Conf. on Communications, June 1999.

[5] C. Weidman, P. Kadlec, O. Nemethova and A. Al Moghrabi. "Combined sequential decoding and error concealment of H.264 video," IEEE 6th Workshop on Multimedia Signal Processing, p. 299-302, Sept. 2004.

[6] Y. Wang and S. Yu. "Joint Source-Channel Decoding for H.264 Coded Video Stream," IEEE Trans. on Consumer Electronics, vol. 51, no 4, p. 1273-1276, Nov. 2005.

[7] G. Sabeva, S. Ben Jamaa, M. Kieffer and P. Duhamel. "Robust Decoding of H.264 Encoded Video Transmitted over Wireless Channels," IEEE 8th Workshop on Multimedia Signal Processing, p. 9-13, Oct. 2006.

[8] W. T. Lee, H. Chen, Y. Hwang and J. J. Chen. "Joint Source-Channel Decoder for H.264 Coded Video Employing Fuzzy Adaptive Method," IEEE Int. Conf. on Multimedia and Expo, p. 755-758, July 2007.

[9] D. Levine, W. E. Lynch and T. Le-Ngoc. "Iterative Joint Source-Channel Decoding of H.264 Compressed Video," IEEE Int. Symposium on Circuits and Systems, p. 1517-1520, May 2007.

[10] N. Q. Nguyen, W. E. Lynch and T. Le-Ngoc. "Iterative Joint Source-Channel Decoding for H.264 video transmission using virtual checking method at source decoder," 23rd Canadian Conf. on Electrical and Computer Engineering, p.1-4, May 2010.

[11] R. Farrugia and C. Debono. "Robust decoder-based error control strategy for recovery of H.264/AVC video content," IET Communications, vol. 5, no 13, p. 1928-1938, Sept. 2011.

[12] L. Trudeau, S. Coulombe and S. Pigeon. "Pixel domain referenceless visual degradation detection and error concealment for mobile video," 18th IEEE Int. Conf. on Image Processing, p. 2229-2232, Sept. 2011.

[13] Y. Chen, Y. Hu, O. Au, H. Li and C. W. Chen. "Video Error Concealment Using Spatio-Temporal Boundary Matching and Partial Differential Equation," IEEE Trans. on Multimedia, vol. 10, no 1, p. 2-15, Jan. 2008.

[14] X. Qian, G. Liu and H. Wang. "Recovering Connected Error Region Based on Adaptive Error Concealment Order Determination," IEEE Trans. on Multimedia, vol. 11, no 4, p. 683-695, June 2009.

[15] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG. "H.264/AVC JM reference software," 2011, http://iphome.hhi.de/suehring/tml/.

[16] J. B. Lee and H. Kalva. "The VC-1 and H.264 Video Compression Standards for Broadcast Video Services (Multimedia Systems and Applications)," Springer, Aug. 2008.

[17] W. Kwok and H. Sun. "Multi-directional interpolation for spatial error concealment," IEEE Trans. on Consumer Electronics, vol. 39, no 3, p. 455-460, June 1993.

[18] H. Sun and W. Kwok. "Concealment on Damaged Block Transform Coded Images Using Projections onto Convex Sets," IEEE Trans. on Image Processing, vol. 4, no 4, p. 470-477, Apr. 1995.