Error Concealment by Data Partitioning

Raj Talluri, Iole Moccagatta, Yashoda Nag and Gene Cheung

DSPS R & D Center, MS 8374
8330 LBJ Freeway
Dallas, Texas 75243

Abstract

This paper presents an error concealment strategy for improving the quality of compressed video data when transmitted over noisy communication channels. Data partitioning is used to enable the recovery of motion information when the compressed bitstream is corrupted by channels errors. At low bitrates the motion data is a significant part of the entire video stream and its recovery enables the decoder to perform motion compensated error concealment and hence maintain adequate video quality. The technique is shown to provide superior decode video quality over wide variety of channel errors, bit rates and test sequences. In order to enable effective error concealment the partitioned motion and texture data are separated a *motion marker*. The motion marker needs to be unique from nay valid combination of the motion VLC data. In this paper, we also present an efficient search algorithm to identify such a motion marker with good error resilience properties. This error concealment technique was proposed to the ISO MPEG4 standard and based on its performance, it has been accepted as part of the evolving standard definition.

1 Introduction

Current video compression techniques used in real time video communications [1, 2] achieve efficient compression by using predictive coding techniques such as motion compensation and also variable length entropy coding techniques such as Huffman codes. When the compressed video stream is transmitted over real world communication channels such as analog phone lines or wireless links to the decoder, it is often corrupted by channel noise and other channel degradations. Variable length coding schemes are highly susceptible to these errors introduced into the bitstream due to channel noise. As a result, the decoder loses synchronization with the encoder. Predictive coding techniques make matters much worse since the errors in one video frame quickly propagate across the entire video sequence, rapidly degrading the decoded video quality and rendering it totally unusable. Hence, unless the video decoder takes proper remedial steps, the video communication system totally breaks down.

2 Error Resilience Methodologies

In order to make the video codec more resilient to channel degradations the following stages are typically required at the decoder : 1) error correction 2) error detection and localization 3) resynchronization and 4) error concealment.

Forward Error Correction (FEC) codes such as Reed-Solomon codes, BCH codes etc. are employed by the encoder to protect the bit stream before transmitting to the decoder. At the decoder these codes are then used to correct errors in the bitstream due to the channel noise. FEC techniques prove to be quite effective against random bit errors, but their performance is usually not adequate against longer duration burst errors. These FEC techniques also come with an increased overhead in terms of the overall bitstream size and hence some of the gains achieved by the video compression are lost. Typically we apply FEC to provide a certain level of protection to the compressed bit stream and the residual errors are handled by the following stages.

Error detection techniques enable the video decoder to detect when a bit stream is corrupted by channel errors. In a typical block-based video compression technique that uses motion compensation and DCT, the following checks are applied to detect bit stream errors. 1) the motion vectors are out of range 2) an invalid VLC table entry is found, 3) the DCT coefficient is out of range, 4) the number of DCT coefficients in a block (8x 8) exceeds 64. When the decoder identifies any of these conditions in the process of decoding a video bitstream, it flags an error and jumps to the error handling procedure. Due to the nature of the video compression algorithms, the location in the bitstream where the decoder detects an error is not the same location where the error has actually occurred but some undetermined distance away from it. Hence once the decoder detects an error it

loses synchronization with the encoder. Resynchronization schemes are then employed for the decoder to fall back into lock step with the encoder. The encoder while constructing the bitstream inserts unique resynchronization words into the bitstream at approximately equally spaced intervals. These resynchronization words are chosen such that they are unique from the valid video bitstream. That is, no valid combination of the video algorithm's VLCs tables produce these words. The decoder upon detection of an error seeks forward in the bitstream hunting for this known resynchronization word. Once this word is found the decoder then falls back in synchronization with the encoder.

At this point, the decoder has detected an error, regained synchronization with the encoder and also isolated the error to be between the two resynchronization points. Typically in existing video coding techniques the data corresponding to the macroblocks between the two resynchronization points is discarded and the pixel values from the same macroblock locations in the previous frame are copied to conceal the effects of the erroneous macroblocks.

3 Optimal Resynchronization Word

From the above discussion it is clear that resynchronization is an essential part of the decoding strategy in the presence of channel errors. It is intuitively clear that the resynch word should be unique from any bit patterns of the bitstream itself; otherwise, part of the bitstream will be mistaken as resynch word during resynchronization and errors will continue to propagate. In this paper, we propose a search strategy that identifies these unique resynch words. Section 3.1 presents the optimal resynch word search strategy in detail. In Section 3.3, we discuss the theoretical aspects of our proposed search strategy and also present results of computer simulations are presented to verify the effectiveness of the resynch word.

3.1 Search Strategy

Since our objective is to identify resynch words that are unique from all possible bit patterns of the bitstream, a natural metric for comparing performance of potential resynch words is the minimum Hamming distance,$H_{\min}$,between the proposed resynch word and the bitstream. A resynch word that is unique from the bitstream is equivalent to one with $H_{\min} >= 1$ from the bitstream. For a given resynch word length R, we would like to identify the resynch word that has the largest $H_{\min}$ from the bitstream among 2^R potential resynch words. We continue to increment R until we find a resynch word with $H_{\min} >= 1$.

To find $H_{\min}$ between a potential resynch word and the bitstream, we need to consider all possible bit patterns of length R and compare them to the resynch word. Since we

assume the bitstream is composed of codewords from a set of VLC tables, the set of all possible bit patterns is embedded in a set of finite combinations of codewords. Our search strategy divides the search space into 3 subspaces: i) subspace containing single codewords which have individual lengths $\geq R$, ii) subspace containing combinations of 2 codewords that have combined lengths $\geq R$, iii) subspace containing combinations of 3 or more codewords such that each of the embedded codewords has length $< R$. Assuming that these 3 subspaces span the original space, (see section 7), we see that $H_{\min}$ of the proposed resynch word from the original space is the minimum of the 3 $H_{\min}$'s from the 3 subspaces. We will now discuss the procedures to find $H_{\min}$'s of each of the 3 individual subspaces.

3.1.1 One Search

In this subspace, we need to find $H_{\min}$ between the proposed resynch word and all possible bit patterns of length R embedded in single codewords. We will formally denote this subspace as:

$$S_1 = \{c_i^k | l(c_i^k) \geq l(r)\} \tag{1}$$

where $l(a)$ is a function that returns the length of the bit segment a; c_i^k denotes codeword i from VLC table k; and, r denote the candidate resynch word. So for each c_i^k in S_1, we do the following:

1. Initialize $H_{\min,c_i^k}$ to $l(r)$.
2. Let the total number of shifts, or the number of iterations needed to find $H_{\min}$ between c_i^k and r, be $N = l(c_i^k) - l(r) + 1$.
3. Define function $D(a, b)$ which returns the number of bit difference (Hamming distance) between bit pattern a and b, assuming $l(a) = l(b)$. Define $\hat{c}$ as a length $l(r)$ segment of c_i^k starting at bit N. Update $H_{\min,c_i^k}$ as:

$$H_{\min,c_i^k} = \min[H_{\min,c_i^k}, D(\hat{c}, r)] \tag{2}$$

4. Decrement N by 1. If $N > 0$, goto step 3.

After performing above procedure for all codewords with $l(c_i^k) \geq l(r)$, $H_{\min}$ for this subspace is the minimum of all $H_{\min,c_i^k}$'s in S_1.

3.1.2 Two Search

We will define this subspace as the following:

$$S_2 = \{c_i^k + c_j^m \quad | \quad l(c_i^k) + l(c_j^m) \geq l(r); \\ L(c_i^k, c_j^m) = 1\} \tag{3}$$

where $L(ab)$ returns 1 if ab is a permissible combination of codewords, 0 otherwise. [1] We will employ a procedure similar to One Search for combinations of codewords in S_2:

1. Initialize $H_{\min,(c_i^k+c_j^m)}$ to $l(r)$.
2. Let the number of shifts be $N = l(c_i^k) + l(c_j^m) - l(r) + 1$.
3. Define $\hat{c}$ as a length $l(r)$ segment of combination of (c_i^k, c_j^m) starting at bit N. Update $H_{\min,(c_i^k+c_j^m)}$ as:
$$H_{\min,(c_i^k+c_j^m)} = \min[H_{\min,(c_i^k+c_j^m)}, D(\hat{c}, r)] \tag{4}$$
4. Decrement N by 1. If $N > 0$, goto step 3.

$H_{\min}$ of the subspace is the minimum of all $H_{\min,(c_i^k+c_j^m)}$'s in S_2.

3.2 Multiple Search

This subspace is defined as:

$$S_3 = \{c_i^k + ... + c_j^m \quad | \quad \text{if } c_q^p \text{ is embedded,} \\ \text{then } l(c_q^p) < l(r); \\ L(c_i^k ... c_j^m) = 1\} \tag{5}$$

We first define a coherent block as a chosen codeword c_q^p from the VLC tables such that $l(c_q^p) < l(r)$. This is the center piece from which other codewords are concatenated to the left and right. For every coherent block in the VLC tables, we do the following:

1. Initialize $H_{\min,c_q^p}$ to be $l(r)$.

[1] Because of the predefined structure of ordering of codewords within bitstream, some combinations of codewords may be illegal. For example, in MPEG4 video coding algorithm [3, 4], a codeword that indicates no motion vectors in current macroblock cannot be followed by a motion vector codeword.

2. Let the number of shifts be $N = l(r) - l(c_q^p) + 1$, where c_q^p is a coherent block.

3. Define $\hat{r}_2$ as a length $l(c_q^p)$ segment of r starting at bit N. Define $\hat{r}_1$ as the first $N-1$ bits segment of r. Define $\hat{r}_3$ as the last $l(r) - l(c_q^p) - N + 1$ bits segment of r.

4. We need to find $H_{c_q^p,N}$, the minimum Hamming distance for the given coherent block c_q^p at the present shift N. To accomplish that, we need to find the combination of codewords around the coherent block that yields the smallest Hamming distance. This is done by recursion: (See Figure 1)

   (a) Left Recursion – recursively create a legal combination of codewords to the left of coherent block until its length $\geq N-1$. Define $\hat{c}_1$ as the last $N-1$ bits of this combination.

   (b) Right Recursion – for the given combination of codewords, (the present left combination and the coherent block), recursively create a legal combination of codewords to the right of coherent block until its length $\geq l(r) - l(c_q^p) - N + 1$. Define $\hat{c}_3$ as the first $l(r) - l(c_q^p) - N + 1$ bits of this combination.

   (c) Hamming distance of this combination is:

$$H^i = D(\hat{c_1}, \hat{r}_1) + D(c_q^p, \hat{r}_2) + D(\hat{c_3}, \hat{r}_3) \tag{6}$$

   After considering all possible combinations around the coherent block at present shift N, the minimum Hamming distance is the minimum of all the combinations:

$$H_{c_q^p,N} = \min[H^1, H^2, ...] \tag{7}$$

5. The minimum Hamming distance for the coherent block, c_q^p, is updated:

$$H_{\min,c_q^p} = \min[H_{\min,c_q^p}, H_{c_q^p,N}] \tag{8}$$

6. Decrement N by 1, If $N > 0$, goto step 3.

After performing the above procedure for all the coherent blocks in the VLC tables, the minimum Hamming distance for subspace S_3 is the minimum of all $H_{\min,c_q^p}$'s in S_3.

To show that the 3 subspaces span the search space, we need to show that any bit patterns of length R within the bitstream is a member of the 3 subspaces defined. Let a bit pattern P of length R, be a part of bitstream of a legal combination of codewords, $(c_i...c_j)$. Let Q be the number of codewords in the combination. Clearly if $Q = 1\text{or}2$, then $P \epsilon S_1$ or $P \epsilon S_2$ respectively; this follows from definition 1 and 3. If $Q \geq 3$, then $\forall$ embedded codewords, c_k's, $l(c_k) < R$. The reason is that if $l(c_k) \geq R$, then c_k cannot be embedded and Q can be at most 2. Therefore if $Q \geq 3$, then $P \epsilon S_3$. Therefore every bit pattern of length R must be a member of the 3 classes, and thus the spanning of search space.

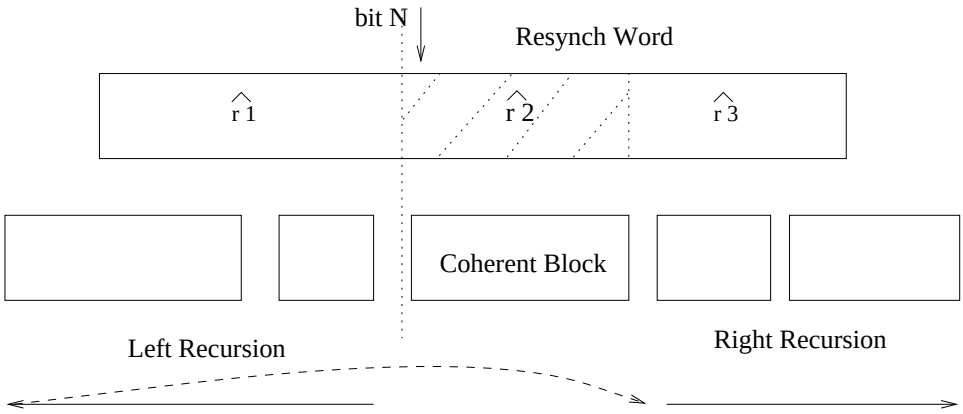

Figure 1: Multiple Search Procedure

codeword	bit pattern
c_1	1 0
c_2	0 1 1
c_3	0 1 0
c_4	0 0
r_o	1 1 1 1
r_r	1 0 1 0

Table 1: VLC Huffman Codewords and Resynch words

3.3 Resynch Word Analysis

Our search strategy relies on the assumption that there exists at least one resynch word with $H_{\min} = 1$ for the given set of VLC tables, for arbitrary large resynch word length R. It can be shown that if the VLC tables is a set of prefix-free code tables with at least one table being incomplete (at least one codeword entry not being used), then there exists a resynch word with $H_{\min} = 1$. In order to detect error in the erred bitstream, the VLC tables are frequently constructed to be incomplete; an error is detected when one of the unused codewords are found while decoding. Therefore, given the VLC tables are made incomplete, our search strategy will always output an optimal resynch word in finite time.

To test the effectiveness of the optimal resynch word, we compare the performance of the optimal resynch word, r_o, to a randomly generated resynch word of equal length, r_r. We construct an arbitrary incomplete Huffman table for testing. (See Table 1).

We first create a bitstream by generating 1.5 million codewords. Each codeword is chosen independently with probability $P(c_i) = \frac{2^{-l(c_i)}}{1-2^{-2}}$, where 2 is the length of the missing entry from the Huffman table. We insert the optimal resynch word for every 10 codewords for resynchronization at the decoder. We pass the bitstream through a binary symmetric

Resynch	error bits	error codewords	skipped bits
r_o	3188	15871	32566
r_r	3525	39705	32904

Table 2: Comparison between Optimal Resynch Word and Random Resynch Word

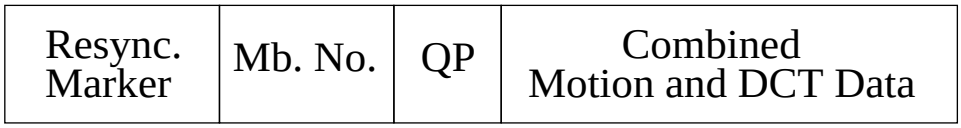

Figure 2: Traditional bitstream organization within the video packet

channel with bit error probability of 10^{-3}. When decoding the bitstream, we detect an error when the missing entry is found. We resynchronize for the next resynch word and continue decoding. Same procedure is performed for the randomly generated resynch word. Table 2 shows the statistics that are collected from the experiment. We see that for a similar number of error bits injected into the bitstream, the random resynch word results in more than twice the number of erroneously decoded codewords. We also see that the number of skipped bits, which are the number of bits decoder skips without decoding during resynchronization, is similar for both cases. In the optimal resynch word case, the decoder correctly identifies the resynch locations and error stops propagating. In the random resynch word case, the decoder often confuses existing bit sequence as resynch words and continue to decode at those locations. Error continues to propagate until missing Huffman entry is found again. So although the number of skipped bits is similar in both cases, the optimal resynch word has much better performance.

4 Error Concealment by Data Partitioning

As described in the Section 2, the amount of error concealment that can be accomplished in the existing video coding standards is limited to copying macroblocks from the previous frame. One of the main reasons for this is that between two resynchronization markers, the video data is coded in units of macroblocks. In addition, the motion and DCT data for each of the macroblock units is coded all together. Hence when the decoder detects an error, whether the error occurred in the motion part or the DCT part, all the data in the video packet (video data between the two resynchronization markers) needs to be discarded. Due to the uncertainty of the exact location where the error occurred, we cannot be sure that either the motion or the DCT data of any of the macroblocks in the packet is not erroneous. Figure 2 shows the organization of the video data within a packet for a typical video compression scheme. *The Mb No.* field denotes the Macroblock number of the first macroblock of the video packet within the frame. The *QP* field denotes the default

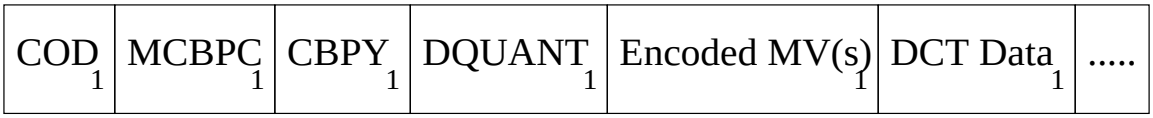

Figure 3: Bitstream components of each of the Macroblocks within the video packet

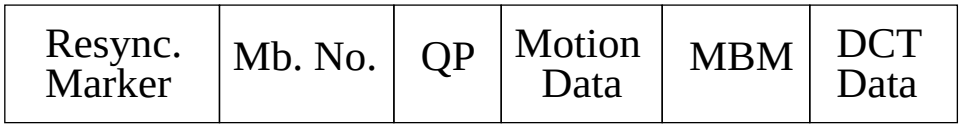

Figure 4: Bitstream organization with data partitioning for the motion and the DCT data

quantization parameter used to quantize the DCT coefficients in the video packet.

Within the *Combined Motion and DCT* part, each of the Macroblocks (MBs) motion vectors and the DCT coefficients are encoded. Figure 3 shows the syntactic elements for each of the MB. This data is repeated for all the MBs in the packet. The *COD* is a 1 bit field used to indicated whether a certain Macroblock is coded or not. The *MCBPC* is a variable length field and is used to indicate two things: 1) the mode of the MB such as INTRA, INTER, INTER4V (8 x 8 morion vectors), INTRA+Q (the quantization factor is modified for this MB from the previous MB) etc. 2) which of the 2 Chrominance blocks (8x8) of the MB (16x16) are coded. *DQUANT* is an optional two bit fixed length field used to indicate the incremental modification to the quantization value from the previous MB's quantization value or the default *QP* if this is the first MB of the video packet. *CBPY* is a VLC that indicates which of the 4 blocks of the MB are coded. *Encoded MV's* are the motion vector differences that are by a VLC. Note the motion vectors are predictively coded with respect to the neighboring motion vectors and hence we only code the motion vector differences. *DCT(s)* are the 64 DCT coefficients that are actually encoded via zig-zag scanning, run length encoded and then a VLC table. See [4, 3, 1] for more details.

Previous researchers have applied the idea of partitioning the data into higher and lower priority data in the context of ATM or other packetized networks to achieve better error resilience [5, 6]. However, this may not be possible over channels such as existing analog phone lines or wireless networks where it may be difficult, if not impossible, to prioritize the data being transmitted. Hence, it becomes necessary to resort to other error concealment techniques to mitigate the effects of channel errors.

We present below a method for achieving much better error concealment properties in the video coding schemes by the use of data partitioning. The basic idea is to partition the data within a video packet into a motion part and a texture part separated by a unique Motion Boundary Marker (MBM), as shown in Figure 4. Now the motion data for all the macroblocks in the video packet comes first followed by the texture data.

Figure 4 shows the bitstream organization within each of the video packets with data partitioning. Note that as compared to Figure 2 the Motion and the DCT parts are now

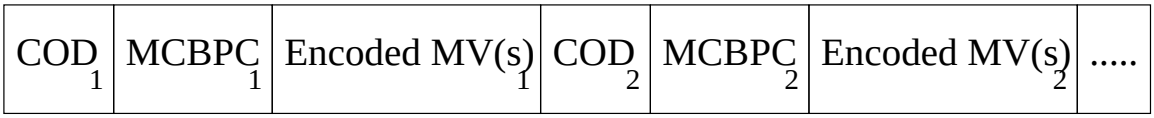

Figure 5: Bitstream components of the motion data.

Figure 6: Bitstream components of the DCT data.

separated by an MBM. In order to minimize the differences with respect to the conventional method, in this methods we maintain all the same syntactic elements as the conventional method and we organize them to enable data partitioning i.e., all the syntactic elements that have motion related information are placed in the motion partition and the all the syntactic elements that relate to the DCT data are placed in the DCT part. Figure 5 shows the bitstream elements after reorganization of the motion part and Figure 6 shows the DCT part. Note that we now place *COD*, *MCBPC* and the *MVs* in the motion part and relegate the *CBPY*, *DQUANT*, and the *DCTs* to the DCT part of the packet.

The MBM is computed from the motion VLC tables using the search program described above such that the word is Hamming distance 1 from any possible valid combination of the motion VLC tables. An example of one such word we used is **1111 1000 0000 00001**. It is 17-bits long. This word is uniquely decodable from the motion VLCs and gives the decoder knowledge of where to stop reading motion vectors before beginning to read texture information. The number of macroblocks(NMB) in the video packet is implicitly known after encountering the MBM. When an error is detected in the motion section, the decoder flags an error and replaces all the macroblocks in the current packet with skipped blocks until the next resynch marker. Resynchronization occurs at the next successfully read resynch marker. If any subsequent video packets are lost before resynchronization, those packets are replaced by skipped macroblocks as well. When an error is detected in the texture section (and no errors are detected in the motion section) the NMB motion vectors are used to perform motion compensation. The texture part of all the macroblocks is discarded and the decoder resynchronizes to the next resynch marker.

If no error is detected either in the motion or the texture sections of the bitstream but the resynch marker is not found at the end of decoding all the macroblocks of the current packet, an error is flagged and only the texture part of all the macroblocks in the current packet is discarded. Motion compensation is still applied for the NMB macroblocks as we have a higher confidence in the motion vectors since we got the MBM.

Let MB_B denote the number of the first macroblock in the next packet and MB_A denote the number of the first macroblock in the current video packet. An additional check can be performed in the case when no error is detected in either the motion or the texture

part of the packet and the next resynch marker is found correctly. In this case, we check if (MB_B - MB_A) is equal to NMB (computed from MBM). If not, we discard the data in the next video packet (packet corresponding to MB_B) since there is a high probability that MB_B is in error. The chance of an error in the current packet is small since we know that we decoded NMB motion vectors and DCT data correctly and found the resynch marker in the correct place. Contrast this with the case without data partitioning where when this error condition occurs, we have to discard both the current packet and the next packet since we can not rely on either of them.

Hence the two advantages of this data partitioning method are 1) we have a more stringent check on the validity of the motion data since we need to get the MBM at the end of the decoding of motion data in order for us to consider the motion data to be valid and 2) in case we have an undetected error in the motion and texture but don't end on the correct position for the next resynch marker, we do not need to discard all the motion data as we can salvage the motion data as it is validated by the detection of MBM.

5 Results

In this section we present results of testing the error resilient video codec with and without data partitioning.

The technique has been rigorously tested over a wide variety of test sequences, bitrates, and error conditions. In this paper we report simulation results obtained when testing a set of sequences in QCIF format [2] at 24 and 48Kbps. Resynchronization words are inserted into the bitstream at regular intervals. The test sequences, with corresponding coding bitrates and video packet lengths, are presented in Table 3.

Test Sequence	*Target Bitrate (kbps)*	*Packet Length (bits)*
Silent	24	480
Container Ship	24	480
Mother&Daughter	24	480
Foreman	48	736
Coastguard	48	736

Table 3: The test sequences and corresponding coding bitrates and video packet lengths used during the testing.

Because of the Motion Boundary Marker insertion in the video packet, the Data Partitioning technique results in a higher overall bitrate. The amount of overhead bitrate

[2] Frame size of 144×176 pixels, progressive scan, $4:2:0$ subsampling format.

depends from the frequency at which the video packets occur. The overhead bitrates resulting from our testing conditions are reported in Table 4.

Test Sequence	*Bitrate without Data Partitioning (Kbps)*	*Bitrate with Data Partitioning (Kbps)*	*Data Partitioning Overhead (%)*
Silent	25.88	26.57	2.65%
Container Ship	22.02	22.63	2.77%
Mother&Daughter	24.36	25.08	2.97%
Foreman	50.17	51.18	2.01%
Coastguard	45.55	46.50	2.10%

Table 4: Bitrates with and without data partitioning.

The compressed bitstream is corrupted using random bit errors, packet loss errors, and burst errors. A wide variety of random bit errors (BER of $10e^{-2}$, $10e^{-3}$), burst errors (of durations 1 msec, 10msec, and 20msec), and packet loss errors of varying lengths (96-400bits) and error rates ($10e^{-2}$, and $3 \times 10e^{-2}$) have been tested. To provide statistically significant results, 50 tests have been performed for each of the mentioned error conditions. In each test, errors are inserted in the bitstreams at different locations. This is achieved by changing the seed of the random number generators used to simulate the different error conditions.

For each test, the Peak Signal to Noise Ratio (PSNR) of the video decoded from the corrupted stream and the original video is computed. Then, the average PSNR of the 50 runs is computed for each frames in the sequence. To evaluate the performance of the proposed technique, the average PSNR values generated by the error resilient video codec with and without data partitioning are compared.

Figures 7 to 10 show the performance of the error resilient video codec with and without data partitioning when the compressed video stream of the test sequence "Coastguard" is corrupted by some of tested error conditions. For the same error conditions, Figures 11 to 14 show the performance of the error resilient video codec with and without data partitioning when the compressed video stream of the test sequence "Mother&Daughter" is corrupted.

The data partitioning approach achieves an average of 2 dB performance gain over the wide range of test sequences and error conditions. To confirm this conclusion, Table 5 to 12 summarize the performance of the error resiliente video codec with and without data partitioning for the entire set of test sequences and error conditions. Each entry in the table is the average of the PSNR (Luminance component only) of all the 100 frames across all the 50 runs. The superior error concealment that is possible due to the data partitioning showed consistently higher gains in performance over this entire range.

This data partitioning technique was proposed to the evolving ISO MPEG4 standard. The technique has been rigorously tested over a wide variety of error conditions, test

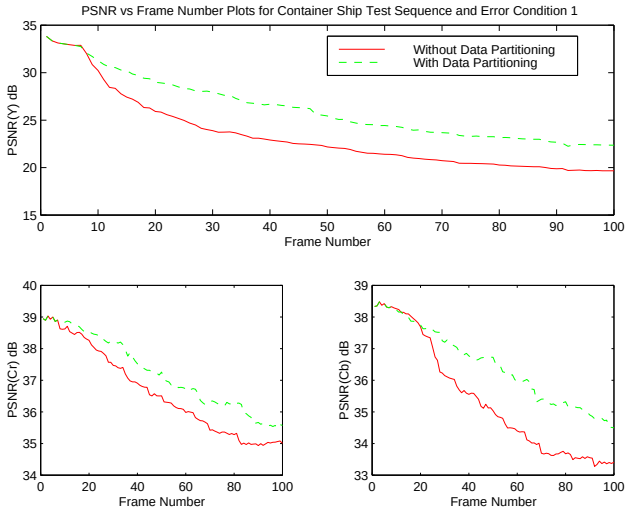

Figure 7: Performance of the error resilient video codec with and without data partitioning for Random Errors (BER $10e^{-3}$) on the test sequence "Container".

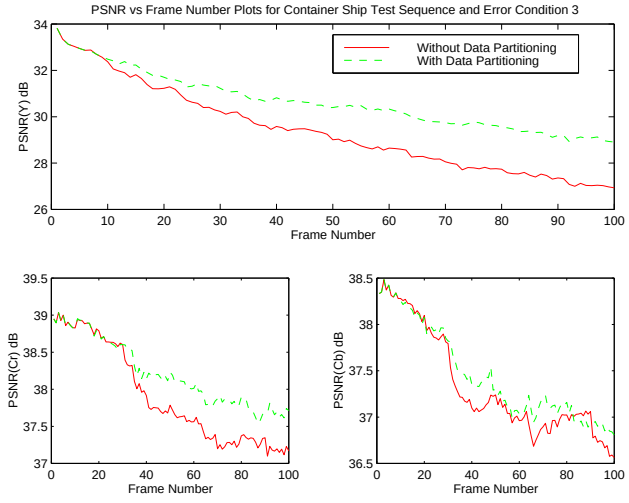

Figure 8: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-2}$, burst length 10ms) on the test sequence "Container".

Figure 9: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-3}$, burst length 1ms) on the test sequence "Container".

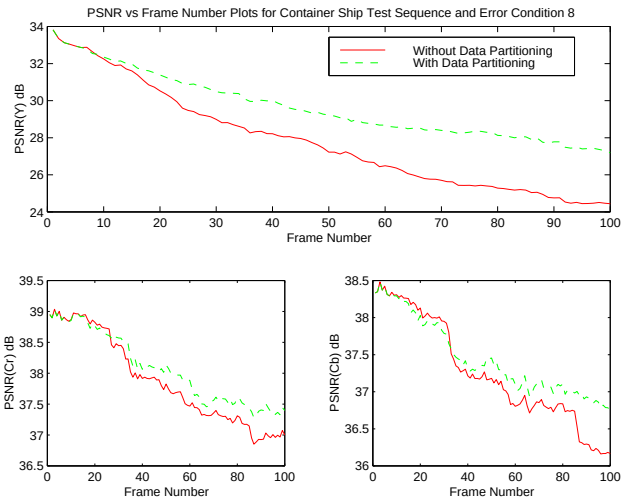

Figure 10: Performance of the error resilient video codec with and without data partitioning for Packet Loss (variable length of each packet 96-400 bits, packet loss rate $3*10e^{-2}$) on the test sequence "Container".

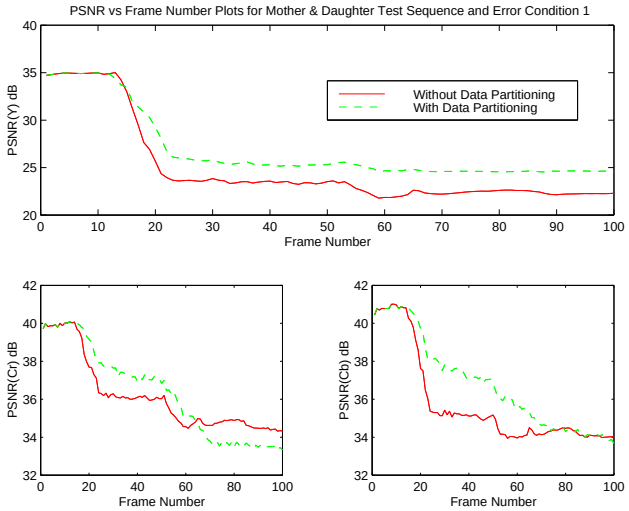

Figure 11: Performance of the error resilient video codec with and without data partitioning for Random Errors (BER $10e^{-3}$) on the test sequence "Mother&Daughter".

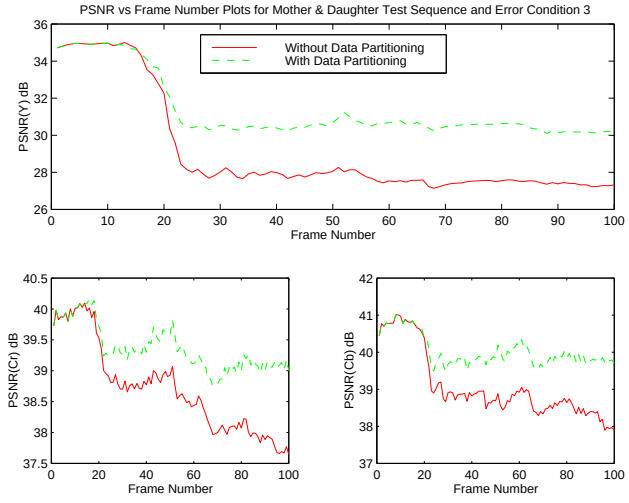

Figure 12: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-2}$, burst length 10ms) on the test sequence "Mother&Daughter".

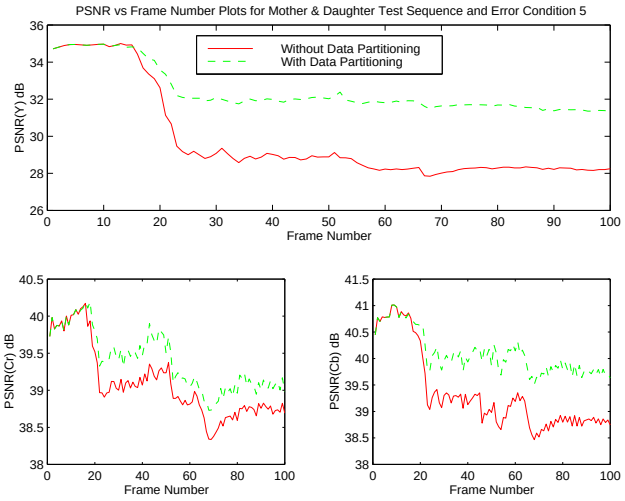

Figure 13: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-3}$, burst length 1ms) on the test sequence "Mother&Daughter".

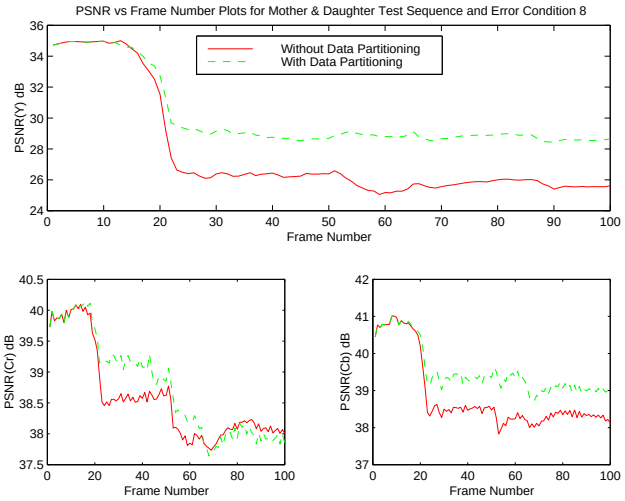

Figure 14: Performance of the error resilient video codec with and without data partitioning for Packet Loss (variable length of each packet 96-400 bits, packet loss rate $3*10e^{-2}$) on the test sequence "Mother&Daughter".

sequences and bitrates [7]. The results have been independently confirmed by two different parties and based on this, the technique is now accepted as part of the evolving ISO MPEG4 standard [3, 4].

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning (dB)*
Silent	24.81	25.92
Container Ship	25.58	28.30
Mother&Daughter	27.21	29.17
Foreman	19.34	20.19
Coastguard	20.74	22.02

Table 5: Performance of the error resilient video codec with and without data partitioning for Random Errors (BER $10e^{-3}$).

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning (dB)*
Silent	24.51	25.58
Container Ship	25.02	27.66
Mother&Daughter	27.05	28.51
Foreman	20.45	20.85
Coastguard	21.55	22.58

Table 6: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-2}$, burst length 1ms).

6 Conclusions

In this paper we presented a data partitioning approach that results in much improved error concealment when transmitting compressed video over noisy communication channels. We presented a bitstream organization and a syntax that maintains the compatibility of the existing standards in terms of using the same VLC tables and syntactic elements but achieves the gains of the data partitioning by a simple and efficient reordering of the bitstream components. We also presented the need for resynchronization markers to delineate the boundary between the motion and the texture parts. The resynch markers need to be unique from the valid bitstream data and at the same time have superior error resilience properties. We presented an efficient search program that generates these resynch marker in a efficient manner from a given set of VLC tables. We also exhaustively tested this data partitioning approach with our resynch markers across a wide range of bit rates, sequences and channel error conditions¿ we showed that on the average for a very small increase in

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning (dB)*
Silent	28.77	29.17
Container Ship	30.83	32.03
Mother&Daughter	32.00	32.28
Foreman	25.18	24.46
Coastguard	26.06	26.32

Table 7: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-2}$, burst length 10ms).

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning (dB)*
Silent	29.74	29.88
Container Ship	31.22	32.56
Mother&Daughter	32.75	33.12
Foreman	26.65	26.01
Coastguard	27.46	27.50

Table 8: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-2}$, burst length 20ms).

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning (dB)*
Silent	29.49	30.23
Container Ship	31.86	32.80
Mother&Daughter	32.96	34.06
Foreman	26.86	27.26
Coastguard	27.00	28.65

Table 9: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-3}$, burst length 1ms).

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning (dB)*
Silent	31.76	31.76
Container Ship	33.46	33.60
Mother&Daughter	35.54	35.69
Foreman	31.03	30.86
Coastguard	29.96	29.96

Table 10: Performance of the error resilient video codec with and without data partitioning for Burst Errors (BER $10e^{-3}$, burst length 10ms).

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning with (dB)*
Silent	29.75	29.98
Container Ship	31.83	32.66
Mother&Daughter	32.95	33.37
Foreman	25.23	25.35
Coastguard	25.86	26.78

Table 11: Performance of the error resilient video codec with and without data partitioning for Packet Loss (variable length of each packet 96-400 bits, packet loss rate $1*10e^{-2}$).

Test Sequence	*Average PSNR without Data Partitioning (dB)*	*Average PSNR with Data Partitioning (dB)*
Silent	27.50	27.87
Container Ship	29.32	30.51
Mother&Daughter	30.29	31.16
Foreman	21.98	22.37
Coastguard	23.22	23.92

Table 12: Performance of the error resilient video codec with and without data partitioning for Packet Loss (variable length of each packet 96-400 bits, packet loss rate $3*10e^{-2}$).

overhead (2-3%) the data partitioning approach improves the quality of the compressed video data by over 2 dB. Based on this performance, this technology is now accepted as part of the evolving ISO MPEG4 standard.

7 Appendix

To show that the 3 subspaces span the search space, we need to show that any bit patterns of length R within the bitstream is a member of the 3 subspaces defined. Let a bit pattern P of length R, be a part of bitstream of a legal combination of codewords, $(c_i...c_j)$. Let Q be the number of codewords in the combination. Clearly if $Q = 1or2$, then $P \epsilon S_1$ or $P \epsilon S_2$ respectively; this follows from definition 1 and 3. If $Q \geq 3$, then $\forall$ embedded codewords, c_k's, $l(c_k) < R$. The reason is that if $l(c_k) \geq R$, then c_k cannot be embedded and Q can be at most 2. Therefore if $Q \geq 3$, then $P \epsilon S_3$. Therefore every bit pattern of length R must be a member of the 3 classes, and thus the spanning of search space.

References

[1] "Video coding for low bitrate communication, draft recommendation H.263," *ITU-T, Study Group 15*, October 1995.

[2] K. Rijkse, "ITU standardization of very low bitrate video coding algorithms," *Signal Processing: Image Communication*, vol. 7, pp. 553–565, November 1995.

[3] ISO/IEC/JTC1/SC29/WG11 14496-2, "MPEG-4 Video Verification Model Version 8.0," Tech. Rep. M 1796, July 1997.

[4] ISO/IEC/JTC1/SC29/WG11 14496-2, "MPEG-4 Visual Working Draft Version 4.0," Tech. Rep. M 1797, July 1997.

[5] M. Ghanbari, "Two-layer coding of video signals for vbr networks," *IEEE Journal Selected Areas in Communications*, vol. 7, pp. 771–781, June 1989.

[6] M. Nomura, T. Fujii, and N. Ohta, "Layered packet-loss protection for variable rate video coding using dct," in *Proceedings International Workshop on Packet Video*, September 1988.

[7] ISO/IEC JTC1/SC2/WG11 Ad hoc Group on Core Experiments on Error Resilience Aspects of MPEG-4 Video, "Description of Error Resilinet Core Experiments," Tech. Rep. N 1473, November 1996.