Structured design of standard-compatible error-resilient video coding, with application to H.263

David W. Lin[a], Yen-Lin Chen[b], and Chin-Tien Lee[c]

[a]Dept. Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan 300, ROC
[b]Avanti Taiwan Corporation, Taipei, Taiwan 110, ROC
[c]Integrated Technology Express, Inc., Hsinchu, Taiwan 300, ROC

ABSTRACT

Many methods for video transmission error control have been proposed recently, especially in relation to transmission over bursty-error channels. However, a thorough, structural taxonomical framework for analysis and design of such methods seems lacking. Such a framework helps clarity in thought and aids in inspiring new error control methods or combinations thereof. We present a framework for classification of the various transmission error control techniques. We then consider error control for H.263. Several techniques are presented from the viewpoint of the proposed analysis framework and they illustrate how different techniques can be integrated coherently to achieve enhanced error resilience in the overall system. In particular, we employ slotted multiplexing at the multiplex level to reduce synchronization errors in variable-length coded data and to randomize the locations of the remaining error-corrupted image areas. At the source level, we introduce two standard-compatible schemes, called length-based intra refresh and motion vector pairing, respectively, which further limit spatial-temporal error propagation.

Keywords: Error resilience, H.263, slotted multiplexing, intra refresh, motion vector pairing

1. INTRODUCTION

International video coding standards instituted in the past few years invariably employ motion-compensated frame prediction, differential coding of motion vectors, DCT (discrete cosine transform), and VLC (variable-length coding) to achieve high compression. As a result, the decoded video quality is quite prone to channel error effects. Consequently, error-resilient video coding and transmission has received much recent attention, including the current work on MPEG4 and revision of H.263.[1–3]

To facilitate a systematic investigation into error-resilient video coding and transmission, it is of interest to construct a suitable model of the coding and transmission process and to conduct a proper classification of the somewhat bewildering array of existing error-resilience methods accordingly. This has been done to a certain degree. For example, one classification emphasizes the procedure of error control in the receiver and divides the methods into the categories (or *stages*) of error detection, error localization, and error concealment, where error detection refers to the determination of whether a bit error has occurred, error localization refers to the narrowing down of the bitstream segment where the error is located or (with some stretching of the meaning) the limiting of the extent of error effects by some means, and error concealment refers to the act of reducing or correcting the effects of the detected errors on the decoded video. A somewhat orthogonal (and hence complementary) classification emphasizes where and how redundancies in information are added and/or exploited and divides the methods into the categories (or *modes*) of forward, postprocessing, and interactive.[3] As the names imply, forward methods may require the action of both the transmitter and the receiver, but the information flow is one-way; postprocessing methods require the action of only the receiver; and the interactive methods require the transmitter to adjust its operation based on information fed back from the receiver. While this classification can be applied rather broadly,[3] in the context of the present work it is interpreted as in reference to a given standard. In this regard, we further divide the forward category into the forward and the preprocessing categories, where the former requires matching modifications to both the transmitter and the receiver, while the latter only the transmitter.

The above classifications may be termed *functional* and *interactivity* classifications, respectively. While instructive and helpful, they appear not thorough and structured enough for a comprehensive methodical error-resilience study. A more thorough and structured model of the error occurrence mechanism and an associated classification of the error-resilience methods should aid in a comparative study of different error-resilience methods and in inspiring possible new methods. We present such a model and classification in Sec. 2. The model and classification is based on the *architectural* representation of the video

Work supported by National Science Council of ROC under grants NSC 86-2221-E-009-050 and NSC 87-2213-E-009-150.
Send correspondence to D. W. Lin: dwlin@cc.nctu.edu.tw.

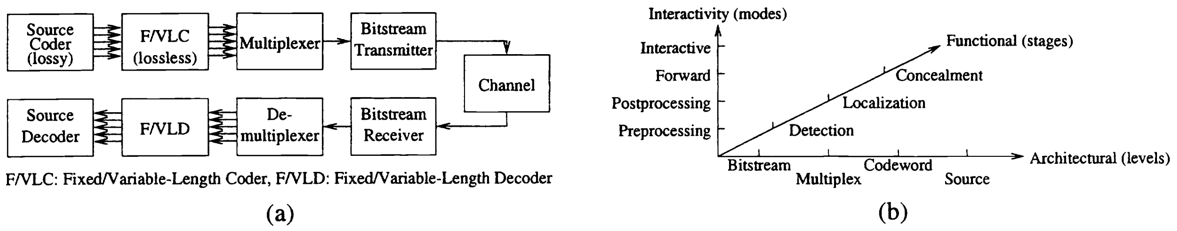


Figure 1. Video transmission system model (a) and a three-dimensional analysis framework for error-resilience studies (b).

coding and transmission system as shown in Fig. 1(a), augmented by the functional and the interactivity sorts of classification noted above.

The functions of each block at the transmitting end of Fig. 1(a) are relatively well-defined. With an eye to error-resilience design, the functions of some receiving-end blocks are as follows. The demultiplexer partitions the received bitstream into a collection of FLC and VLC words. Upon invalid syntax or VLC word patterns due to transmission errors, the demultiplexer may flag the error and advances to the next recognizable bitstream segment. The F/VLD converts each of the FLC and VLC words back into the numerical quantities they represent. The F/VLD may conduct additional syntax and semantics checks such as forbidden FLC words, incorrect number of certain quantities (e.g., DCT coefficients), and range violation of certain quantities. The source decoder generates the reconstructed video from the decoded numerical quantities. It may perform error concealment when there exist transmission errors.

According to this model, and especially according to the receiver model therein, the channel error effects can be categorized into four *levels*, namely, the bitstream level (concerning errors at the bitstream receiver output), the multiplex level (concerning errors at the demultiplexer output), the codeword level (concerning errors at the F/VLD output), and the source level (concerning errors at the source decoder output). While this model correlates well with the standard video codec architecture and is therefore straightforward and natural, it seems to have not been employed much in recent error-resilience studies.

An interesting and practical issue in error control studies is that of standard compatibility. Viewed in light of the above architectural model, different kinds of compatibility can be defined. The fully compatible methods are self-contained procedures operating either at the transmitter end or the receiver end without the other end's coorperation. The incompatible methods may be further categorized according to which and how many inverse function pairs (bitstream transmitter-receiver, multiplexer-demultiplexer, F/VLC-F/VLD, and source coder-decoder) they alter. The methods altering only a subset of the inverse function pairs may be considered partially standard-compatible.

This paper is organized as follows. Sec. 2 describes the proposed analysis framework for error-resilience study. Secs. 3 and 4 present some example error control methods at the multiplex and the source levels, respectively, for H.263 video, from the viewpoint of the proposed analysis framework. The examples serve, besides demonstrating the individual methods' performance, to illustrate how the error control ability of the methods at different levels can be integrated together coherently for enhanced error resilience in the overall system. While the basic ideas of these methods have appeared elsewhere (and some material in Secs. 3 and 4 draws on earlier publications of the present authors[12,13]), their interpretation from the angle of the proposed analysis framework is novel. And all of them are standard compatible to different degrees. Finally, Sec. 6 draws the conclusion.

2. AN ANALYSIS FRAMEWORK

Fig. 1(a) and the associated four-level categorization of channel error effects provide an architectural framework of thought for classification and study of error-resilience methods. Since the earlier functional and interactivity classifications are somewhat orthogonal to this, we may consider a three-dimensional classification framework as illustrated in Fig. 1(b), with the architectural dimension serving as the primary "axis" in analysis.

Note that the architectural coordinates refer to positions in the receiver and not the transmitter. For example, in error control methods which require transmitter action (i.e., the preprocessing, the forward, and the interactive methods), the action may be a function of any one or more of the transmitter components as shown in Fig. 1(a). In this regard, we may well define a four-dimensional classification framework with the fourth dimension giving positions in the transmitter. But this causes complexity in graphical and tabular presentation, and is thus not employed in the present paper. Nevertheless, this fourth "dimension" is

Table 1. Classification of Some Bitstream Techniques

$\rightarrow$ Stages $\downarrow$ Modes	Error Detection	Error Localization	Error Concealment
Preprocessing			
Postprocessing	FEC-based error detection		FEC-based error correction
Forward	Error-detective FEC		Error-corrective FEC
Interactive			ARQ

borne in mind in the following discussion. Note also that the very definition of the coordinates along each dimension precludes the filling in of some spatial coordinates. For example, it is unlikely to have a error-resilience technique in the preprocessing mode carrying out the error detection function. Note further that, since interactive techniques are geared at reducing the error effects, along the functional dimension they may all be classified into the error concealment "stage." In principle, any forward error detection or localization technique at any architectural level can be constructed as a part of an interactive technique by associating it with a feedback path to instruct the transmitter to take proper remedial actions upon errors. A down side of interactive error control is that it is more difficult to conduct in point-to-multipoint communication than in point-to-point.

We now turn to a more in-depth discussion of the analysis framework from the architectural standpoint. Examples are provided to illustrate the positions of different error control methods in the framework. Being examples, they do not form an exhaustive list of all existing or all possible methods. In addition, the division into levels is for convenience in thought and in no way claims independence of error-resilience techniques across levels. Usually, inner-level properties provide the conditions under which the outer levels operate, and the inner levels may also inform the outer levels of the error situation for the latters' use. This is a typical way of operation even in joint-level designs, such as joint source-channel coding. Conversely, inner-level error control schemes may, in principle, employ a distortion measure at an outer level for its decision making, although such schemes seem uncommon. Schemes that iterate between inner- and outer-level processing can also be conceived.

2.1. The Bitstream Level

At the bitstream level, without regarding to the physical layer techniques (equalization, coded modulation, diversity reception for wireless transmission, etc.), the errors are normally controlled via FEC (forward error control coding) or ARQ (automatic repeat request). Such techniques are not the primary concern of the present work. Nevertheless, let us classify them along the functional and the interactivity dimensions of our analysis framework.

Error-detective FEC obviously belongs to the error detection stage. It can be classified into the postprocessing mode if, for the system under consideration, the code is specified and one is only concerned with the design or choice of a decoding method under the given code. On the other hand, it is in the forward mode if one considers using a different code than specified. Perhaps less conventionally but intuitively clear, error-corrective FEC can be classified into the error concealment stage, for it corrects transmission errors and thus alleviates error effects. As in the last case, it is in the postprocessing or the forward mode depending on whether the code is given or is to be chosen. ARQ is an interactive approach and, since it is geared at reducing the amount of bit errors, can be viewed as working in the error concealment stage along the functional dimension. In forward-mode and interactive-mode operation, the required transmitter function is normally considered as residing in the bitstream transmitter. Table 1 summarizes the above classification.

2.2. The Multiplex Level

Recall that the main task of the demultiplexer is to separate the received bitstream into a collection of FLC and VLC words. Inherently, therefore, it also checks the validity of bitstream syntax and codeword patterns. Error detection aside, a number of error-resilience methods have been introduced for this level. Since a most detrimental kind of error at this level is the loss of VLC word synchronization, which almost inevitably ends in severely degraded decoded pictures, a major goal of the error-resilience methods at this level is to enhance the synchronization property of the VLC words. Most of the proposed methods are forward in nature and fall into three categories: employing codewords with better synchronizability, using more synchronization words, and structuring of the transmission sequence of codewords and bits for better synchronizability. Examples of the first category are the reversible VLC[1,4,5] and the fixed-length entropy coding.[6] Examples of the second category are the motion boundary marker (MBM) of MPEG4[1] and the slice structured mode of draft H.263 version 2. And examples of the third category are codeword ordering and error-resilient entropy coding (EREC).[7–13,3]

Table 2. Classification of Some Error-Resilience Methods at the Multiplex Level

$\rightarrow$ Stages $\downarrow$ Modes	Error Detection	Error Localization	Error Concealment
Preprocessing			
Postprocessing	Syntax check, codeword pattern check		Statistical decoding
Forward		Reversible VLC, fixed-length entropy coding, additional sync words, codeword ordering, slotted multiplexing	
Interactive			

Architecturally, the different categories of methods have different implications on the system components. On the transmitter side, the use of codewords with better synchronizability involves a redefinition of the F/VLC. The insertion of more synchronization words may be a matter of the multiplexer only, if the bitstream syntax allows it; otherwise it requires modification of the F/VLC. And the restructuring of codeword and bit sequence requires changes to the multiplexer only. On the receiver side, they all have impact on the demultiplexer. And those affecting the F/VLC will also affect the F/VLD.

Functionally, the above techniques are primarily for error localization. The basic principles of adding synchronization words are evident and require no explanation. The technique of reversible VLC uses VLC words that have the prefix property in the reverse direction. This way, when an error is encountered in the forward direction and it is difficult to continue, reverse decoding can be started from the next synchronization point backwards so as to salvage some data previously difficult with conventional VLC.[1,4,5] Fixed-length entropy coding may be less efficient in data compression than VLC, but the use of fixed-length codewords can improve synchronizability and reduce error effects.[6] The above techniques either inserts additional codewords into the bitstream or alters the codeword set. In contrast, codeword ordering and EREC do not alter the VLC words but only reorganize the bitstream sequence. Codeword ordering refers to the sequencing of codewords for better synchronizability. Since FLC words do not lose synchronization in bit errors, when mixing FLC and VLC words together to form one bitstream segment, the synchronizability can be maximized by placing the FLC words as close to the synchronization points as allowed by the syntax as possible. EREC reorganizes the codeword and bit sequence more completely.[7] It defines a number of fixed-length slots and places the codewords into the slots in a certain order. The use of fixed-length slots gives a number of codewords well-defined starting positions and thereby reduces synchronization losses in channel errors. In view that it is a technique for multiplexing the bits generated by some encoder, we re-name it slotted multiplexing, or SM in short.

Perhaps due to its complexity, somewhat less pursued is the route of statistical decoding. An example in this category is the detection of the presence or absence of an MPEG2 start code at a bit position by hypothesis testing which attempts to minimize an expected cost.[14] Similar to bitstream error correction through FEC, this can be classified as error concealment along the functional dimension, since it not only detects or better localizes errors, but also corrects some of them to the benefit of decoded video quality. Along the interactivity dimension, it can be classified as postprocessing.

Table 2 is a summary of the above classification. An interesting issue concerning error control at the multiplex level surfaces when we note an important feature of typical error concealment techniques at the source level, namely, that these techniques often employ some kind of spatial and/or temporal interpolation. For such techniques to attain their best performance, it is desirable to have the error-corrupted video quantities dispersed over a large region rather than clustered together. Therefore, in addition to limiting the amount and extent of errors, an ideal error-resilience technique at the multiplex level should disperse as much as possible the video quantities affected by the remaining errors. In this respect, SM is an effective method.

2.3. The Codeword Level

As said, the F/VLD block converts each of the fixed- and variable-length codewords into the numerical quantities they represent. Also as mentioned, the F/VLD may perform certain checks such as forbidden FLC words, incorrect number of certain quantities, and range violation of certain quantities. Simple error concealment may be considered at this level, such as forced replacement of forbidden FLC words, clipping of quantities subject to range violation, filling of unexpected data vacancies, and truncation

Table 3. Classification of Some Error-Resilience Methods at the Codeword Level

$\rightarrow$ Stages $\downarrow$ Modes	Error Detection	Error Localization	Error Concealment
Preprocessing			
Postprocessing	Data quantity check, range violation check		Forced replacement, range clipping, vacancy filling, overflow truncation
Forward			
Interactive			

of extra data. Probably due to the straightforward nature of the F/VLD, error control at this level seems to have not received much attention. Table 3 summarizes the classification of the example methods.

2.4. The Source Level

Error-resilience schemes at the source level deal with the decoded video directly, or with the "component quantities" representing it, such as the motion vectors and/or the DCT coefficients. There is a wealth of error control techniques at this level,[3]

Consider the preprocessing methods first. They are of necessity probabilistic, for they can only conjecture where errors are likely to occur. For example, one method[15] estimates the expected distortion in a given part of a decoded picture due to noisy transmission, where the estimation is carried out assuming a known bit error probability at the bitstream level. Then intraframe coding is executed over the parts with highest expected distortion. Later in this paper, we discuss a method in a parallel spirit, but much simpler. Named length-based intra refresh (LBIR), it carries out intraframe coding on one or more macroblocks (MBs) of a predictively coded video frame where the MB(s) are selected based on the data lengths of the coded MBs in the previous frame that are used in motion-compensated prediction of the current frame. Some joint source-channel coding techniques, such as optimized quantization for transmission over noisy channels, are also of the preprocessing nature.[3] Since the primary effect of these methods is to limit the error effects in the decoded pictures, they can be classified into the error concealment "stage" along the functional dimension.

In comparison to preprocessing, there seem to be a greater number of postprocessing methods proposed. The term error concealment usually refers to such methods. In the domain of decoded video, the postprocessing methods typically conduct some kind of spatial-temporal interpolation[3] and some of them are iterative. In the domain of the component quantities, some methods to recover known lost or erroneous motion vectors have been developed.[16,17] They basically estimate a lost or erroneous motion vector from its spatial-temporal neighbors, where the estimation may take the form of simple linear or nonlinear filtering, or it may involve a search over a number of candidate vectors (constructed from filtering these spatial-temporal neighbors) for the best under a goodness measure. In the latter case, the goodness measure may be based on boundary match, meaning how close the pixel values at the boundary of the area (say, an MB) associated with the lost or erroneous vector matches its neighbors' in the decoded video. In fact, boundary-match measures may also be used to detect possible errors at the source level.

Concerning forward error control methods at the source level, most of them impact the source coder. Viewed in the context of a given standard, these methods can be further divided into two types: those which are compatible in syntax and semantics at the interface between the F/VLD and the source decoder, and those which are not. The former type usually reduces the source coder's compression efficiency and leaves some extra redundancy in its output which the decoder can exploit to reduce the error effects. However, since the data structure is standard-compatible at the F/VLD-decoder interface, a source decoder which is unable to perform the coorperative error control functions can still decode the pictures. An example of such methods is the motion vector smoothing technique in Park *et al.*[18] In this method, the motion vector of an image block is made equal to one of the motion vectors of its neighboring blocks. The decoder can capitalize on this feature to detect/localize errors in motion vectors or to recover lost/erroneous motion vectors. We consider a motion vector pairing technique in this study.

The incompatible methods alter not only the source coding algorithm but also the syntax and semantics of the data at the F/VLD-decoder interface so that, in addition to modifying the source decoder, a corresponding modification of the F/VLD or the demultiplexer is also required. To accommodate these methods into an already defined standard requires augmentation of

Table 4. Classification of Some Error-Resilience Methods at the Source Level

$\rightarrow$ Stages $\downarrow$ Modes	Error Detection	Error Localization	Error Concealment
Preprocessing			Expected-distortion-based refresh, length-based intra refresh, quantization for noisy transmission
Postprocessing	Boundary-match check		Many methods
Forward	MV smoothing, MV pairing, parity MV	MV smoothing, MV pairing, parity MV	MV smoothing, MV pairing, parity MV, leaky interframe prediction, semi-randrom reference frame, codeword ordering
Interactive			Error tracking, reference picture selection

the standard. In the context of H.263 video, an example of the incompatible methods is the technique of parity motion vectors.[19] For one out of every few frames, say N, it transmits an additional set of motion vectors for the blocks of the frame, where these vectors are obtained from motion estimation using the Nth past frame as the reference frame. The source decoder can then employ these extra motion vectors for estimation of the lost ones. The technique can be viewed as parity coding in the source domain, in contrast to usual FEC which adds parity in the bitstream domain. Incidentally, multiple description coding[3] may also be interpreted as performing parity coding in the source domain. Several methods have also been proposed for more error-robust interframe prediction. An example is leaky prediction.[16] Another is to maintain more than one past frames and semi-randomly choose one as the reference for interframe prediction.[20] This breaks the definiteness of temporal error propagation in interframe decoding. And it can be made compatible to the reference picture selection mode of H.263+.[2]Functionally, all these methods attempt to reduce the error effects and are thus of the error concealment category. Some, by infusing a parity structure into the data, can also be used for error detection and localization.

One forward error control method which does not impact the source coder is codeword ordering (by the multiplexer). Note that a feature of VLC streams is that the probability of synchronization loss increases with the distance of a codeword from the synchronization point. Therefore, ordering of the codewords such that those corresponding to the more error-sensitive quantities are placed closer to the synchronization words can achieve error concealment and improve the decoded video quality.[8,9,13]

As said, interactive error control methods can always be built on top of one-way methods. In the context of H.263, two such methods are error tracking and reference picture selection.[2,21] Error tracking employs negative acknowledgments (NAKs) to inform the encoder of the error situation. But since the roundtrip transmission delay may cover several frame periods, the source coder tracks the evolution of the footprints of each image block in motion-compensated prediction. This way, when a NAK is received, the source coder would know which parts of the current picture the errors have affected and may intraframe-encode these parts to stop further error propagation in the decoder. Reference picture selection maintains multiple reference frames in the source coder and the source decoder. The source coder is informed, via the feedback channel, of either which parts in which reference frames in the decoder are free of error (the ACK mode) or which parts in which reference frames contain errors (the NAK mode). The parts without errors are then used in motion-compensated predictive coding. These methods are geared towards reduction of error effects and thus may be classified into error concealment.

Table 4 summarizes the classification discussed above.

2.5. Introduction to the Following Discussion

The analysis framework discussed above helps clarity in thought and aids in inspiring new error-resilience methods, or combinations thereof, for video coding and transmission. We present some example designs below in the context of H.263 transmission over wireless channels. We only use the base H.263, with all options turned off. The simulation software is adapted from a publicly available package.[22] The quantizer step size of the first frame (intra-coded) is fixed at 20 and that of the subsequent frames (inter-coded) fixed at 32. We consider error control at the multiplex and the source levels. And we only consider one-way, non-interactive modes.

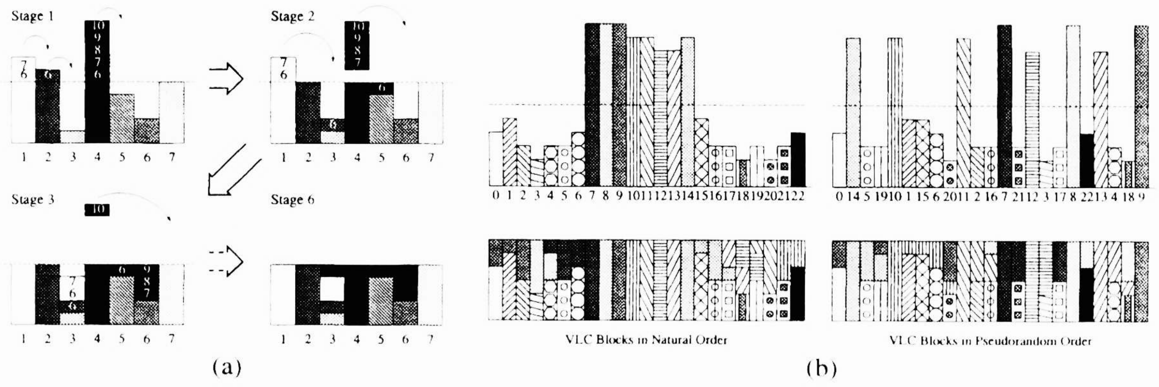


Figure 2. Illustration of SM. (a) Basics. (b) Use of pseudorandom slotting order (top: before SM, bottom: after SM.)

3. MULTIPLEX-LEVEL ERROR CONTROL

For error control at the multiplex level, recall that two important issues are the decrease of codewords subject to synchronization loss and the dispersion of the remaining error-hit image blocks for the benefit of interpolative error concealment in the source decoder. As indicated, we consider schemes based on SM (slotted multiplexing). SM organizes the bitstream to reduce the extent a bit error can dislodge VLC word synchronization. It can thus be regarded as transcoding and considered standard-compatible in this sense. A main purpose of the following discussion is to illustrate how the design of SM integrates into the overall design of error control measures spanning multiple architectural levels.

3.1. Slotted Multiplexing

Consider H.263 video. The original H.263 bitstream structure uses the PSCs (picture start codes) and the GBSCs (group-of-blocks start codes) as synchronization words. Upon an illegal VLC word due to transmission errors, a decoder may may simply give up decoding until the next PSC or GBSC, causing significant degradation in decoded video quality. SM places blocks of VLC words into fixed-length slots without (much) overhead and redundancy. The use of fixed-length slots gives each VLC block a well-defined starting position and thereby reduces synchronization losses in channel errors.

The basic SM algorithm is as follows.[7] An SM frame is composed of N slots, each of length s_i in number of bits ($i = 1, \cdots, N$). The SM frame is used to transmit N VLC blocks, each of length b_i. Obviously, we must have $\sum_{i=1}^{N} s_i \geq \sum_{i=1}^{N} b_i$. And the difference between the two sums should be minimized to minimize unnecessary transmission redundancy. One way of setting s_i is to make them equal and equal to the average length of the N VLC blocks. In our simulation, we perform one SM for each GOB (group of blocks), and thus N represents the total number of MBs in a GOB plus one for the GOB header. The slot lengths s_i must be communicated over the channel; they are assumed to be well-protected and error-free.

Fig. 2(a) shows an illustrative example of the SM coding process with seven VLC blocks of differing lengths and seven slots of $s_i = 5$ bits each. In the first stage of the process, each VLC block is assigned a separate slot. Any block with $b_i > s_i$ will leave $b_i - s_i$ bits to be placed into the slots in later stages. For the example shown, the yet-unplaced bits for each block are shifted to the right (circularly) by one slot in each stage and placed into that slot until the slot is full.

3.2. Pseudo-Randomly Ordered Slotting

For simplicity, the above example has employed a sequential order for bit placement into the slots. Other slotting orders can be considered and may be more beneficial. To see why, note first that the VLC blocks needing more stages to be fully placed and decoded are subject to higher error probabilities. This is because an error in a VLC block may cause the remainder of the block to be incorrectly decoded, which will likely be accompanied by a false identification of the end of the block, leading to further SM decoding errors in later stages. Consider again the example in Fig. 2(a). If a bit error occurs in block 3 such that the end point of this block is mis-identified, then the error will certainly propagate into block 2, and perhaps blocks 1 and 4 as well, since they all have data that follow block 3 in slot 3. Block 4 notwithstanding, the reason why block 1 stands in the error chain is that its data immediately precede a block (i.e., block 2) that overfills its slot.

Now, in videophone scenes, image blocks needing more bits to encode are often gathered together in the lower center part of the pictures where motion is typically concentrated. And the clustering together of long VLC blocks elongates the SM stages

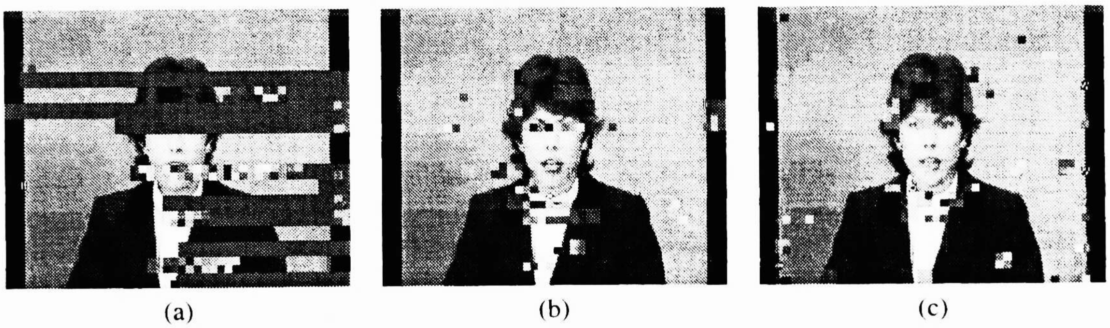


Figure 3. Decoded frame 0 of H.263-coded CIF Claire sequence after transmission over Rayleigh fading channel with average BER of approximately 0.28%. (a) No error control. (b) SM with sequentially ordered slotting. (c) SM with PR-ordered slotting.

needed to fully place them in sequentially ordered slotting. To limit the length of error propagation, one technique is to sort the VLC blocks into several groups according to their lengths and perform SM on each group separately.[8] We consider pseudo-randomizing the VLC block order so that blocks having more bits are interspersed with those having less. Fig. 2(b) gives an illustrative example, where sequential slotting is carried out on pseudo-randomly ordered VLC blocks. It is not difficult to see that, by proper transformation, this can be formulated as SM with a pseudo-random (PR) slotting order on VLC blocks in their natural order. From the figure, it can be calculated that the average number of SM stages undergone by longer-than-average blocks is 12.625 without PR slotting and 6.125 with PR slotting. Besides limiting error propagation, PR slotting also helps disperse the remaining error-corrupted blocks. For example, consider a single bit error in block 11. Without PR slotting, it may affect the decoding of blocks 10, 9, 8, and 7, in that order, while with PR slotting, only block 8. Therefore, not only fewer blocks are subject to error (i.e., better error localization), but the blocks are also at a distance apart (hence benefiting interpolative error concealment in the source decoder). Incidentally, PR slotting was also mentioned in Redmill and Kingsbury.[7]

Fig. 3 illustrates the subjective effects of SM with the two different slotting orders. Note that sequential slotting is already effective in reducing error-hit blocks, but PR slotting gains further in making the remaining error-hit blocks more dispersed.

3.3. Codeword Ordering

We noted earlier that, in mixing FLC and VLC words to form one bitstream segment, synchronization losses can be minimized by placing the FLC words as close to the synchronization points as possible. Under SM, the synchronization points are the slot beginning positions. Upon examining the H.263 syntax, we see that the DC values of intraframe-coded MBs are fixed-length coded. The transmission sequence of intra MB data is thus reordered to have DC preceding AC. Some simulation results for wireless transmission are shown in Fig. 4. The significant gain with intra-DC-first ordering at the first frame, with and without SM, shows the effectiveness of codeword ordering in protecting the fixed-length coded DC values. The average PSNR under SM with PR slotting is similar to SM with intra-DC-first codeword ordering. Joint use of these techniques yields somewhat better performance. Subjectively, without such ordering, the loss of DC values in transmission errors causes some image blocks in the decoded video to appear green or pink, which are not their correct colors. This effect is reduced with such ordering.

4. SOURCE-LEVEL ERROR CONTROL

For error control at the source level, we consider several types of methods, namely, codeword ordering, preprocessing at the source coder, and forward techniques affecting both the source coder and the source decoder.

4.1. Codeword Ordering

The sole method described earlier for source-level error control without involving the source codec is codeword ordering, which places VLC words associated with the more error-sensitive video quantities as close to the synchronization points in the bitstream as possible so as to minimize their loss in channel errors. Independently, the idea of sensitivity-based data ordering in SM has also been entertained by others.[8,9] Simulation shows that, among the various MB- and block-layer quantities, the PSNR of H.263 video is most sensitive to errors in MCBPC and it therefore must be protected well. Following MCBPC, data in decreasing order of sensitivity are CBPY, MV, DC, and AC, for the Y component. Results for Cb and Cr show a similar order, except the positions of MV and DC are switched. No result for DQUANT is obtained, for we have used a constant

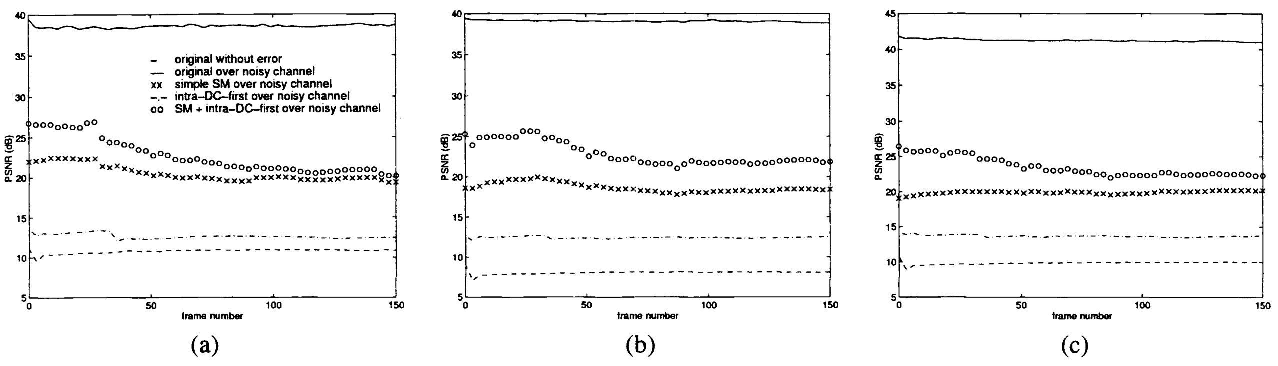


Figure 4. Performance of SM and intra-DC-first codeword ordering for coded Claire over Rayleigh fading channel at SNR = 20 dB. (a) Y component. (b) Cb component. (c) Cr component.

quantizer parameter. That MCBPC has the highest PSNR sensitivity is no surprise, since it contains structural information concerning the MB. Sensivity order of other data is also intuitively reasonable. The above order of error sensitivity largely accords with the order of MB data in H.263, except for DC. But in the last section, we have reordered DC and AC data for better synchronizability. Hence no additional codeword reordering is needed from the consideration of error sensitivity.

4.2. Preprocessing Error Control by Length-Based Intra Refresh (LBIR)

In the context of standard compatibility, preprocessing error control methods refer to fully compatible schemes operating in the transmitter without requiring the receiver's being aware of it. To proceed, recall first that H.263 employs motion-compensated interframe prediction to exploit temporal redundancy in video. In the presence of transmission errors, the reference frame used in predictive reconstruction in the source decoder may be corrupted, resulting in spatial-temporal error propagation.[21]

Spatial-temporal error propagation can be blocked by intraframe-encoding the MBs which would be predictively reconstructed using corrupted data had they been predictively coded. In preprocessing error control, therefore, the source coder has to guess where in the decoder's reference frame will likely contain corrupted data. Now, the SM technique is such that longer VLC blocks and VLC blocks that take more stages to fully slot are subject to higher error probabilites. Hence, a simple scheme is to presume the MBs in the reference frame that have been coded into more bits as more error-prone, and presume the MBs in the current frame that are predicted using these error-prone MBs as likely to suffer spatial-temporal error propagation. In our length-based intra refresh (LBIR) method, therefore, we code some such MBs in the current frame in the intra mode. More precisely, P MBs with longest data in the reference frame are found and marked. Then Q of the predictively coded current-frame MBs which have the largest overlap (after motion compensation) with each of the P previous-frame MBs are chosen and coded in the intra mode. A counteracting factor in LBIR is that, since the amount of data for intra-coded MBs are usually far greater than inter-coded ones, the forced-intra blocks may take many more SM stages to fully slot and get split into many more slots than if they were inter-coded. This may aggravate the error propagation phenomenon under SM and reduce the effectiveness of LBIR in performance improvement. Thus one should be careful in determining the number of MBs for intra refresh.

Fig. 5 illustrates the performance of LBIR and compares it with some other schemes. SM is applied in all cases. The option of periodical intra refresh refreshes three MBs per frame so that each MB could be intra-coded once per 132 frames, complying with the H.263 specification of intra-coding an MB at least once per 132 times the MB's coefficients are transmitted. Also for this reason, the LBIR was done with $P = 3$ and $Q = 1$, although this does not guarantee satisfaction of the H.263 requirement because some MBs may be refreshed more than once in 132 frames. The advantage of combined LBIR and intra-DC-first codeword ordering is apparent.

4.3. Forward Error Control by Motion Vector Pairing

As discussed, forward error control methods require the coorporation of the source decoder with the source coder to achieve the methods' best performance. Under a given standard, however, compatible methods may be developed such that decoders not equipped to perform the coorporative functions can still decode the data, only possibly with some performance penalty. The method introduced in this subsection belongs to this category.

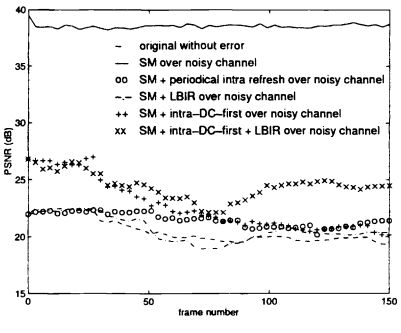
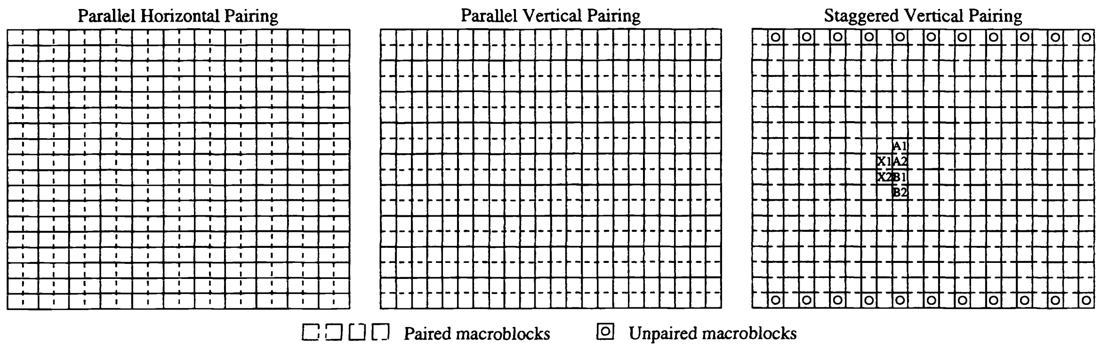


Figure 5. Performance of LBIR and other schemes for Y component of Claire over Rayleigh fading channel at SNR = 20 dB.



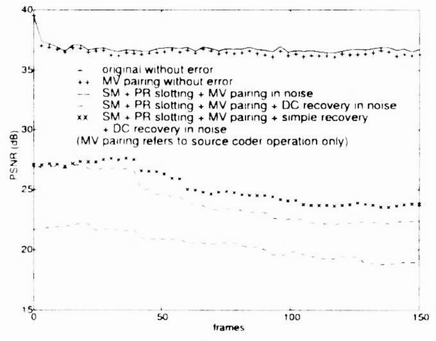
Parallel Horizontal Pairing
Parallel Vertical Pairing
Staggered Vertical Pairing
A1
X1
A2
X2
B1
B2
Paired macroblocks
Unpaired macroblocks

Figure 6. Ways of pairing motion vectors for CIF images.

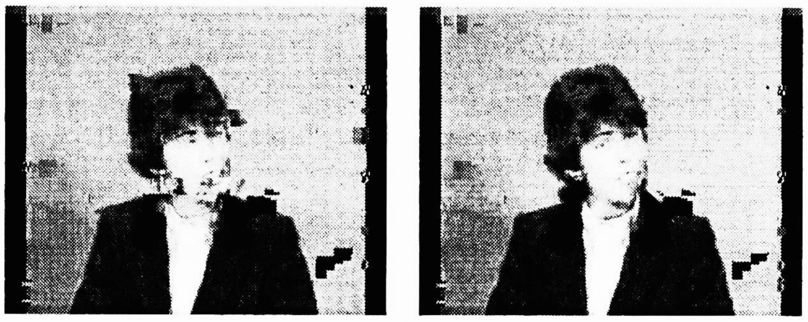
Termed motion-vector pairing, this technique basically pairs up the MBs in a frame and forces each pair to have the same motion vector, thus constituting a kind of parity coding at the source data level. This parity can be exploited in the source decoder for error detection and recovery. Such source-level parity coding also provides a guard against undetected errors through the bitstream receiver, the demultiplexer, and the F/VLD, such as corrupted but legal VLC words.

For operation under differentially coded motion vectors, an important issue is how to pair up the MBs. Two easily thought of ways are parallel horizontal pairing and parallel vertical pairing, as illustrated in Fig. 6. With parallel horizontal pairing, motion vector error in the right MB of each pair can be detected (by having a nonzero differential motion vector) but that in the left MB cannot. With parallel vertical pairing, a motion vector error will result in unequal decoded motion vectors in an MB pair and can thus be detected, but there is difficulty in determining which in the pair is in error. Staggered vertical pairing, also illustrated in Fig. 6, remedies the problem. For example, assume an isolated error in differential motion vector occurs in the MB marked X1. After differential decoding, the reconstructed motion vector of A2 will be different from that of A1, but those of B1 and B2 will be equal. From this, we can infer that the MB in error is X1, but not X2. The motion vector of X1 can thus be corrected, together with the differentially decoded motion vectors of the MBs following X1 in the GOB. This scheme works unless A2 and B1 are intraframe-coded, or X1 is an unpaired MB.

In general, the source decoder may compare the decoded motion vectors of an MB pair for error detection. If they are not equal, then one or both are in error. If one of the following MBs' motion vectors is considered correct (say, by being equal to its pair), then the present MB's motion vector is declared correct and used to replace its pair. If both motion vectors in a pair are damaged, then the source decoder can still correct them by tracing through several MB pairs to the right of the damaged pair until a correct pair is found and then tracking leftwards back to correct the errors. In our simulation, however, we have taken a simpler approach in which each of the damaged pair is replaced by its previous MB's motion vector plus the decoded differential motion vector for the current MB. Further, the damaged motion vector of an unpaired MB is replaced by the motion vector of the MB to its left. For ease of reference, the approach is termed *simple recovery*. With a corrupted motion vector, the



(a)



(b)

Figure 7. Objective and subjective performance of motion vector pairing on Claire over Rayleigh fading channel. (a) PSNR. (b) Reconstructed frame 31 (left: without simple recovery, right: with simple recovery.)

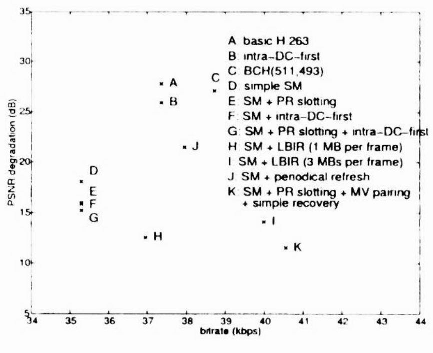


Figure 8. Comparison of the rate-distortion performance of some error control schemes. The SM rates do not include bits needed to transmit and protect slot length information, and it is assumed that SM frames need no use of synchronization codes.

MB data may also be corrupted. The recovered motion vector can be used for error concealment such as motion-compensated spatial-temporal interpolation.

The PSNR curves for the Claire sequence (Y component) under several different conditions are shown in Fig. 7(a). In the simulation, the motion vector for an MB pair has been obtained by performing conventional single-MB motion estimation for each MB and choosing from the two motion vectors the one yielding a lower total motion-compensated prediction error for the MB pair. This approach is acknowledgeably suboptimal, but adopted here for programming ease. In addition, intra-DC codewords were not placed before the VLC words and were therefore subject to error propagation effects. To conceal such errors, a simple DC recovery method was devised in which boundary-match checks were performed for the image blocks and the DC value of a block showing significant boundary mismatch was replaced by the average value of the boundary pixels of its neighbors. The details are omitted. Suffice it to say that the performance gain with DC recovery is on the order of that attainable with LBIR enhanced with intra-DC-first codeword ordering.

Fig. 7(a) shows that the use of motion-vector pairing leads to a small PSNR drop in error-free transmission. However, it yields about 1 dB PSNR gain in wireless transmission with simple recovery. The results indicate that some erroneous motion vectors did possess legal VLC words and they were recovered by the method. Visually, MBs that were compensated from incorrect positions before recovery are now almost recovered and the image quality is more pleasing, as illustrated in Fig. 7(b).

4.4. Rate-Distortion Performance

Fig. 8 summarizes the rate-distortion performance of several individual and combined error control schemes on Claire over Rayleigh fading channel. We see that schemes G, H, and K are the best performers in the rate-distortion sense.

5. CONCLUSION

We set forth a general framework for categorization of the various error control techniques for digital video transmission. The framework helps clarity in thought and aids in inspiring new error-resilience methods and combinations thereof. We then described some standard-compatible (to different degrees) methods in the context of H.263 video from the viewpoint of this framework. The discussion illustrates how the error control ability of methods at different architectural levels can be integrated together coherently for enhanced error resilience in the overall system.

REFERENCES

1. R. Talluri, "Error-resilient video coding in the ISO MPEG-4 standard," *IEEE Commun. Mag.* **36**(6), pp. 112-119, June 1998.
2. N. Färber, B. Girod, and J. Villasenor, "Extensions of ITU-T Recommendation H.324 for error-resilient video transmission," *IEEE Commun. Mag.* **36**(6), pp. 120–128, June 1998.
3. Y. Wang and Q.-F. Zhu, "Error control and concealment for video communications: A review," *Proc. IEEE* **86**(5), pp. 974–997, May 1998.
4. Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Trans. Commun.* **43**(2/3/4), pp. 158–162, Feb/Mar/Apr. 1995.
5. J. Wen and J. D. Villasenor, "A class of reversible variable length codes for robust image and video coding," in *Proc. IEEE Int. Conf. Image Processing* **II**, pp. 65–68, 1997.
6. R. Llados-Bernaus and R. L. Stevenson, "Fixed-length entropy coding for robust video compression," *IEEE Trans. Circuits Syst. Video Technol.* **8**(6), pp. 745–755, Oct. 1998.
7. 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), pp. 565–574, Apr. 1996.
8. T. Kawahara and S. Adachi, "Video transmission technology with effective error protection and tough synchronization for wireless channels," in *Proc. IEEE Int. Conf. Image Processing* **II**, pp. 101–104, 1996.
9. R. Swann and N. Kingsbury, "Transcoding of MPEG-II for enhanced resilience to transmission errors," in *Proc. IEEE Int. Conf. Image Processing* **II**, pp. 813–816, 1996.
10. R. Chandramouli, N. Ranganathan, and S. J. Ramadoss, "Adaptive quantization and fast error-resilient entropy coding for image transmission," *IEEE Trans. Circuits Syst. Video Technol.* **8**(4), pp. 411–421, Aug. 1998.
11. H.-S. Jung, R.-C. Kim, and S.-U. Lee, "On the robust transmission technique for H.263 video data stream over wireless networks," in *Proc. IEEE Int. Conf. Image Processing*, paper WA11.02, 1998.
12. Y.-L. Chen and D. W. Lin, "Error control for H.263 video transmission over wireless channels," in *Proc. IEEE Int. Symp. Circuits Syst.*, paper MPA13-7, 1998.
13. C.-T. Lee and D. W. Lin, "Combat of transmission error effects for H.263 video," in *Proc. IEEE Int. Symp. Consumer Electronics*, paper WPB1-04, 1998.
14. R. Matzner, P. Eck, and X. Changsong, "On MPEG-2 decoding of noisy input data," in *Proc. IEEE Int. Conf. Image Processing* **III**, pp. 751–754, 1996.
15. J. Y. Liao and J. D. Villasenor, "Adaptive intra update for video coding over noisy channels," in *Proc. IEEE Int. Conf. Image Processing* **III**, pp. 763–766, 1996.
16. P. Haskell and D. Messerschmitt, "Resynchronization of motion compensated video affected by ATM cell loss," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing* **III**, pp. 545–548, 1992.
17. W.-M. Lam, A. R. Reibman, and B. Liu, "Recovery of lost or erroneously received motion vectors," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, pp. V-417–V-420, 1993.
18. C. S. Park, J. Ye, and S. U. Lee, "Lost motion vector recovery algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.* **3**, pp. 229–232, 1994.
19. C.-S. Kim, R.-C. Kim, and S.-U. Lee, "Robust transmission of video sequence over noisy channel using parity-check motion vector," *IEEE Trans. Circuits Syst. Video Technology* **9**(7), pp. 1063–1074, Oct. 1999.
20. M. Budagavi and J. D. Gibson, "Error propagation in motion compensated video over wireless channels," in *Proc. IEEE Int. Conf. Image Processing* **II**, pp. 89–92, 1997.
21. E. Steinbach, N. Färber, and B. Girod, "Standard compatible extension of H.263 for robust video transmission in mobile environments," *IEEE Trans. Circuits Syst. Video Technology*, **7**(6), pp. 872–881, Dec. 1997,
22. Telenor Research, H.263 TMN-test model software codec version 2.0, ftp://bonde.nta.no/pub/tmn/software/tmn-2.0.tar.gz and tmndec-2.0.tar.gz.