

RESILIENT VIDEO CODING FOR NOISY CHANNELS

Robert Swann

Cambridge Consultants Limited, Science Park, Milton Road, Cambridge CB4 4DW, UK.

ABSTRACT

In this tutorial paper, we consider the effect of errors in compressed video data, and discuss both standard and more novel techniques for increasing error resilience. We consider the performance of block based coding schemes such as MPEG1, MPEG2, and h.26x when transmitted over noisy channels, a subject of relevance to digital terrestrial television, video communication, mobile digital video, and video storage.

INTRODUCTION

Block based video coding schemes such as MPEG and h.26x Achieve compression by removing both spatial and temporal redundancy. They operate by dividing each picture into 8x8 blocks. Blocks are encoded using the discrete cosine transform (DCT) followed by quantisation to reduce the number of coefficients. These coefficients are subsequently coded. Pictures are divided into intra (I) and predicted (P/B) pictures. The predicted pictures are coded using motion compensation from a previous picture.

One error in a block coded video sequence can cause large portions of the picture to be corrupted. The purpose of this paper is to highlight some of the issues in the transmission of video over noisy channels, and to discuss some of the remedies.

ERROR CONTROL

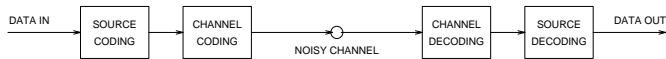


Fig. 1. Source and channel coding

Traditionally, *source coding* and *channel coding* have been separated. Figure 1 shows the coding of digital data. In the source coding stage, the data is compressed, and as much uncontrolled redundancy as possible is removed. In the channel coding stage, controlled redundancy is put back in to allow error detection and correction of errors at the channel decoder.

Bitrate Quality Tradeoff

If video is passed over a noisy channel of capacity C , then the number of databits N and the number of controlled redundancy checkbits, R must be fewer than C . $N+R \leq C$. The picture quality in the absence of errors is a function of N alone. This is illustrated by Figure 2. The standard MPEG2 curve (-) shows no bits allocated to forward error correction (FEC), thus $R=0$. The three curves (--), (-.), and (..) show MPEG2 with the addition of FEC. This is the result of trading a few databits (N) for FEC (R). With FEC a lower quality is noticed at low error rates. However, a higher quality is achieved at high error rates. FEC also produces a much sharper curve, where the picture suddenly deteriorates very quickly with increasing error rate. This occurs when the error correcting capability of the code has been exceeded, and the code attempts to correct multiple errors incorrectly. Thus

video protected by FEC degrades suddenly with little notice, whereas unprotected video degrades more gracefully.

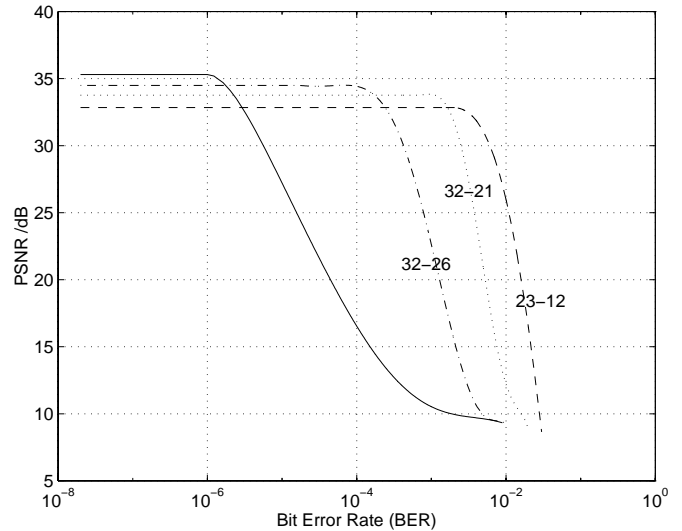


Fig. 2. The error performance of standard MPEG2 (-). MPEG2 with the Golay (23,12) triple error correcting ($t=3$) code (--). MPEG2 with the augmented BCH (32,21) double error correcting code (..). MPEG2 with the augmented Hamming (32,26) single error correcting code (-.).

Scalable Coding

Scalable coding is a popular technique for achieving robust video delivery. Scalable coding divides a signal across more than one channel of differing priorities. Typically a high priority signal or base layer is transmitted which provides a low quality picture. The higher, or enhancement, layers are sent to add extra detail. The base layer is heavily protected, whereas the upper layers are not. Error resilience is achieved during bursts of high error rate by protecting the base layer at the expense of enhancement layers. Scalable coding therefore allows:

- Compatibility with existing formats – e.g. a TV signal could be transmitted as the base layer of an HDTV system.
- Graceful degradation – if a transmission becomes noisy, the decoder will switch to the lower quality base level
- Low resolution preview – this is useful during fast forward, when the decoder may not be powerful enough to decode a full picture
- Differing quality of service (QOS) for different customers
- Concealment of transmission errors – it can be possible to use information from all the scalable levels to assist the detection and concealment of transmission errors.

There are various flavours of scalable coding which are used:

- Frequency scalability – where transform coefficients from the DCT are split into groups. The low frequency coefficients form the base layer, and the other groups are

enhancement layers. At the decoder, the coefficients are combined before reversing the transform (IDCT). One of the problems with frequency scalability is that it suffers from drift because predictions are not constructed the same way in the decoder and in the encoder.

- Spatial scalability is where the picture is first downsampled to form a low pass image, which is coded to form a base layer. The difference between the original picture and the downsampled picture is encoded to form the enhancement layer. This form of scalability does not suffer from drift.
- Signal to Noise scalability is where a base layer is formed using coarse quantisation. Successive enhancement layers quantise the difference more finely. SNR scalability does suffer from drift, but it can be limited.
- Temporal scalability is where groups of pictures are partitioned into a base layer containing I and P pictures, and an enhancement layer of B pictures. As errors in B pictures do not propagate, the B picture stream can be considered to be of lower priority.
- Chroma simulcast can be used to send different qualities of chroma information in different levels. For example, the 4:2:0 colour format could form the lower layer, and the extra information for 4:2:2 sent as a higher layer.

Data Partitioning

Data Partitioning is similar to scalable coding. The stream is divided into two or more channels of differing priority. All data is sent down a high priority channel until a breakpoint is reached, when all following data is transmitted in the lower priority channel. This means that information such as block positions and encoding type parameters may be assigned a higher priority than the data specifically about the contents of a block.

Automatic Repeat Request

Often a reply channel is available in image communication. In this case an error detecting code can be used. If an error is detected, the decoder simply requests a repeat transmission of the corrupted portion. This system is used in internet communication, yet it is not suited to broadcast video for two reasons. Firstly, a back channel is not always available. Secondly, by the time the repeat request has been dealt with, and the corrupted portion retransmitted, the video picture will have missed its presentation time – thus repeat data will often arrive too late.

ARQ is often used for concealment. Even if a repeat transmission arrives after the picture presentation time, the data can still be of use, as it allows the decoder to update the reference image, such that the decoder and encoder regain synchronisation, and errors do not propagate for many frames.

LOSS OF BITSTREAM SYNCHRONISATION

Once an error occurs in the bitstream, it often causes the decoder to lose bitstream synchronisation. Once synchronisation is lost, all the remaining data is misinterpreted. This point can be illustrated by the following example:

Suppose we have four possible events, A,B,C, and D. Event A is the most probable, and C and D are less probable. To

achieve compression. We assign a short code to A and longer codes to B and C.

Event	A	B	C	D
Code	0	11	101	100

Table 1. A Variable Length Code

If we now consider a sequence of events, [A,B,A,B,A,A]. This is coded as: [0 1 1 0 1 1 0 0]. If we now consider one single error in the second bit, the received signal is [0 0 1 0 1 1 0 0]. The received signal is interpreted as [A,A,C,D]. Not only has one error corrupted everything after it, but the received code contains a different number of symbols (events) from the transmitted code. This means that the decoder has lost symbol-synchronisation, and all the following data is liable to be of no use.

RESYNCHRONISATION

The key to robust video is to ensure that resynchronisation is achieved as quickly as possible following an error event.

Fixed Length Codes

Using fixed length codes (FLC) means that single bit errors will not cause loss of synchronisation. Vector Quantisation (VQ) is a compression technique which can produce fixed length codes. NxN blocks of image are treated as vectors of length N^2 . Scalar quantisers take each element of the vector and independently compare it to the decision levels of the quantiser. VQ systems, however, take the whole vector, and compare it to a set number of reference vectors. An index is then transmitted to identify which of the reference vectors gave the best match. If the number of reference vectors is n , then a fixed length code of $\log_2 n$ is required to store each index. Compression is then achieved by carefully choosing each reference vector to be equiprobable, and choosing an index length so that it is less than the number of bits required to send the block as PCM.

Although VQ is inherently resilient to transmission errors, it is a lossy scheme so it would not be possible to transcode losslessly from MPEG and h26x to VQ and back again.

Resynchronising codewords

Synchronisation can easily be guaranteed by the regular insertion of a universal synchronising sequence. This unique sequence may only occur at certain points in the bitstream. If a synchronising sequence is decoded, the decoder can guarantee to regain bitstream synchronisation.

As these synchronising sequences tend to be long (in MPEG2 it is 24 bits long) they should not be transmitted too often. The inclusion of frequent resynchronising codewords adds significant redundancy

Bit Shift

Bit-shift is a simple technique to cope with a single error burst between two resynchronising codewords. The moment loss of synchronisation is detected, the decoder stops, shifts a bit, and continues decoding. This process is repeated until a correct bitstream is found. Once resynchronisation has occurred, concealment algorithms can be used.

Bi-directional codes

With careful design of the set of variable length codes, it is possible to design a code which may be decoded either forwards or backwards. Following an error, the decoder skips to the next resynchronising point, and then decodes backwards from this point. These codes are not optimal in terms of compression, and they can only correct one error between synchronisation points.

Statistical Resynchronisation

A binary prefix code such as a Huffman code, is said to be self-synchronised if it contains synchronising codewords. A synchronising codeword has the property that after the decoder receives this codeword, it will be able to resynchronise regardless of any preceding slippage.

Consider a prefix-condition source code, ξ . A codeword σ of ξ is a synchronising codeword, and hence ξ is a self-synchronising code, if it satisfies the following conditions:

- If $\sigma = (\alpha\beta)$ where $\alpha \neq \sigma$ is a suffix of some codeword of ξ , then β is a string of codewords of ξ
- If σ is a substring of a string of codewords τ of ξ , then σ is a suffix of τ .

Symbol	Prob	Synchronous code ξ_1	Non-Synchronous code ξ_2
A	0.28	00	10
B	0.26	10	11
C	0.13	010	010
D	0.12	110	011
E	0.06	0110	0010
F	0.05	0111	0011
G	0.05	1110	0001
H	0.05	1111	0000

Table 3. A synchronous code

Both codes ξ_1 , ξ_2 , have the same length vector, and hence identical efficiencies. ξ_2 could be generated using the Huffman algorithm, whereas ξ_1 could not. In code ξ_1 , the two codewords $\sigma_1 = 010$ and $\sigma_2 = 0110$ are synchronising codewords. This is because $010 = 0.10$ and $0110 = 0.110$ where 10 and 110 are valid codewords. Code ξ_2 has no synchronising codewords.

The probability of codeword σ_1 occurring, $p_1 = p_C + 0.5(p_B(p_A + p_B + p_D + p_G)) = 0.2223$. Similarly, the probability of σ_2 occurring is 0.1026. Hence the resynchronisation probability is $0.2223 + 0.1026 = 0.3249$, so the code should resynchronise within about three symbols.

The AC coefficients in MPEG and h.26x are encoded using variable length codes. Errors in these AC coefficients can lead to the following cases:

1. The error propagates and reduces the number of received blocks in a line, so that $n_{received} < n_{expected}$. Here it can be assumed that an error occurred which merged two blocks together into one. All the blocks decoded after the error will be correct but shifted by one block to the left.
2. The error propagates such that $n_{received} > n_{expected}$. The assumption is that one block has been split into two, following a corrupted end of block (EOB) code. All the blocks decoded after the error will be correct but shifted by one block to the right.

3. The error does not propagate, affects only one block, and no shift occurs.
4. The error propagates, but $n_{received} = n_{expected}$. This is an extremely rare occurrence.

Concealment can be achieved by decoding the picture slice at a time. At the end of each slice, $n_{received}$ is compared with $n_{expected}$. Calculations are performed to find the positions of the block shift, and then conceal any missing blocks.

Comma Codes

Some codes guarantee synchronisation. One example is the comma code where each codeword (except the last) is a zero preceded by a different number of ones. E.g. A=0, B=10, C=110, D=1110, E=1111. It can be seen that a single bit inversion will not cause synchronisation to be lost for long. For example, if the code BCAAE [10110001111] is subjected to a single bit error in the first bit [00110001111], then the decoded output would be AACAAE. Thus the code has recovered in two symbols.

Despite this property, Comma codes are not widely used, as they are suboptimal in all cases where the event probabilities do not fit the distribution.

Error Resilient Entropy Code

The Error Resilient Entropy Code (EREC) is a method for coding variable length blocks of data with low overhead and with high resilience to transmission errors. The EREC effectively reorders the data so that each block starts at a known position in the bitstream. The EREC fills data into the spaces such that the data from the longer blocks fill up the spaces left by the shorter blocks.

Figure 4a shows 10 variable length blocks of data. A slot structure is generated where the height of each slot is equal to the average block length. Initially as much data as possible is fitted into the slot structure. Any data which overfills the slot structure is retained (Figure 3b). The slots are moved to the left, and the data drops into any spaces available (Figure 3c). This process continues until all the databits have been placed (Figure 3d).

The decoding stage is similar to the encoding process except it operates in reverse. In order to successfully decode the EREC the number of slots and the slot heights must be known. These two parameters are often known (i.e. they are a function of picture size), but if not, they must be transmitted over a highly protected channel. This overhead can be managed to be very small.

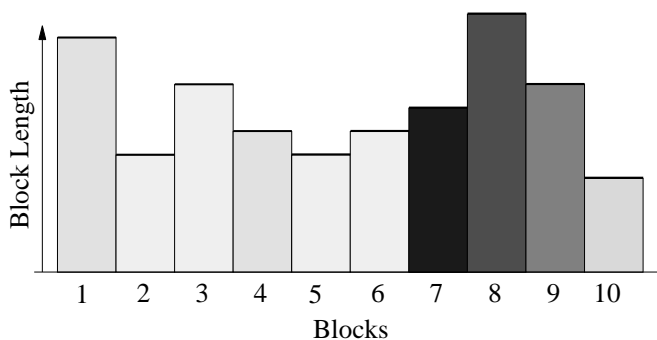


Fig. 3a Variable Length Blocks

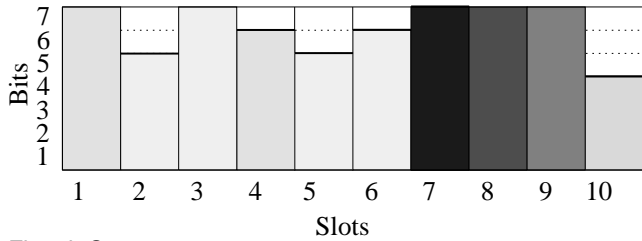
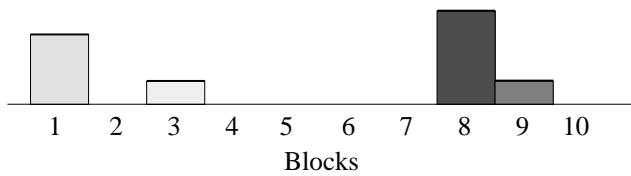


Fig. 3b Stage 1

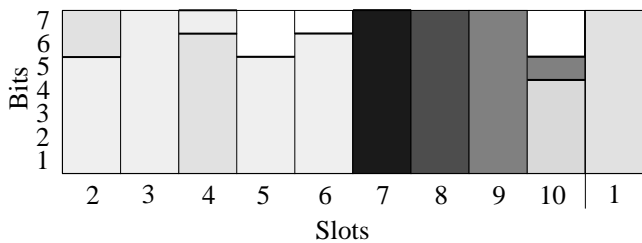
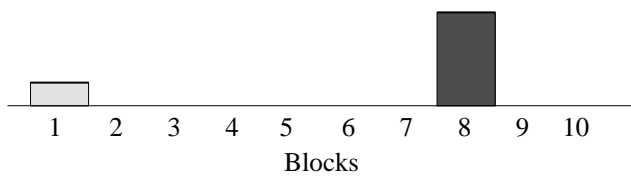


Fig. 3c Stage 2

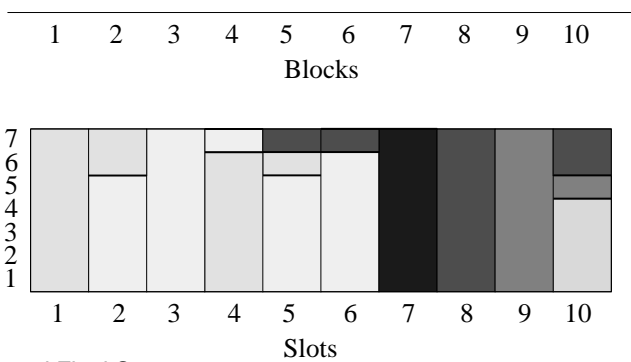


Fig. 3d Final Stage

RESILIENT DIFFERENTIAL CODING

In MPEG and h.263, adjacent DC coefficients and motion vectors are very similar, as are contiguous pictures. Compression is achieved by coding these coefficients relative to previous ones. If the previous coefficient, from which the prediction is to be made, is corrupted, then all the subsequent data will also be corrupted.

Differential Pulse Coded Modulation

Differential Pulse Coded Modulation (DPCM) is one of the simplest techniques used for lossless data compression. In DPCM, each coefficient is coded as an offset on the last coefficient. This offset follows a laplacian distribution for DC

coefficients, and so this difference can be variable length coded. MPEG1 and MPEG2 use DPCM.

Enhanced error resilience can be achieved using a different type of predictor. 2D-DPCM operates by taking the average of the coefficient to the left and the coefficient above. Errors appear as patches which fade away.

Using non-linear predictors such as median filters can achieve even higher error resilience due to the inherent rejection of outliers.

In addition to using different types of predictor (eg average/median/weighted median), resilience can be enhanced by changing the predictor topology. In particular, a quincunx type predictor can greatly reduce the amount by which errors propagate.

CONCLUSIONS

Using some of the techniques outlined in this paper can greatly enhance the error resilience of transmitted video. Although some techniques require the addition of controlled redundancy, many do not, or only add a very minimal overhead.

To illustrate the effectiveness of resilience schemes, Figure 4 shows an intra video picture which was coded with MPEG2 at 5Mb/s. The sequence is subjected to a 0.1% bit error rate. Using techniques to force regular resynchronisation, and techniques to limit the propagation of errors in differentially coded data, the effect of the errors can be substantially reduced, without the addition of extra redundancy.



Fig. 4a: The original video image



Fig. 4b: MPEG2 Coded - BER = 0.1%

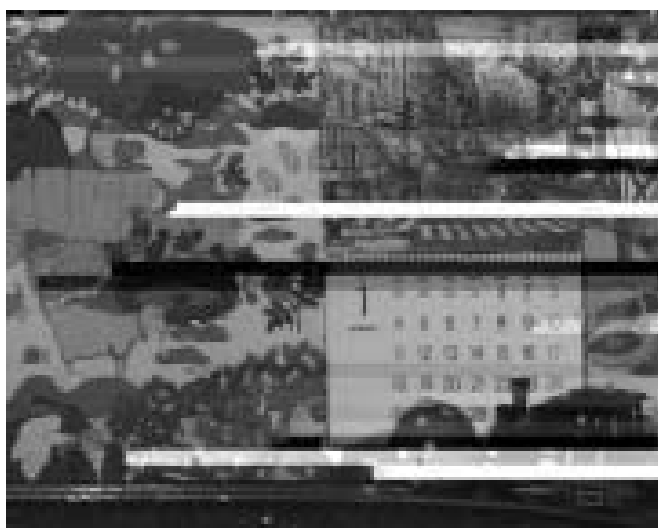


Fig. 4c: MPEG2 Coded - BER = 0.1%. In addition resynchronisation has been achieved using the EREC. No extra bits have been added



Fig. 4d: MPEG2 Coded - BER = 0.1%. In addition resynchronisation has been achieved using the EREC. The differential DC information has been coded using a quincunx median predictor. No extra bits have been added

REFERENCES

Error Resilient Transmission of MPEG-II over Noisy Wireless ATM Networks"

R E M Swann and N. G. Kingsbury

In IEEE Proceedings of the International Conference on Image Processing, Santa Barbara. October 1997.

Bandwidth Efficient Transmission of MPEG-II video over noisy mobile links

R E M Swann and N G Kingsbury

In Signal Processing : Image Communication Special Issue on Mobile Image / Video Transmission, Vol.12. No. 2 April 1998 pp.105-115

O.A. Aho and J.Juhola.

Error resilience techniques for MPEG-2 compressed video signals.

In Proceedings of the International Broadcasting Convention. IEE, September 1994.

S.H. Lee, J.S. Youn, S.H. Jang, and S.H. Jang.

Transmission error detection, resynchronisation and error concealment for {MPEG} video decoders. Proceedings of the SPIE, 2094:195--204, November 1993.

J.E. Fowler, M.R. Carbonara, and S.Ahalt.

Image coding using differential VQ.

IEEE Transactions on Circuits and Systems for Video Technology, 3(5), October 1993.

M.Ghanbari and V.Seferidis.

Cell loss concealment in ATM video codecs.

IEEE Transactions on Circuits and Systems for Video Technology, 3:238--247, 1993.

D.W. Redmill and D.Bull.

Error resilient arithmetic coding of still images.

Proceedings of the International Conference on Image Processing, pages II.109--112, September 1996.

S.Chan and A.Leon-Garcia.

Block loss for ATM video.

In Proceedings of the SPIE, 2094:1701--1713, 1993.

W.M. Lam and A.R. Reibman.

Self synchronising variable length codes for image transmission.

In ICASSP 92, pages C477--480, March 1992.

Huifang Sun and Wilson Kwok.

Concealment of damaged block transform coded images using projections onto convex sets.

IEEE Transactions Image Processing, 4(4):470--477, April 1995.