Error Resilience in Video and Multiplexing Layers for Very Low Bit-rate Video Coding Systems

Wee Sun Lee, *Member, IEEE,* Mark R. Pickering, *Member, IEEE,*
Michael R. Frater, *Member, IEEE,* John F. Arnold, *Senior Member, IEEE*

Abstract— **The transmission of audio-visual services on low bit-rate, wireless telecommunications systems requires the use of coding techniques that are both efficient in their use of bits and robust against errors introduced in transmission. In this paper, we present efficient techniques for improving the error resilience of audio-visual services. These techniques are based on coding simultaneously for synchronization and error protection or detection. We apply the techniques to improve the performance of the multiplexing protocol (which combines the video and audio streams so that they can be transmitted on a single circuit) and also to improve the robustness of the coded video. We show through simulations that the techniques are efficient in their use of bits and effective against bursty errors common in wireless channels. For a simulation of the DECT channel at a bit error rate of 10^{-3}, the techniques give an order of magnitude improvement in the probability of lost packets in the multiplexer layer over more conventional techniques. In the video layer, the techniques give an improvement of between 1 to 2 dB over ITU-T Recommendation H.263. The techniques proposed for the video layer also have the advantage of permitting simple transcoding with bitstreams complying with H.263.**

KeyWords— **Error Resilience, multiplexing protocols, robust video coding**

I. Introduction

Error resilient coding is of great importance when transmitting information which can tolerate a limited amount of error such as audio and video over many low bit-rate channels, especially those used for wireless telephony such as GSM [1] and DECT [2]. Because of the limited capacity of the channels, it may not be feasible to uniformly protect the information with error correcting codes without sacrificing the quality significantly. Instead, it is preferable that the error protection be added to the bit-stream in such a way that the most important information is given the greatest protection. The error protection should also allow for a graceful degradation of transmission quality as the errors in the channel increase, rather than a complete loss of information if a vital part of the received data is in error. For low bit-rate channels, the efficiency of the error protection can be improved if the same error protection data can be used to achieve both of these desired functions.

Error resilience schemes can be roughly classified into the following three categories:

This work was supported by the Australian Research Council. Part of the work was presented at the Australian Telecommunication Networks and Applications Conference, Melbourne, December 1996

The authors are with the School of Electrical Engineering, University College, The University of New South Wales, Australian Defence Force Academy, Canberra ACT 2600, Australia.

Resynchronization and Localization

Channel errors can cause loss of synchronization at the receiver. The loss of synchronization may be caused by insertion and deletion losses due to instability of the clock. Bit errors can also cause loss of synchronization when variable length codes are used. When compression schemes with memory are used, errors caused by synchronization loss are compounded by the propagation of errors when synchronization is regained.

One way of introducing resilience to this type of errors is by attempting to localize the error. This is usually done by including unique synchronization code words such as in MPEG 2 [10] and H.263 [12]. These synchronization codewords are usually quite long and therefore must be used relatively infrequently. For coding schemes with memory, including synchronization codes may not be sufficient and the coding schemes may have to be periodically restarted. Periodically coding the frames in intra mode in MPEG 2 is an example of this.

The use of synchronization codes and periodic restarting is sufficient to prevent catastrophic losses for all types of channel errors described above. Certain measures have also been introduced to target errors caused by using variable length codes. Variable length codes allow the propagation of errors because errors in a code word can cause loss of synchronization. A simple way to prevent this, possibly at a cost of some coding efficiency is to use fixed length code words. This is done for vector quantization in [5] and for coding sparse data in [3]. It is also possible to rearrange the variable length codes so that the code words start at known positions in the bitstream as done in [16].

Unequal Error Protection

In video coding, some information is more important than others. For example, loss of information about quantization levels and motion vectors is more damaging than loss of information about DCT coefficients. Even for DCT coefficients, the DC coefficient and lower frequency coefficients contribute more to the subjective and objective quality of the video than the higher frequency coefficients. Hence, when the capacity of the channel is limited, it makes sense to give different levels of protection to the data according to their importance.

In MPEG 2, the mechanism for unequal error protection is provided by scalable coding where the bitstream is divided into several layers. The quality of received video increases as information from successively higher layers are received. The information from the lower layers is usually more important and can be given higher protection.

This approach works well when different priority levels are available such as in priority-controlled cell-based transmission on the B-ISDN. Rate-compatible punctured convolutional (RCPC) codes have also been proposed for providing unequal error protection [7].

Concealment

When information is lost due to transmission errors, entire 8×8 pixel blocks may be missing from the decoded image. A good approximation of these lost blocks can often be obtained by using information available at the receiver. The simplest method would be to replace the block with the block from the same position in the previous frame. An estimate of the motion vector of the lost block can also be made using the motion vectors of neighbouring blocks.

For very low bit-rate video coding it is desirable to implement resynchronization and localization, unequal error protection and concealment as efficiently as possible. Coding gains can be obtained by coding for more than one of these categories simultaneously. This can often be done without much additional complexity.

We implement simultaneous coding for resynchronization and unequal error protection in the Multiplexing Layer (referred to as the Systems Layer in MPEG [9]). This layer implements time division multiplexing of audio and video data for efficient throughput. The content of each packet of the multiplexing layer is described in the header of the packet. Loss of this information results in the loss of all the information in the packet. Another source of loss in the multiplexing layer comes from loss of synchronization due to insertion and deletion errors and buffer overflows. Losses in the multiplexing layer often have a significant effect on the quality of the video [17]. To protect against these losses, each packet usually contains a synchronization flag and a header which is protected by forward error correcting codes. We show that if forward error correcting codes are used to protect the header then the synchronization flag may be unnecessary because the redundancy present in the error protected header provides enough information for synchronization purposes.

Section 2 describes the functions of the Multiplexing Layer. This section examines several possible techniques to improve the error resilience of this layer over ITU-T Recommendation H.223 [13], and concludes that the use of fixed length packets, in combination with error correcting codes on packet headers and multiple transmission of packet headers gives the best performance for both random errors and the bursty errors typical of digital mobile channels. It is also found that the error-protected packet headers can be used for resynchronization, and that it is unnecessary to use special synchronization flags.

Section 3 describes techniques for improving the performance of the video layer in the presence of errors. This work is restricted to techniques that can be implemented with a simple transcoder, and the results show that significant improvements in quality can be obtained by replacing the video start codes with parity information for the video data following. This can be achieved without significant loss of resynchronization accuracy.

II. Synchronization and Unequal Error Protection in the Multiplexer Layer

Most multiplexing protocols currently used for audio-visual services (such as H.223 [13] and MPEG 2 Systems [9]) take the form of a packet-based multiplex. Data from each of the elementary streams (e. g. a single channel of video or audio) is packetized, and the packets from all streams associated with a call are time-division multiplexed onto a single channel.

In general, a packet consists of four parts:

1. a synchronization flag, to enable the receiver to unambiguously identify the location of the start of a packet,
2. a header, usually containing address and other information,
3. a body, containing the information to be sent, and
4. a trailer.

The error robustness is sometimes improved by the inclusion of error control codes (such as a CRC or BCH codes [15]) in one or more fields. In some cases, one or more of these fields may be specified to be empty.

A. Packet Loss

There are two ways in which all the data in a multiplexer packet may be lost to the receiver. First, incorrect operation of the synchronization process may cause unsynchronized data to be passed on to the receiver or correctly synchronized data to be discarded because the receiver believes incorrectly that it has lost synchronization. Secondly, if the header of the packet is corrupted, the data contained in the body of the packet will be meaningless to the receiver. These two methods of packet loss are explained in more detail in the following sections.

A. Packet Loss Due to Synchronization Errors

The synchronization of incoming multiplexer packets is performed using two basic functions as shown in Figure 1. The first function of maintaining synchronization is performed as shown in Figure 1.

The synchronization flag of a packet is usually a codeword with good autocorrelation properties such as a Barker code or a pseudo noise sequence [8]. The first step in the procedure for maintaining synchronization is to compare

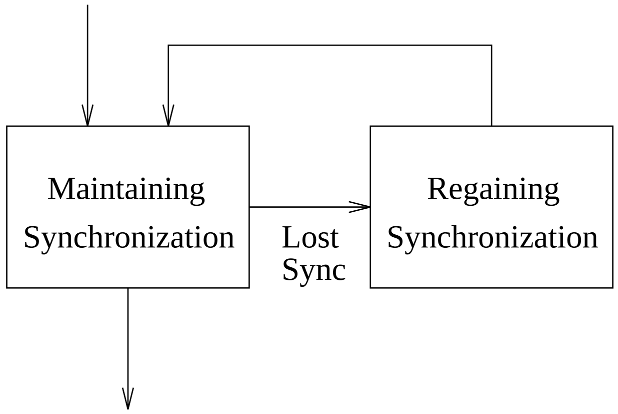

Fig. 1. Block diagram of the synchronization process.

the correct codeword with the received synchronization flags of the next k_1 packets. If the total number of bits differing between the correct codeword and the received flags is greater than some threshold (denoted by T_1), then it is decided that synchronization has been lost and the regain synchronization function is performed. If the total number of errors is less than T_1, then it is decided that the incoming bits are in synchronization and the packets are passed on to the receiver.

The second function of regaining synchronization is performed as follows:

Step through the incoming bitstream one bit at a time looking for a set of flags in the next k_2 packets for which the total number of bit errors when compared to the correct codeword is less than some threshold (denoted by T_2). When a set of k_2 flags have been found with total errors less than T_2 it is decided that synchronization has been regained and the maintaining synchronization function is continued.

Packet loss can occur in this process in the following three ways:

1. In the maintaining synchronization function, if T_1 is too small channel errors in the synchronization flags may cause the function to falsely decide that synchronization has been lost. As a result one or more packets of correctly synchronized data will be discarded while synchronization is regained unnecessarily. The probability of this occuring is P(deciding not in sync|in sync) and is denoted by PDNS_S.
2. In the maintaining synchronization function, if T_1 is too large the algorithm may fail to detect that synchronization has been lost because of a bit insertion or deletion. This failure will cause a packet of unsynchronized data to be passed on to the receiver. The probability of this occuring is P(deciding in sync|not in sync) and is denoted by PDS_NS.
3. In the regaining synchronization function, if T_2 is too large, a set of k_2 groups of bits may be similar enough to the correct codeword for the function to falsely decide that synchronization has been regained. Alternatively, if T_2 is too small and the correct set of k_2 flags is compared with the codeword, the function may falsely decide that synchronization has not been regained. The regaining synchronization function is used only when synchronization has been lost. It is expected that synchronization will be lost only rarely, hence for simplicity, we combine measurements of the two situation. The probability that the regaining synchronization function fails to correctly regain synchronization at the next available set of k flags is P(not regaining sync) and is denoted by PNS. This probability is an upper bound for the probabilities of the two types of failures described above. At least one packet of correctly synchronized data will be lost each time the function fails to correctly regain synchronization.

B. Packet Loss Due to Header Errors

The header of a multiplexer packet usually contains bits describing the type of data contained in the packet. If this header is corrupted, the data in the body of the packet becomes meaningless to any higher level decoding processes. For this reason, it is necessary to protect the data in the packet header using FEC codes.

In the following section, a new method of configuring the header and synchronization information of a packet is described and compared with other conventional packet configurations.

B. Packet Configuration

In this section, we describe the configuration of a multiplexer packet which sends the FEC protected header information twice and does not use a synchronization flag. For comparison, we also describe two other more conventional configurations.

In order to increase the robustness of the multiplexing protocols for transmission over wireless networks, we will consider configurations with fixed packet lengths.

The two conventional configurations we use for comparison are the following:

BLOCK The packet starts with a synchronization flag and a header protected by error correcting codes. This is followed by the information bits.

INTERLEAVE The packet starts with a synchronization flag. The FEC protected header is interleaved throughout the information bits so that they are evenly spread throughout the packet.

The BLOCK configuration is a simple protocol similar to H.223 (except for the fixed packet length). The INTERLEAVE configuration is used in the H.22P/M [11] proposal for low bitrate multimedia mobile communication.

The new packet configuration, which we call DOUBLE, has no synchronization flag and an FEC protected header that is sent twice, once at the beginning of the packet and once at the end of the packet. For this configuration the synchronization functions use the following simple strategy: Match the first and second code words in the packet and compare the number of mismatched bits (summed over k_1 or k_2 packets) to the threshold values T_1 and T_2 to decide whether the packets are properly synchronized. In addition, we use an unequally spaced interleaving of the header to improve the synchronization property of the strategy. We now describe unequal spaced interleaving and other implementation issues for the DOUBLE configuration.

A. Unequal Interleaving of the Codeword

One potential disadvantage of using the DOUBLE structure, as opposed to using a synchronization word with good autocorrelation properties such as a Barker code, is that the number of mismatched bits will be small if the sequence has only shifted a small amount (such as from an insertion or deletion error). This means that P(deciding in sync|not in sync) will be high for bit insertions or deletions and the probability of acquiring synchronization incorrectly is also high. We mitigate this effect by using unequally spaced interleaving for the code words. Note that equally spaced interleaving will still suffer badly from increased probability of acquiring synchronization incorrectly. Since we would like the two code words to be as far away temporally as possible, it is important that the interleaved code words are interleaved over as small a region as possible. We use a simple scheme where the number of bits between the nth bit and the $n+1$st bit is n for the first code word. To concentrate most of the bits of the second code word near the end of the packet we use an interleaving scheme which is the mirror image of the first scheme, i.e. the nth bit and the $n+1$st bit are separated by $code\ size - n$ bits. (For the (31,6) BCH code words, a slightly different scheme, where the difference between the nth and $n+1$st bit is approximately $n/4$ is used so that the code words are concentrated near the ends of the packets.)

Note that it is possible to use the code words for synchronization purposes even if only one code word is sent, such as in the BLOCK and INTERLEAVE structures. In this case, we can use the number of errors detected as information for deciding whether we are in synchronization. If cyclic code words such as the BCH code are used, it is also advisable to use some unequal spaced interleaving for the same reason that it is used in the DOUBLE structure (in cyclic codes, a cyclic shift of a code word is another code word).

B. Decoding the Header

For the DOUBLE configuration, decoding of the header information is done by first detecting the errors in the two code words. The code word with the smaller number of detected errors is assumed to give the correct header.

If the two copies of the headers are separated far enough temporally, this has a similar effect to sending the headers on two independent channels. In such a case, the probability of both copies being corrupted by errors is the squared of the probability of a copy being corrupted by errors in a single channel. As we show in Section II.D, this can greatly improve the performance of the system.

C. Decoding Delay

One of the disadvantages of using the DOUBLE and INTERLEAVE configurations is the delay caused by having to receive the whole packet before decoding can begin. By using the DOUBLE configuration, delay can be avoided when the channel is in a good state, by accepting a slight reduction in performance. This is achieved by assuming that the first word is correct if no error is detected. If errors are detected in the first word, then the code word with the smaller number of detected errors is used. In this way, delay happens only when an error is detected. This decoding strategy can be extended so that the second word is used only when more than one (or any number) of errors are detected in the first word.

In comparison to the DOUBLE configuration, the INTERLEAVE configuration needs to receive almost the whole packet before decoding even when the channel is in a good state. The BLOCK configuration has much lower delay than the DOUBLE configuration when the channel is in a bad state but it suffers from the problem of passing an incorrectly decoded packet to the higher layer instead of a delayed but correctly decoded one.

We conducted the experiments described in the following section to determine the packet loss performance of the three packet configurations for several different channel error characteristics.

C. Experimental Procedure

The experiments were conducted for six channel models:

1. a simulation of a DECT channel with bit error rate 2×10^{-2} (DECT1) [4],
2. a simulation of a DECT channel with bit error rate 2×10^{-3} (DECT2) [4],
3. a Gilbert channel with bit error rate 2×10^{-2}, an average burst length of 240 bits and a bit error probability of 0.2 in the bad state (GILB1) [6],
4. a Gilbert channel with bit error rate 2×10^{-3}, an average burst length of 240 bits and a bit error probability of 0.2 in the bad state (GILB2),
5. a binary symmetric channel with bit error rate 2×10^{-2} (BSC1), and
6. a binary symmetric channel with bit error rate 2×10^{-3} (BSC2).

The errors in DECT1 and DECT2 are extremely bursty with the bursts in DECT1 much longer and more frequent than the bursts in DECT2. The Gilbert channel is a simple model of a bursty channel (see e.g. [14] for a description).

For the BLOCK and INTERLEAVE structures we used the 31 bit pseudo-noise sequence "1111100011011101010000100101100" as the synchronization flag and a (31,6) BCH code to protect the header. The same pseudo-noise sequence is also used in the H.22P/M proposal. For the DOUBLE structure we tested two different configurations. For the first configuration we used two copies of a (31,6) BCH protected header at both ends of the packet and in the second configuration we used a (15,7) BCH code.

A. Determining values for T and k

The values of T_1, T_2, k_1 and k_2 were determined for each strategy by first determining the values of PDNS_S, PDS_NS and PNS versus threshold for several different values of k. Here, k is the number of packets used for maintaining or regaining synchronization and is used for choosing k_1 and k_2. Figure 2 shows a plot of these values versus threshold when $k = 1$ and $k = 4$ for the DECT1 channel using the DOUBLE configuration with a packet length of 1024 and a (15,7) BCH code.

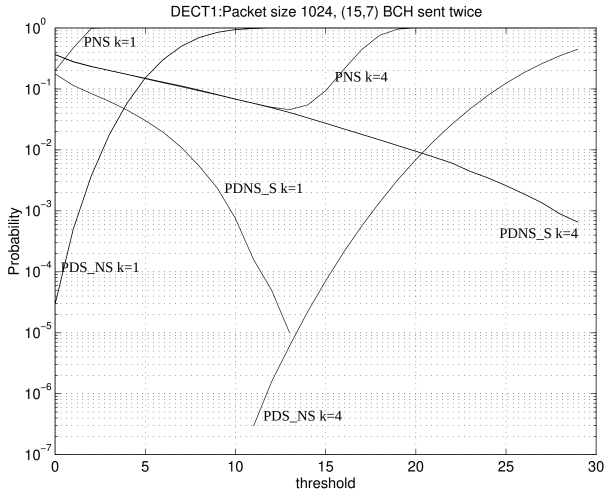

Fig. 2. Plots for synchronization experiments using DOUBLE structure. DECT1, packet size 1024.

The values for PDS_NS were calculated by taking the average number of times that the maintaining synchronization falsely assumed synchronization for every positions in a sample corrupted bitstream except the correct position. We calculated PNS by running the regaining synchronization function from the first position in the packet after the correct synchronization position up to the next correct synchronization position and checking if we have acquired synchronization correctly.

Without statistics on the insertion and deletion errors, it is not possible to choose optimal values of T_1. For the experiments we conducted, we assumed that the probability of losing synchronization because of a bit insertion or deletion is negligible. However, in order to choose a realistic value for T_1 we have restricted the value of PDS_NS to be less than 0.2. With this restriction, we pick values of T_1 and k_1 such that the synchronization errors would be much smaller than header errors. T_2 and k_2 were similarly chosen as the smallest values which gave a satisfactory value for PNS. (We pick values of T_2 and k_2 such that PNS is less than 0.2.)

The performance of the overall system is not very sensitive to the values of T_1, k_1, T_2 and k_2 as long as they are chosen such that packet losses from synchronization error is much smaller than packet loss from header error. This is quite easily achieved in our experiments in Section II.D. If the error characteristics of the channel is not known exactly, the values of T_1, k_1, T_2 and k_2 can be chosen for a worst case channel.

For the case shown in Figure 2, k_1 was chosen as 4 and T_1 was chosen as 26. It can be seen from Figure 2 that these values guarantee PDS_NS is less than 0.2 and PDNS_S is suitably small with a value of approximately 2.5 x 10^{-3}. Then k_2 was chosen as 1 and T_2 was chosen as 0 since these values give a PNS value of approximately 0.2. This value of PNS is satisfactory since the probability of the regaining synchronization function falsely being called (PDNS_S) is 2.5 x 10^{-3}. Figures 3, 4 show the same results plotted for the GILB1 and BSC1 channels. The values for T and k were chosen in a similar manner for these channels. For most of the experiments in Section II.D, the value of k_1 is 4 and the values of the T_1's are approximately 25 for (15,7) BCH protected headers (approximately 55 for all other cases such as the (31,6) BCH DOUBLE configurations, the BLOCK configurations and the INTERLEAVE configurations). The values of k_2 is between 1 and 2 and the values of T_2 between 0 to 5. The resulting total packet loss probabilities are not sensitive to these values since the probability of losing the header is usually at least an order of magnitude larger than the probability of packet loss through synchronization errors.

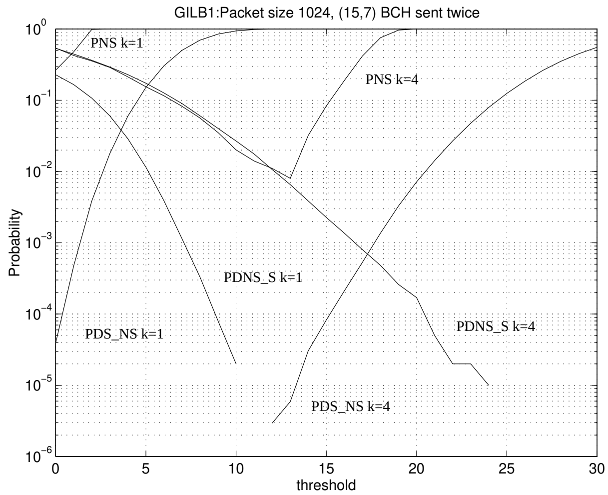

Fig. 3. Plots for synchronization experiments using DOUBLE structure. GILB1, packet size 1024.

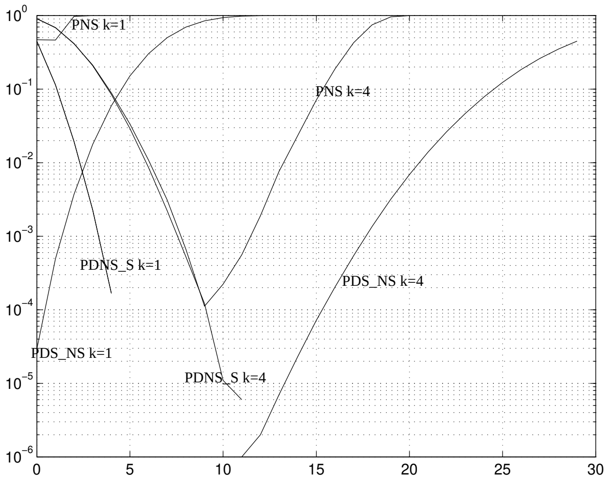

Fig. 4. Plots for synchronization experiments using DOUBLE structure with BSC1.

B. Determining Packet Loss

The following simple strategy for determining packet loss due to synchronization is used. When a synchronization loss is detected (falsely in our experiments since we do not consider actual synchronization loss), the receiver will consider the next packet as a lost packet and call the function to regain synchronization. For our measurements, all the packets until synchronization is correctly regained are considered lost. (This includes the packets lost because the receiver acquired synchronization at the incorrect place. In such a case, the packets are lost until the error is detected again and synchronization is acquired at the correct place.)

D. Results

The results of the packet loss experiments are shown in Table 1. These results show that, when using the same number of bits for synchronization and header protection, the DOUBLE structure performs better for bursty channels such as DECT1, DECT2, GILB1 and GILB2. The DOUBLE structure performs much better in combating header loss when the channels are bursty even though its synchronization performance is slightly poorer. Since the errors are dominated by the header loss, it gives a much better overall performance. Another effect of the errors being dominated by header loss is that the performance of the overall system is not sensitive to the choice of parameters in the synchronization component.

The use of the headers for synchronization purposes means that the probability of header loss is related to the probability of synchronization loss. However, since a window of several packets (k_1 and k_2) is used for synchronization purposes, the probability of synchronization loss is much smaller than the probability of header loss.

To check the performance of the DOUBLE structure over a variety of bit error rates, we also performed the experiment over a variety of bit error rates for the Gilbert channel. The resulting plot of probability of packet loss (from both synchronization errors and header errors) vs the bit error rates is shown in Figure 5.

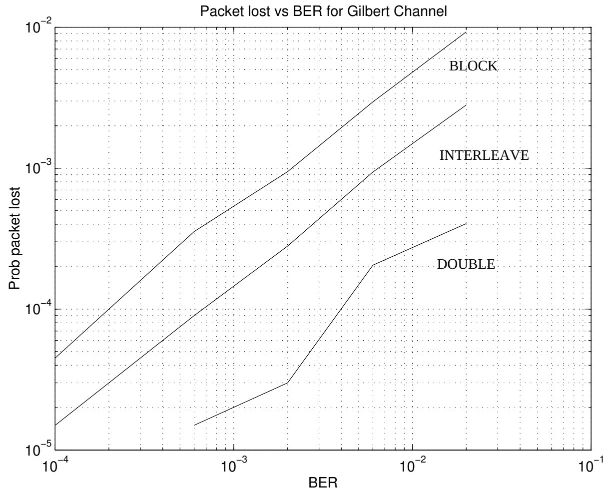

Fig. 5. Probability of multiplexer packet loss vs bit error rate for Gilbert channel.

Table 2 shows the results when the DOUBLE structure is used with two (15,7) BCH codes instead of two (31,6) BCH codes. The performance of this DOUBLE structure is comparable to that of the INTERLEAVE structure even though it uses only half as many bits.

It is important to compare the results of sending a single (31,6) BCH protected header interleaved throughout the packet with sending two (15,7) BCH protected headers (since it is also possible to use the single (31,6) BCH protected header for synchronization purposes although we have not done so). The results show that for the binary symmetric channel, sending one longer code word is better than sending two smaller ones as we would expect. However, for the DECT channels, the results show that, for a long enough packet, sending two shorter code words at the beginning and end of the packet gives a better result than sending one longer code word. This is because, when most of the bursts are shorter than the packet length , the two code words at opposite ends of the packets are rarely affected by error bursts at the same time. In such a situation, the decoding error from taking the code word with the smaller detected error of the two code words will be small. In contrast, each error burst will still corrupt a significant portion of at least one interleaved code word if the interleaving (which is restricted by the packet size) is not deep enough.

III. Synchronization and Error Detection in the Video Layer

Of the currently available video coding standards, H.263 is the most suitable for very low bitrate channels, particularly those with rates below 40 kbit/s. The syntax of H.263 breaks each video frame into one or more Groups of Blocks (GOB), each of which consists of one or more

Packet Size	*Channel*	*Strategy*	*PH*	*PS*	*PT*
1024	DECT1	BLOCK	2.1×10^{-2}	$< 1 \times 10^{-5}$	2.1×10^{-2}
		INTERLEAVE	1.6×10^{-2}	$< 1 \times 10^{-5}$	1.6×10^{-2}
		DOUBLE	7.9×10^{-3}	9.9×10^{-4}	8.6×10^{-3}
	DECT2	BLOCK	1.9×10^{-3}	$< 1 \times 10^{-5}$	1.9×10^{-3}
		INTERLEAVE	5.4×10^{-4}	$< 1 \times 10^{-5}$	5.4×10^{-4}
		DOUBLE	2×10^{-5}	$< 1 \times 10^{-5}$	2×10^{-5}
	GILB1	BLOCK	9.0×10^{-3}	$< 1 \times 10^{-5}$	9.0×10^{-3}
		INTERLEAVE	6.9×10^{-4}	$< 1 \times 10^{-5}$	6.9×10^{-4}
		DOUBLE	5×10^{-5}	$< 1 \times 10^{-5}$	5×10^{-5}
	GILB2	BLOCK	9.2×10^{-4}	$< 1 \times 10^{-5}$	9.2×10^{-4}
		INTERLEAVE	3×10^{-5}	$< 1 \times 10^{-5}$	3×10^{-5}
		DOUBLE	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	BSC1&BSC2	ALL	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
512	DECT1	BLOCK	2.1×10^{-2}	$< 1 \times 10^{-5}$	2.1×10^{-2}
		INTERLEAVE	2.0×10^{-2}	$< 1 \times 10^{-5}$	2.0×10^{-2}
		DOUBLE	1.5×10^{-2}	5.3×10^{-3}	1.9×10^{-2}
	DECT2	BLOCK	2.0×10^{-3}	$< 1 \times 10^{-5}$	2.0×10^{-3}
		INTERLEAVE	1.2×10^{-3}	$< 1 \times 10^{-5}$	1.2×10^{-3}
		DOUBLE	4.8×10^{-4}	$< 1 \times 10^{-5}$	4.8×10^{-4}
	GILB1	BLOCK	9.3×10^{-3}	$< 1 \times 10^{-5}$	9.3×10^{-3}
		INTERLEAVE	2.8×10^{-4}	$< 1 \times 10^{-5}$	2.8×10^{-3}
		DOUBLE	4.1×10^{-4}	$< 1 \times 10^{-5}$	4.1×10^{-4}
	GILB2	BLOCK	9.5×10^{-4}	$< 1 \times 10^{-5}$	9.5×10^{-4}
		INTERLEAVE	2.8×10^{-4}	$< 1 \times 10^{-5}$	2.8×10^{-4}
		DOUBLE	3×10^{-5}	$< 1 \times 10^{-5}$	3×10^{-5}

Table 1. Performance of the multiplexer packet configurations. PH is the probability of packet loss from corrupted headers. PS is the probability of packet loss from synchronization errors. PT is the probability of packet loss from both corrupted headers and synchronization errors (this may be less than the sum of the two losses because some packets lost by synchronization errors are also included in the measurement of header errors). Total header + synch bits = 64.

Packet Size	*Channel*	*PH*	*PS*	*PT*
1024	DECT1	1.8×10^{-2}	3.1×10^{-3}	2.0×10^{-2}
	DECT2	1.9×10^{-4}	$< 1 \times 10^{-5}$	1.9×10^{-4}
	GILB1	3.7×10^{-3}	$< 1 \times 10^{-5}$	3.7×10^{-3}
	GILB2	6×10^{-5}	$< 1 \times 10^{-5}$	6×10^{-5}
	BSC1	4.4×10^{-5}	$< 1 \times 10^{-5}$	4.4×10^{-5}
	BSC2	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
512	DECT1	2.9×10^{-2}	1.0×10^{-2}	3.6×10^{-2}
	DECT2	1.2×10^{-3}	$< 1.5 \times 10^{-5}$	1.2×10^{-3}
	GILB1	8.3×10^{-3}	$< 1 \times 10^{-5}$	8.3×10^{-3}
	GILB2	7.2×10^{-4}	$< 1 \times 10^{-5}$	7.2×10^{-4}

Table 2. Performance of Double structure with two (15,7) BCH code words.

rows of macroblocks. Each macroblock is made up of a 16×16 square block of pixels.

In this section, methods for modifying this standard to improve its error robustness are examined. We restrict ourselves to schemes that do not alter in any way the total number of bits required for the video bitstream, and which could be implemented with simple transcoders. This means that all improvements in performance are gained with no cost in the number of bits that must be transmitted. Such an approach has great advantages for implementation at interfaces between different telecommunications networks, but rules out many existing schemes for reordering of data, such as EREC [16].

The restriction of not changing the number of bits transmitted led us to examine techniques in which the 17 bits used in H.263 start codes are replaced with bits that can be used for both the synchronization function of the start code and for providing error protection for the bitstream. Two basic approaches were tried:

1. using the 17 bits to provide error correction for the information in frame and GOB headers,

2. using the 17 bits to provide error detection for the bits used to code the video information in the GOB.

The first approach was found to be almost completely useless in the presence of burst errors. The most likely reason is that where a burst of errors occurs in a frame or GOB header, it is most likely that some bits after the header are also corrupted. In this case, even if the header is correctly decoded, very little if any video data is correctly decoded. Also, because these schemes did not detect errors in the remainder of the bitstream, they did not reduce the occurrence of obnoxious artifacts. The second approach was found to produce a useful improvement in decoded video quality. The remainder of this section describes the method for implementing error detection and the performance obtained.

A. General Description

For video sequences where concealment is effective, it is useful to be able to detect the presence of errors as soon as possible so that concealment can be carried out. Errors are usually detected when illegal variable length codes are found in the video bit-stream. While providing good compression, variable length coded data suffers from the problem that, when transmission errors occur, the incoming bits can still form valid entries in the VLC table and the decoded image will contain incorrect information. The decoder can continue decoding incorrect codewords for some time before an error situation is detected and the decoder must take some action to resynchronize the incoming VLC data. The effect of incorrect VLC codewords is often an unusually large DCT coefficient. When this block is transformed to the spatial domain, the effect is manifested as a block which is similar in appearance to a DCT basis function. This type of error, which causes one DCT basis function to dominate the appearance of the block, is highly perceivable as distortion in the image. Another very noticeable type of distortion is caused by incorrectly decoding the dc component of a chrominance block. This type of error tends to cause the colour of a 16×16 block to be predominantly pink, green or cyan. Eliminating both these types of distortion will significantly improve the perceived quality of the image.

It is possible to code the start code in such a way that it can be used to detect some errors in the video bit-stream (instead of relying on illegal variable length codes) and yet maintain good synchronization properties. Let the start code be of length l. Fix a word length w for the video bitstream. For the ith word after the start code, calculate the parity of the word and replace the ith bit of the start code with the parity bit. A maximum of $l \times w$ bits of GOB data can be protected in this way.

A complication arises in using this scheme when two start codes appear close together in the bit stream. In such a case, the first start code cannot be replaced until the second code word has been replaced by parity bits (so that the parity bits can be correctly calculated for replacing the first start code). This means that the scheme faces potentially long delays. To overcome this problem we simply stop the process of replacing the bits in the start code with parity bits once a new start code has been encountered. Call the start code which has been partially (or fully) replaced by parity bits the *parity flag*. The process of replacing the start codes when two start codes appear close together is shown in Figure 6.

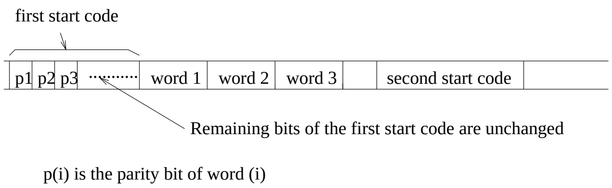

Fig. 6. Transcoding scheme when two start codes are close together.

At the receiver, when searching for synchronization, the parity bits for consecutive words are calculated and compared to the bits of the parity flag. If an error is found when calculating the parity of the ith bit, the ith parity bit together with the remaining $l-i$ bits of the parity flag are compared to the last $l-i+1$ bits in the start code (which is also known to the receiver, e.g. 00000000000000001 for H.263). If the protected words and the parity flag have not been corrupted by errors, then the $l-i+1$ bits should match exactly. So if the number of mismatched bits is below a certain threshold (we allow no mismatched bits in our implementation) or if there is no parity error, we assume that synchronization has been found, otherwise we proceed to the next possible synchronization position. Since, the video bit-stream is generally random, this scheme should have good synchronization properties.

When the receiver is in synchronization, the parity flag is used to detect the presence of errors. As in the case of searching for synchronization, if an error is found when calculating the parity of the ith bit, the ith parity bit together with the remaining $l-i$ bits of the parity flag are compared to the last $l-i+1$ bits of the start code. If no mismatched bits are found, we assume that no errors have occured and proceed with the decoding. If the bits do not match exactly, we decide that an error has been found. Once an error has been found, the macroblock containing the error is discarded and the decoder starts looking for the next resynchronization point.

Implementation of this method involves a small additional delay in encoding and decoding. In the encoder, this delay occurs because all the parity bits are transmitted at the beginning of the data they protect. In the decoder, the delay results from the use of the parity bits with the data they protect for synchronization. In all cases examined here, this delay is approximately 200 bits, corresponding to 10 ms for a video rate of 20 kbit/s. Clearly, performance depends on choosing a suitable value for w. If w is too small, a large amount of the video data will not be protected. If w is too large, the effectiveness of the error detection will be reduced.

B. Implementation

The modified scheme as described in Section III can be implemented by transcoding a standard video bit-stream with only modest resources. We implemented the scheme by transcoding a standard H.263 bit-stream and implementing some mild modification to a H.263 decoder. A word length of $w = 20$ is used. The standard H.263 synchronization flag has length $l = 17$.

The synchronization flag is used in two modes. If the previous GOB has been decoded correctly, it is used to detect the presence of the first error in the GOB. The first macroblock in the GOB with detected error is discarded with the rest of the GOB and concealment is done (Here we assume that the error occurs in the protected words even though the error may have occured in the parity flag). If the previous GOB has has not been correctly decoded (synchronization lost), then the synchronization flag is used for synchronization by checking every byte for a possible synchronization flag (since the H.263 standard requires the synchronization flags to be byte-aligned). To cope with missed synchronization flags (due to errors in the bit-stream), the transmitted GOB number is used if it is no more than two larger than an internal counter. Otherwise, the GOBs are considered lost until synchronization is regained when two consecutive GOB numbers are received.

The H.263 decoder we implemented uses a very simple concealment strategy. When an error is detected, the macroblock and the rest of the macroblocks in the GOB are not updated and the macroblocks from the previous frame are used in their place. If a frame is missed, the current frame is used in its place until the correct number of frames has been received.

C. Simulation Results

Simulations were performed on the scheme using the test sequences "mother and daughter" and "container" at QCIF resolution. We coded 100 frames of each sequence at a bit-rate of 24-30 kbits/s and a frame rate of 10 frames/s. The synchronization flags make up approximately 5% of the data in these sequences. Concealment is effective for both these sequences because there is little motion in the sequences. We used three error sequences: a realistic simulation of a DECT channel with bit error rate of 2×10^{-3} [4], a Gilbert channel with bit error rate 2×10^{-3}, an average burst length of 240 bits and a bit error rate of 0.2 in the bad state (GILB) and a binary symmetric channel with bit error rate of 2×10^{-4}. The errors in the DECT and Gilbert channels are very bursty and have a different effect on the video compared to the binary symmetric channel.

The results averaged over 20 simulation runs are shown in Table 3. For comparison, simulations were also performed for a H.263 bit-stream with the standard synchronization flags. The same block replacement concealment strategy is used in both cases. In the simulations, no errors are included in the first frame to allow good concealment.

It can be seen from Table 3 that the parity strategy provides significant improvements for the "mother and daughter" and "container" sequences. We also did simulations for the Gilbert channel over a range of bit error rates. The resulting plots of PSNR vs BER for the luminance components of "mother and daughter" and "container" can be seen in Figures 7 and 8.

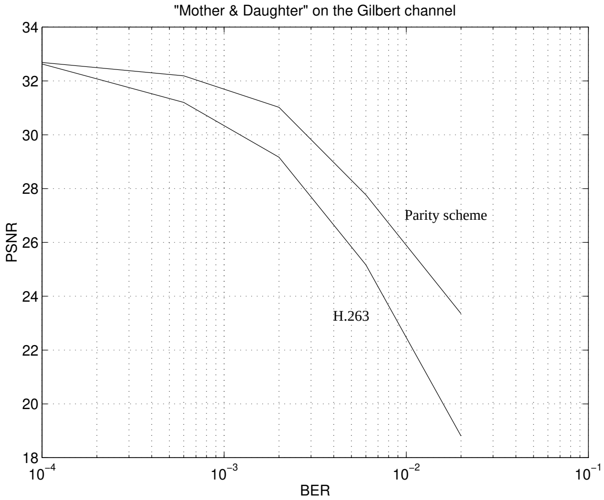

Fig. 7. Luminance component of "mother and daughter".

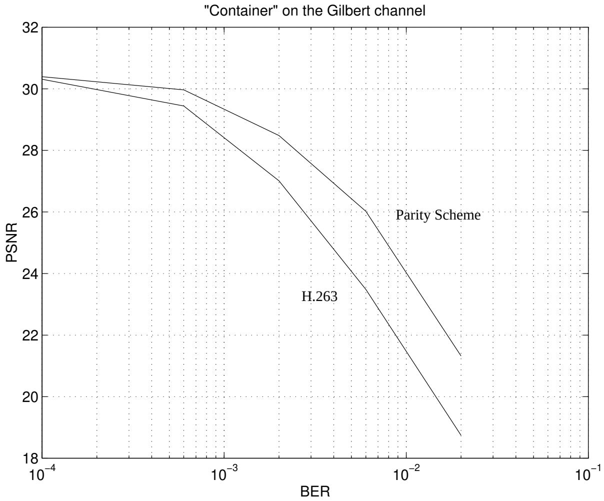

Fig. 8. Luminance component of "container".

Since the parity checking technique allows the detection of errors before an incorrect VLC codeword has been decoded and transformed to the spatial domain, the perceived quality of the images decoded using this technique is also improved by eliminating the highly visible, DCT basis function type errors from the decoded image. Figure 9 shows badly corrupted decoded luminance frames of the parity scheme and H.263. An even greater improvement in subjective quality occured due to a significant

Sequence	*Scheme*	*Channel*	*Y*	*U*	*V*
mother & daughter	Both	no errors	32.69	39.22	39.95
	H263	DECT	29.54	34.56	34.60
		GILB	29.17	34.26	33.45
		random	27.12	31.28	31.43
	Parity	DECT	30.33	37.51	38.13
		GILB	31.02	37.80	37.09
		random	28.95	34.93	35.88
container	Both	no errors	30.33	37.46	36.74
	H263	DECT	27.32	33.35	32.65
		GILB	27.01	33.15	32.38
		random	25.68	30.66	29.71
	Parity	DECT	28.57	35.26	34.43
		GILB	28.48	34.92	33.45
		random	26.63	31.00	31.35

Table 3. PSNR averaged over 20 error sequences.

reduction of the chrominance errors. This can be seen from the improvements in PSNR for chrominance, which can be as high as 3.5 dB.

We also performed simulations on the sequence "foreman" which contains a high degree of motion. The effect of the parity checking technique was unclear for the "foreman" sequence because of the simple concealment strategy used. The PSNR of the images decoded using both techniques was reduced to approximately 20 dB and there was no significant difference between the average PSNR values obtained for the two techniques. The improvement to be gained by using the parity checking technique should be more easily determined if a motion compensated concealment method such as that described in [17] were used. The incorporation of such an improved concealment technique into the decoding process is currently being investigated.

IV. Conclusion

In this paper, we have examined techniques for improving the quality of the video component of audio-visual services in the presence of transmission errors. The techniques presented have emphasised coding efficiency and compatibility with existing and upcoming standards, attempting to redefine those bits used for synchronization such that they can be used simultaneously for error detection. The key advantage of this approach is that it greatly aids the implementation of these techniques in a network interface, without requiring encoders to be able to operate in a special error-robust mode. It also means that all of the improvements gained are essentially "free", at least in terms of the number of bits that must be transmitted.

Most existing systems for the transmission of audio-visual data use a packet-based multiplex to combine the audio and video onto a single channel. The performance of this multiplexer in the presence of transmission errors is very important to the overall video quality achieved. As shown above, the combination of fixed length packets with repeated transmission of packet headers, combined with the use of BCH codes to estimate the number of errors contained in received headers, can reduce the packet loss probability by approximately an order of magnitude compared to a scheme that does not use repeated transmission of data. This means that more received bits can be made available to the video decoder, and hence that the video quality is improved. The use of the BCH codewords to provide both synchronization information and error detection means that this scheme is particularly efficient. Other techniques that attempted to use error correction were not found to be effective, even when used in conjunction with interleaving.

In the video layer, our approach has been to detect bit errors as soon as possible, and to discard all bits between the detection of an error and the next resynchronization point. By replacing the 17 bits used in the start code in H.263 with parity bits, the implementation of this error detection scheme requires no more bits than the original video bitstream. The synchronization properties of the bitstream are not significantly degraded, but the ability to detect errors in the bitstream directly resulted in a modest but noticeable improvement in quality, as measured by PSNR. What is probably more important is that the new technique effectively eliminates the most obnoxious artifacts usually associated with bit errors in video, i. e. the bright pink, green and cyan blocks and blocks that look like a single DCT basis function. We note that this was achieved using identical concealment strategies in the H.263 decoder as in our modified decoder.

In conclusion, we have shown in this paper that the impact of transmission errors on the quality of decoded video can be reduced by the use of a combined error detection / synchronization strategy, in which the same bits are used for both purposes. This facilitates the implementation of simple transcoders to support telecommunications networks that require error robustness, without compromising performance on other systems.

(a) (b)

Fig. 9. Frame 40 (of 100) of the "mother and daughter" sequence corrupted with errors from DECT channel for the (a) parity scheme and (b) H.263.

References

[1] Physical Layer on the Radio-path. Vol G, Group Special Mobile (GSM-Europe). Recommendation 05.02. G.S.M. Standard Committee, 1990.

[2] Digital Enhanced Cordless Telecommunications (DECT) Common Interface, Part 2: Physical Layer (ETS 300 175-2). European Telecommunications Standards Institute (ETSI), 1992.

[3] N. T. Cheng and N. G. Kingsbury. The ERPC: An efficient error-resilient technique for encoding positional information or sparse data. *IEEE Trans. Commun.*, 40(1):140–148, 1992.

[4] P. Crespo, R. Mann Pelz, and J. Cosmas. Channel error profiles for DECT. *IEE Proceedings-Communications*, 141(6):413–420, 1994.

[5] N. Farvardin. A study of vector quantization for noisy channels. *IEEE Trans. Inform. Theory*, 36:799–809, 1990.

[6] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Syst. Tech. Journal*, 39:1253–1266, 1960.

[7] J. Hagenauer. Rate-compatible punctured convolutional codes (RCPC codes) and their applications. *IEEE Trans. Commun.*, 36(4):389–400, 1988.

[8] J. E. Hershey and R. K. R. Yarlagadda. *Data transportation and protection.* Plenum Press, New York and London, 1986.

[9] ISO/IEC International Standard 13818-1. *Generic coding of moving pictures and associated audio information: Systems.* 1995.

[10] ISO/IEC International Standard 13818-2. *Generic Coding of moving pictures and associated audio information: Video.* 1995.

[11] ITU-T Draft Recommendation H.22P/M. *Multiplexing Protocol for Low Bitrate Multimedia Mobile Communication.*

[12] ITU-T Draft Recommendation H.263. *Video Coding for Low Bitrate Communication.*

[13] ITU-T Recommendation H.223. *Multiplexing Protocol for Low Bitrate Multimedia Communication.*

[14] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan. *Simulation of Communication Systems.* Plenum Press, New York and London, 1992.

[15] S. Lin and D. J. Costello. *Error control coding.* Prentice-Hall, Englewood, NJ, 1983.

[16] D. W. Redmill and N. G. Kingsbury. The EREC: An error-resilient technique for coding variable-length blocks of data. *IEEE Trans. Image Processing*, 5(4):565–574, 1996.

[17] J. Zhang, M. R. Frater, J. F. Arnold, and T. M. Percival. MPEG 2 video services for wireless ATM networks. *IEEE Journal on Selected Areas in Communications*, 15:119–128, 1997.

List of Figures

List of Tables

Wee Sun Lee (S'91-M'96) received the B.E. degree in Electrical Engineering from the University of Queensland, Australia, in 1992 and the Ph.D. degree in Engineering from the Australian National University in 1996. He is currently a Research Fellow in the Electrical Engineering Department of the Australian Defence Force Academy. His research interests include error resilience for video and image coding, image and video compression and machine learning.

Mark R. Pickering (S'91-M'95) received a B.E. degree in 1988 from Capricornia Institute of Advanced Education, Rockhampton, Australia, and M.E. and Ph.D. degrees in 1992 and 1995 respectively, from The University of New South Wales, Sydney, Australia. He is currently a Lecturer in the School of Electrical Engineering at the Australian Defence Force Academy in Canberra. His research interests include the application of visual perception to image sequence coding and techniques for rate control, error concealment and motion estimation in video coding.

Michael R. Frater (S'89-M'92) was born in Sydney, Australia, in 1965. He received the B.Sc. degree in mathematics and physics and the B.E. (Hons.) degree in electrical engineering from the University of Sydney, Australia, in 1986 and 1988 respectively. He received the Ph.D. degree from the Department of Systems Engineering at the Australian National University, Australia, in 1991.

He is a Senior Lecturer in Electrical Engineering at the University College, the University fo New South Wales, Australian Defence Force Academy, Australia. His research interests include aspects of the coding and transmission of video services, teletraffic and queueing theory, stochastic processes, signal processing, and control.

John F. Arnold (S'77-M'85-SM'96) received the B.E. and M.Eng.Sc. degrees from the University of Melbourne, Australia in 1976 and 1979, respectively, and the Ph.D. degree from the University of New South Wales, Australia, in 1984.

Since 1978, he has been with the Department of Electrical Engineering, University of New South Wales, initially at the Royal Military College, Duntroon and more recently at the Australian Defence Force Academy, Canberra. His research interests lie in the fields of video coding, asychronous transmission mode (ATM) networks, and coding of remotely sensed data.