Redmill, DW., & Kingsbury, NG. (1996). The EREC: an error-resilient technique for coding variable-length blocks of data. *IEEE Transactions on Image Processing*, *5*(4), 565 - 574. [4]. https://doi.org/10.1109/83.491333

Peer reviewed version

Link to published version (if available):
10.1109/83.491333

Link to publication record in Explore Bristol Research
PDF-document

University of Bristol - Explore Bristol Research
General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available: http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/

The EREC: An Error-Resilient Technique for Coding Variable-Length Blocks of Data

David W. Redmill and Nick G. Kingsbury, *Member, IEEE*

Abstract—**Many source and data compression schemes work by splitting the input signal into blocks and producing variable-length coded data for each block. If these variable-length blocks are transmitted consecutively, then the resulting coder is highly sensitive to channel errors. Synchronization code words are often used to provide occasional resynchronization at the expense of a some added redundant information.**

This paper introduces the error-resilient entropy code (EREC) as a method for adapting existing schemes to give increased resilience to random and burst errors while maintaining high compression. The EREC has been designed to exhibit graceful degradation with worsening channel conditions. The EREC is applicable to many problems and is particularly effective when the more important information is transmitted near the start of each variable-length block and is not dependent on following data.

The EREC has been applied to both still image and video compression schemes, using the discrete cosine transform (DCT) and variable-length coding. The results have been compared to schemes using synchronization code words, and a large improvement in performance for noisy channels has been observed.

I. Introduction

MANY compression algorithms, such as JPEG [1], MPEG [2], H261 [3], and others work by splitting the input signal into fixed blocks of data. These blocks of data are coded using a variety of techniques including DCT, quantization, zig-zag scanning, run-length, and Huffman coding to produce a variable-length block of coded data for each block. Having done this, the variable-length blocks must be multiplexed together and transmitted. This process, which has been labeled in Fig. 1 as multiplex coding and decoding, involves combining all the data to be coded into a form that is both suitable for transmission (e.g., a single binary bit-stream) and can be decoded by the receiver. This stage of the coding process may also include channel coding and the addition of synchronization code words that improve the system performance over noisy channels. The most common way to implement the multiplex coding is to transmit the variable-length blocks consecutively. If this method is adopted, then the compressed data is highly sensitive to the propagation of channel errors. This is due to errors that cause the decoder to lose synchronization with the start and end of blocks. Even in codes where synchronization is regained after the loss of a few blocks [4], [5], the block count is likely to be permanently offset, resulting in the following data being decoded at the wrong location. For some applications, e.g., speech, this may not be a problem as it may only cause a temporal shift in the following data. However, for other applications such as still image and video coding, this can result in large areas of the picture being spatially displaced.

Manuscript received August 18, 1993; revised April 4, 1995. The associate editor coordinating the review of this paper and approving it for publication was Prof. Michel Barlaud.

D. W. Redmill is with the Centre for Communications Research, Department of Electrical and Electronic Engineering, University of Bristol, Bristol, U.K. (e-mail: David.Redmill@bristol.ac.uk).

N. G. Kingsbury is with the Signal Processing and Communications Laboratory, Cambridge University Engineering Department, Cambridge, U.K. (e-mail: ngk@eng.cam.ac.uk).

Publisher Item Identifier S 1057-7149(96)02752-2.

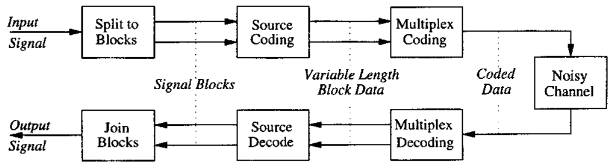


Fig. 1. Source coding by splitting the signal into blocks.

A common solution to this problem is to include at intervals some relatively long unique synchronization code words (sync words) followed by some block address information. These sync words involve the addition of redundant information. Although the necessary length of these sync words has been minimized [6], [7], the nature of the constraints imply that they are still quite long, and thus, in order to maintain low redundancy, they must be used relatively infrequently. These techniques limit the propagation of errors to a maximum of the separation between sync words, which, due to the relative infrequent occurrence of sync words, is still quite large.

Another method that can be applied either alone or in conjunction with sync words or other techniques is error correction coding (ECC). This involves the addition of extra redundant information (parity bits), which degrades the compression efficiency. ECC is effective for dealing with channels with known bit error rates (BER's) and limited length bursts; however, it fails catastrophically with increased BER or longer bursts due to interference or signal fade [8].

Another alternative is to use custom-designed error-resilient schemes [9]–[13] that employ a combination of methods that usually use fixed-length code words. These methods require additional design effort over the more standard schemes and often result in lower compression efficiency.

The error-resilient entropy code (EREC), which was first outlined in [14], can be used to multiplex the variable-length blocks of data produced by standard compression algorithms into an error-resilient structure. The algorithm works by reorganizing the variable-length blocks such that each block starts at a known position within the code. This

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.

means that the decoder is automatically synchronized at the start of each block. This is achieved with minimal additional redundancy.

In a noisy channel environment, the EREC does not suffer from loss of synchronization and large or catastrophic failure typical of many of the above schemes, but rather its performance degrades gracefully with increasing BER. This has many applications in source coding systems for noisy and unpredictable channels where perfect reconstruction is not essential, error protection is too expensive in terms of added redundant information, and some loss of fidelity is preferable to complete breakdown.

The property of graceful rather than catastrophic degradation of a coding scheme, as channel error rate increases, is important for several reasons. First, it provides users with a warning of difficult conditions, allowing them to take steps to improve matters by rerouting the channel, changing the frequency of a radio link, or adjusting the position of the receiver. Second, in the presence of bursts of interference, channel coding schemes often fail badly if the depth of interleaving is insufficient to cope with the duration of the burst, whereas deep interleaving can introduce unacceptable delays to a system. An error-resilient source code, on the other hand, produces disruption only approximately as long as the burst and allows the user to make maximum sense of the uncorrupted received data. Hence, error-resilient source coding may well be more user friendly than channel coding. Example applications include the transmission of speech, images, or video over cellular networks, weak radio links, or noisy telephone lines.

The paper is arranged as follows. Section II describes how the EREC algorithm reorganizes data into an error-resilient structure. Section III examines the performance of the EREC algorithm and discusses the propagation of channel errors. It is shown that the algorithm gives a lower expected number of bits affected per bit error than schemes using sync words. Section IV discusses the overheads and redundancy involved in using the EREC. In Section V, the EREC is applied to a typical image coding scheme using DCT and variable-length coding. The results are compared with the corresponding standard scheme and a scheme using synchronization codewords at regular points. In Section VI, the EREC is applied to produce a video coding scheme based on H.261. The results are compared with the corresponding scheme using H.261 synchronization code words. Section VII suggests improvements and modifications to the EREC and discusses further applications.

II. The EREC Algorithm

The aim of the EREC algorithm is to provide an error-resilient scheme for coding the data produced by many block-based data compression algorithms.

The EREC is applicable to block coding strategies where the input signal is split into blocks that are coded as variable-length blocks of data. Each variable-length block must be a prefix code, which means that in the absence of channel errors the decoder knows when it has finished decoding a block without reference to any previous block or following

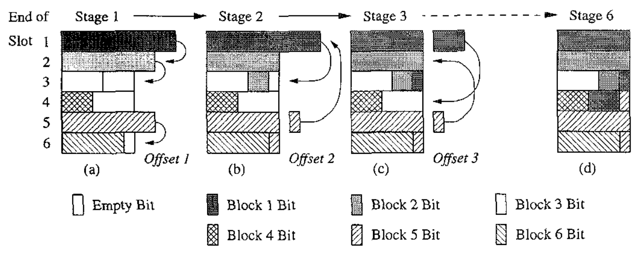


Fig. 2. EREC bit reorganization algorithm.

information. The EREC also assumes that the information contained within each variable-length block is causal in the sense that channel errors affect current and following data only. These assumptions are met by most entropy-based source coding schemes such as Huffman coding or arithmetic coding [15].

The EREC frame structure is composed of N slots of length s_i bits to give a total length $T = \sum_{i=1}^{N} s_i$ bits to transmit. It is assumed that both the encoder and the decoder know the values of s_i, T, and N. Thus, the N slots of data can be transmitted consecutively without risk of any loss of synchronization. Each EREC frame can be used to transmit up to N variable-length blocks of data, each of length b_i, provided the total data to be coded does not exceed the total T bits available.

$$R = T - \sum_{i=1}^{N} b_i \geq 0 \qquad (1)$$

The difference R (between the total data to be coded and the data transmitted) represents unnecessary redundant information. In order to minimize this redundancy, a suitable value of T must either be known in advance or transmitted as highly protected side information. The slot lengths s_i can be predefined as a function of T and N and, therefore, do not need to be transmitted. For larger signals, it is necessary to split the signal into several groups of N blocks, each of which can be coded using the EREC. The choice of these parameters and their transmission is discussed in greater detail in Section IV.

The EREC places the variable-length blocks of data into the EREC code structure by using a bit reorganization algorithm that relies only on the ability to determine the end of each variable-length block. Thus, in the absence of channel errors, the bit reorganization algorithm can be exactly followed by the decoder. This algorithm is a development of that used by the error-resilient positional code (ERPC) [9].

Fig. 2 shows a simple example of the operation of the reorganization algorithm with six blocks of lengths 11, 9, 4, 3, 9, 6, and 6 equal length slots $s_i = 7$ bits.

The first stage of the algorithm is to allocate each block of data to a corresponding code slot. Starting from the beginning of each variable-length block, all or as many bits as possible are placed within the corresponding slot. Thus, blocks with an equal number of bits to those available in the corresponding slot, i.e., $b_i = s_i$, are coded completely, leaving the slot full. Blocks with $b_i < s_i$ are coded completely, leaving the slot with $s_i - b_i$ bits unused. Blocks with $b_i > s_i$ have s_i bits coded to fill the slot and leave $b_i - s_i$ bits remaining to be

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.

coded. For the example of Fig. 2, blocks 1, 2, and 5 have $b_i > s_i$, and blocks 3, 4, and 6 have $b_i < s_i$. Thus, at the end of stage 1 (Fig. 2(a)), slots 1, 2, and 5 are full, whereas slots 3, 4, and 6 have space left to code data for blocks 1, 2, and 5.

In subsequent stages of the algorithm, each block with data still to be coded searches for slots with space remaining. At stage n, block i searches slot $i+\phi_n$ (modulo N), where ϕ_n is a predefined offset sequence. If the slot searched by a block has some space available, then all or as many bits as possible are placed within that slot. For the example of Fig. 2, the offset sequence is 0, 1, 2, 3, 4, 5. Thus, at stage 2, the offset is 1, and each block searches the following slot. Fig. 2(b) shows how, at the end of stage 2, one bit from block 5 has been placed in the space remaining in slot 6 and two bits from block 2 have been placed in slot 3.

Provided (1) is satisfied, then there is enough space to place all the required data, and all the bits will be placed within N stages of the algorithm. Fig. 2(d) shows that by the end of stage 6, all the data has been placed.

In the absence of channel errors, the decoder can follow this algorithm by decoding the variable-length block data as it goes along until it detects the end of each block, at which point, it knows that the remaining contents of the slot correspond to data from other blocks placed at a later stage in the algorithm.

In [9], the choice of offset sequence was discussed, and a pseudo-random offset sequence was shown to be preferable. The advantages of a pseudo-random offset sequence stem from its uncorrelated nature: $\phi_{n+k} - \phi_n$ is independent of n, and thus, two blocks k apart will not search slots in the same order.

III. Error Propagation Properties

The precise analysis of the error propagation properties of the EREC coding structure is a complex problem and is highly dependent on the propagation properties of the variable-length coding used to create each variable-length block.

In order to gain some knowledge of the propagation of errors within the EREC structure, we first examine how the encoder places information at each stage. We assume that the data is distributed randomly among all blocks so that the block statistics are independent of block numbers.

A. Bit Placement

Let $p_k(x)$ be the probability of a block having x bits still to be coded after k stages of the EREC algorithm, and let $q_k(y)$ be the probability of a slot having y empty bits after k stages. Further, let P_k be the marginal probability of a block having some data still to be coded at stage k, and let Q_k be that of a slot having some space.

$$P_k = \sum_{j=1}^{\infty} p_k(j) = 1 - p_k(0) \tag{2}$$

$$Q_k = \sum_{j=1}^{\infty} q_k(j) = 1 - q_k(0) \tag{3}$$

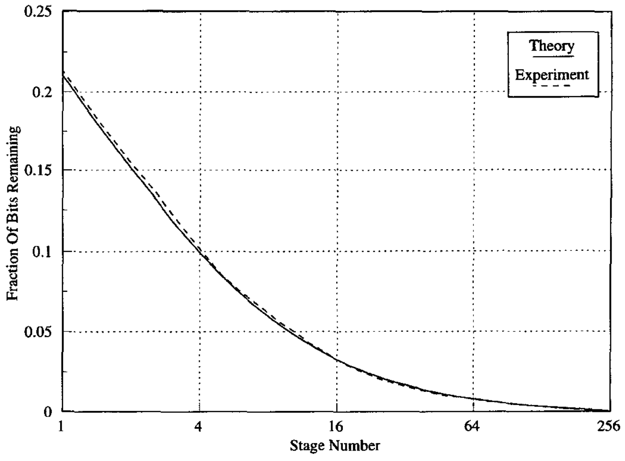


Fig. 3. Fraction of unplaced bits.

We can find expressions for $p_{k+1}(x)$ and $q_{k+1}(y)$:

$$p_{k+1}(x) = p_k(x)\{1 - \gamma Q_k\} + \gamma \sum_{j=1}^{\infty} p_k(x+j) q_k(j) \tag{4}$$

$$q_{k+1}(y) = q_k(y)\{1 - \gamma P_k\} + \gamma \sum_{j=1}^{\infty} q_k(y+j) p_k(j) \tag{5}$$

$$\gamma = \left(\frac{N}{N-k}\right) \tag{6}$$

The first term of (4) represents the probability of a block with x bits left over finding a full slot, whereas the second term represents the probability of a block with $x + j$ bits finding a slot with j bits free. Equation (5) applies similarly to the probability of a slot ending up with y empty bits. The term γ takes into account the knowledge that all the slots searched so far by a searching block are full, and thus, the probability of finding a partially empty slot is increased.

These equations can be iteratively solved to predict the number of bits B_k remaining unplaced at stage k

$$B_k = N \sum_{i=1}^{\infty} i.p_k(i) \tag{7}$$

and the number of bits coded at stage k as

$$B_k - B_{k-1}. \tag{8}$$

We have measured the initial probabilities $p_0(x)$ and $q_0(x)$ for real data derived from prediction error image coding with 256 blocks. This is a data source that is known to have highly variable data lengths from block to block. Values for $p_k(x)$ and $q_k(x)$ have been found by iterative use of (4) and (5). From this, a prediction of the number of bits still to be placed at stage k has been found. Fig. 3 shows that this prediction approximates very well the experimentally measured values, confirming the validity of the above analysis.

B. Error Propagation

In this section, we attempt to analyze how channel errors may propagate within the EREC structure, given the above

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.

placement statistics. We present a simple worst-case analysis in order to provide an upper bound on the fraction of bits affected by channel errors.

In order for the EREC to decode the coded data, the decoding algorithm needs to know when it reaches the end of each block (EOB).

At the end of stage k, let $\hat{b}_i(k)$ be the number of bits still to be decoded from block i and $\hat{s}_i(k)$ be the number of bits remaining to be decoded from slot i for the error-free case. Let $s_i(k)$ and $b_i(k)$ be the same quantities with errors taken into account. An error within a block may cause the EOB to be missed or falsely detected so that $b_i(k) \neq \hat{b}_i(k)$. The actual value $b_i(k)$ is then unpredictable. This will cause all remaining data for that block to be incorrect, and the block is termed as erroneous for the current and subsequent stages in the decoder algorithm. An erroneous block may cause the corresponding slot to be incorrectly decoded, i.e., $s_i(k) \neq \hat{s}_i(k)$. These slots are then said to be in error. At later stages within the EREC algorithm, erroneous blocks may cause partially empty slots to become erroneous, and erroneous slots can also cause searching blocks to become erroneous. Note that full slots and fully decoded blocks cannot be affected.

The analysis that follows involves the following assumptions:

1) Any erroneous block will remain erroneous.
2) Any erroneous slot will remain erroneous.
3) A correct searching block will become erroneous if it searches an erroneous slot.
4) A correct slot with space will become erroneous if it is searched by an erroneous block.
5) All blocks and slots are statistically equivalent; therefore, the amount of remaining data $\hat{b}_i(k)$ is independent of whether the block is erroneous and the block number i.

Let E_k represent the total probability of a block being in error at stage k for all blocks, and let C_k be the probability that a block is both erroneous and still searching $b_i(k) \neq \hat{b}_i(k)$ & $\hat{b}_i(k) > 0$). Similarly, let F_k be the total probability of a slot being in error at stage k for all slots, and let D_k be the probability of a slot being erroneous and still have space $(s_i(k) \neq \hat{s}_i(k)$ & $\hat{s}_i(k) > 0)$.

We can derive iterative equations as follows:

$$C_{k+1} = C_k + (1 - C_k).\gamma F_k \tag{9}$$
$$D_{k+1} = D_k + (1 - D_k).\gamma E_k \tag{10}$$
$$E_{k+1} = E_k + (1 - C_k)P_k.\gamma F_k \tag{11}$$
$$F_{k+1} = F_k + (1 - D_k)Q_k.\gamma E_k. \tag{12}$$

The term E_k on the right-hand side of (11) refers to blocks that were erroneous and remain erroneous from assumption 1. The term $(1 - C_k)P_k$ is the probability of a block searching and not being erroneous. These are the only blocks that can be affected by erroneous slots. The term γF_k is the probability that the block searches an erroneous slot and becomes erroneous from assumption 3. (γ takes into account the knowledge that all previously searched slots must have been filled and correct when they were searched and are therefore still correct.)

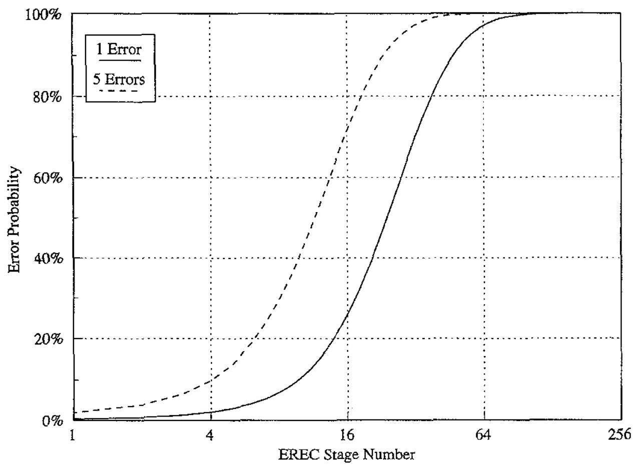


Fig. 4. Probability of a bit placed suffering from channel error propagation effects for an average of one and five errors.

For (9), C_k refers to a probability of a searching block being erroneous, and $(1 - C_k).\gamma F_k$ is the probability of the block becoming erroneous. Thus, the right-hand side of (9) refers to the probability of a block searching at the end of stage k being erroneous at the end of stage $k + 1$. From assumption 5, the probability of searching blocks being fully placed is independent of whether they are erroneous, and thus, C_{k+1} (the probability of a searching block at the end of stage $k + 1$ being erroneous) is given by the right-hand side of (9).

Equations (10) and (12) are similar to (9) and (11).

The number of bits affected by channel error propagation for each stage can be found by multiplying C_k (the probability of a searching block being in error) by the number of bits placed at stage k.

Equations (9)–(12) have been iterated starting from initial conditions of M blocks and the corresponding slots being in error. Fig. 4 shows how C_k increases as the algorithm progresses. Fig. 5 shows how the total number of bits affected increases with the number of errors M introduced. We have compared this to values calculated for a scheme using 16 sync words in each frame of 256 blocks.

The analysis presented is worst case and assumes that all erroneous blocks E_k have the wrong number of bits remaining and that all erroneous slots F_k have the wrong number of bits free at all stages. In practice, when slots become full $s_i(k) = 0$ and blocks fully decoded $b_i(k) = 0$, the above assumptions are not valid as it is likely that they are now correct in the sense that $s_i(k) = \hat{s}_i(k)$ and $b_i(k) = \hat{b}_i(k)$. The analysis also assumes that all channel errors occur at the beginning of blocks and thus cause maximum damage, whereas the values calculated for the sync word scheme assume an even distribution.

As the algorithm progresses from stage to stage, the number of unfilled slots in error will increase, and the data placed at later stages of the algorithm are more likely to suffer from the propagation of channel errors. From this, it can be seen that the bits most likely to suffer from channel error propagation are those near to the ends of blocks with large amounts of data. In the case of image and video coding (Sections V and VI), this

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.

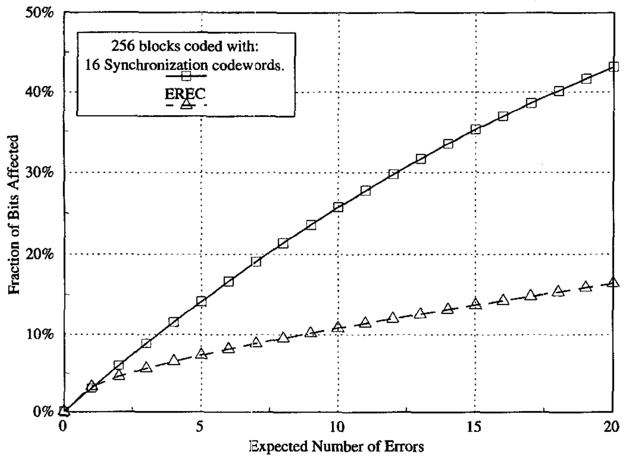


Fig. 5. Total propagation of channel errors.

data typically corresponds to high-frequency information from blocks with high activity. Thus, most of the effects of channel error propagation will be seen as high-frequency errors in high activity regions of the image. Subjective testing has shown that distortions in these regions of an image are significantly less noticeable than errors in low activity regions, and therefore, we expect a subjective benefit as well [16].

It is further noticed that if two or more errors occur, then the total number of bits affected will not be much greater than that for one error. This means that the EREC is well equipped to degrade gracefully at higher error rates.

IV. Overhead Information and Total Redundancy

For the decoder to operate correctly, it must know the parameters T and N that specify the EREC code structure.

In order to satisfy both the inequality of (1) and minimize the redundancy R, T should be chosen to be close to and not less than $\sum b_i$. Since $\sum b_i$ is unlikely to be known in advance, it may be necessary to transmit a suitable value of T to the decoder using a small amount of highly protected side information.

The cost $C(T)$ in bits of transmitting T must be viewed as added redundant information and added to R to give the total redundant information R_t as $R_t = C(T) + R$. If T is finely quantized, then $T - \sum b_i$ is small, but $C(T)$ is larger, whereas coarsely quantizing T will make $T - \sum b_i$ larger and $C(T)$ smaller.

To gain some knowledge of the typical redundancy, we consider the system simulated in Section V. For a 256 by 256 pixel image coded between 0 and 1 bits per pixel, the ideal T requires 16 bits. Quantizing this to steps of 16 bits and using a (32, 6) error correction code[1] gives an expected redundancy of $\frac{32}{6} \times 12 + 8 = 72$ bits. As a comparison, a scheme using efficient end-of-line sync words [7] requires approximately 190 bits.

The encoder often has some of control over compression achieved and, thus, can control the value of $\sum b_i$. It is also

[1] This code has a Hamming distance of 16 and provides adequate protection for up to 5% BER. This has been used in [17]–[19].

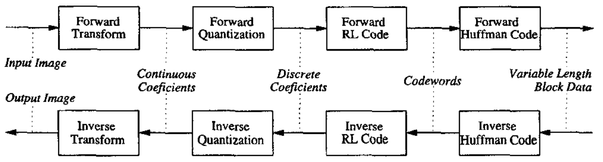


Fig. 6. Image coding using Huffman coding of transform blocks.

possible to selectively ignore small unimportant coefficients in order to make fine adjustments to $\sum b_i$ and, thus, lower the redundancy.

The number of EREC slots N can usually be fixed in advance and, thus, need not be transmitted. However, some applications, such as prediction error coding, produce a sparse set of active blocks (containing data). In these cases, it is useful to keep N close to (but not less than) the number of active blocks. By doing this, the slot lengths s_i are higher, and a greater error resilience can be achieved. (Section VI describes how the positions of the active blocks can be transmitted.) Thus, it may be useful to transmit a value for N in a similar way to T. Once the values of T and N have been determined, both the encoder and decoder can calculate the slot lengths s_i in a deterministic manner (e.g., even length slots.)

A. Implementation Issues

The EREC algorithm requires that all the data for the N blocks be known before commencement of the algorithm. This implies significant memory requirement and delay. The complexity of the EREC also increases with the number of blocks N and is found to be of order $N\log(N)$ for an efficient implementation [17]. The simplest solution to these problems is to split large signals into several subframes of N blocks each. In this way, N can be reduced at the expense of slightly more redundant header information.

V. Application to Image Coding

In order to examine the effectiveness of the EREC for real coding schemes, we have simulated a typical block-based image compression algorithm (which is similar to H.261 intraframe mode) using Huffman coding of quantized transform coefficients for 8×8 pixel blocks, as shown in Fig. 6.

First, the image is split into blocks of 8×8 pixels. These blocks are transformed using an eight-point DCT separably in the horizontal and vertical directions. The 8×8 transform coefficients are quantized using a linear quantizer with a double-width center band. The quality of the coded picture and the compression can be varied by adjusting the step size of the quantizer. The 8×8 quantized coefficients are scanned in a zig-zag order from low frequencies to high frequencies. The nonzero coefficients for each block are run length coded to produce data words that consist of run-length and level. These data words together with an end-of-block (EOB) word are coded as variable-length code words using Huffman coding. Thus, each variable-length block consists of a number of variable-length code words terminated by an EOB code word. We have used an 8-bit word for the dc coefficient followed

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.

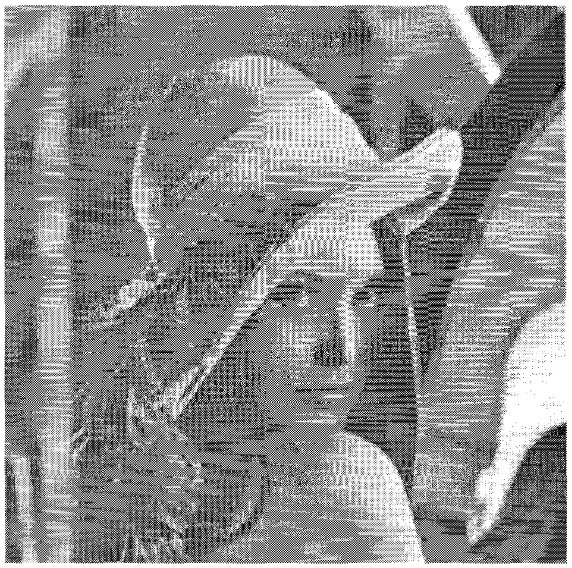


Fig. 7. Original 'Lenna.'

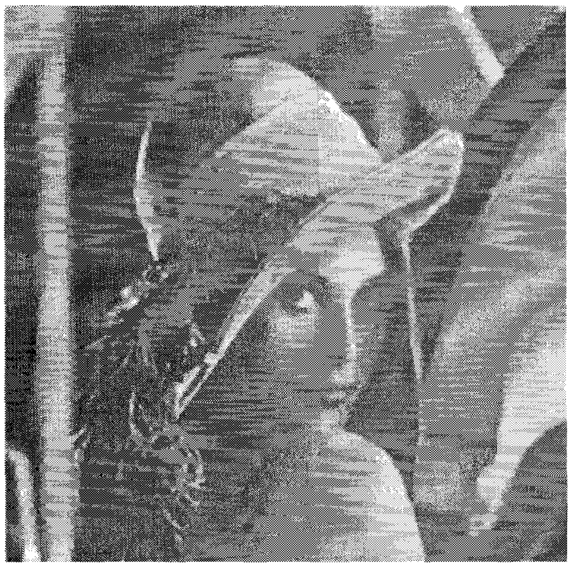


Fig. 8. Coded at 0.65 b/pixel.

by the Huffman codes specified by H.261 [3] for the ac coefficients.

Due to both the presence of the EOB code word and the prefix nature of Huffman coding, each variable-length block satisfies all the constraints for the EREC outlined at the start of Section II.

The scheme above has been simulated with various transmission methods:

- each block consecutively with no synchronization codes
- using a sync word and line number at the start of each line of blocks
- using the EREC coding structure. In this case, N is chosen according to image size, and a suitable value of T together with image size and quantizer setting must be transmitted to the decoder as protected header information. This is considered more fully in [17] and [18]. For the example shown, $N = 1024$, $T = 42\,608$ and $s_i = \{41, 42\}$.

For these simulations, we have ignored the effect of any error concealment strategies and error correction coding. These techniques can be used in conjunction with all of the above methods to further improve error resilience.

Fig. 7 shows the original "Lenna"[2] image. Fig. 8 shows "Lenna' at 0.65 b/pixel. Fig. 9(a) demonstrates how a single bit error can affect the synchronization of all the following data. Fig. 10(a) shows that the use of end of line sync-words can limit this propagation. Fig. 11(a) shows that the EREC structure can further limit this propagation. (Note that the error is in the same bit for Figs. 9(a), 10(a), and 11(a).) Figs. 9(b), 10(b), and 11(b) show typical decoded versions from the three schemes at a random bit error rate (BER) of 0.1%.

VI. Application to Video Coding

To further demonstrate how the EREC can be applied to a more complicated coding scheme, we have applied the EREC to an H.261 type of video coder [3].

[2]"Lenna" is an 8-b 256 by 256 pixel monochrome still image.

A. Simulation Details

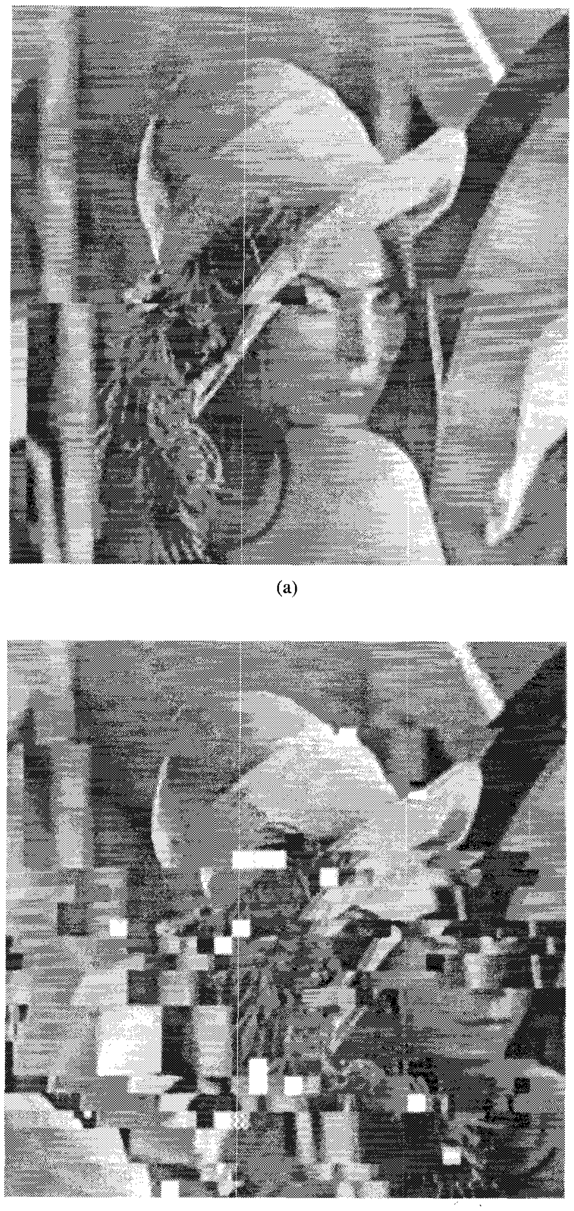
We have simulated a standard coding scheme by slightly adapting H.261 as follows:

- The predictive coding of motion vectors has been removed in order to improve the error resilience of their coding.
- We ignore the error correction coding specified. It is noted that the error correction coding can equally well be applied to the output of the EREC.
- We ignore the frame and group of blocks (GOB) header information with the exception of the specified sync word.
- The quantizer has been fixed in advance for the whole sequence, and a macro-block quantization code-word (MQUANT) has thus been omitted.
- For standard H.261, the GOB's have been changed to be 8 by 4 macro-blocks (MB's) in order to code test sequences of 256 by 256 pixels.
- In order to code monochrome sequences, the conditional block pattern (CBP) code has been replaced by a simple 4-bit mask (one bit for each luminance block).

In order to limit the temporal propagation of channel errors, we have used intraframe coding for $\frac{1}{16}$th of each frame. Thus, after 16 frames, all portions of the picture will be updated.

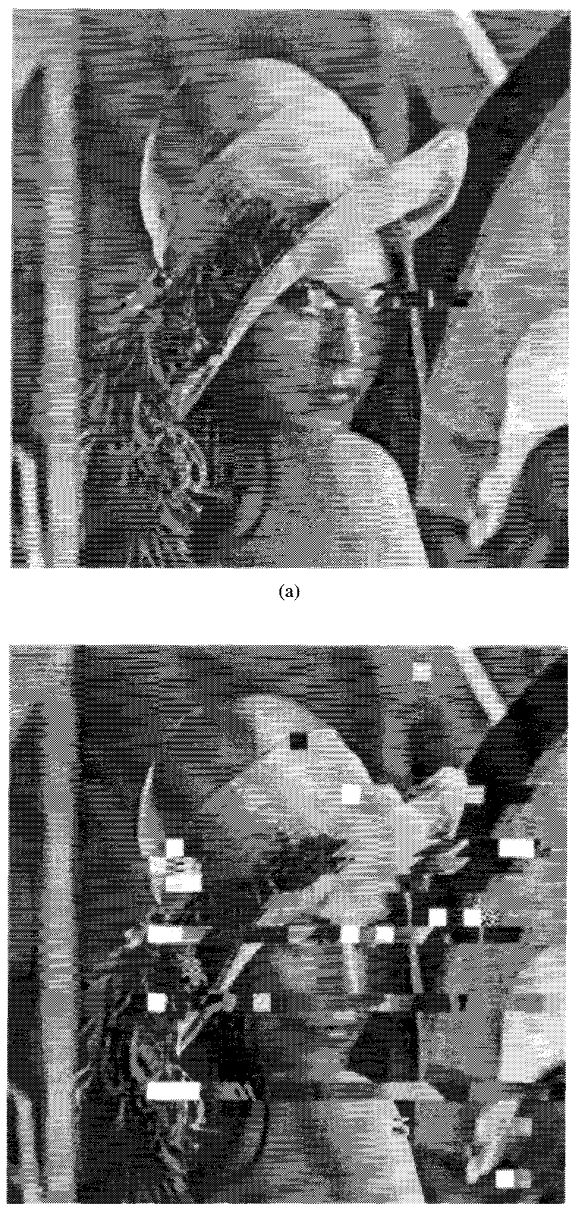
The EREC has been applied to above scheme as a two-stage process. First, the synchronization codes are ignored, leaving only the MB layer and block layer to be coded. The first stage involves coding the MB layer. Before doing this, the use of macro-block address (MBA) codes specified in H.261 is replaced by a single bit for each MB (a bit map). For the standard system, this is equivalent to using MBA codes that are a string of zeros for each inactive MB terminated by a 1 for the next active MB. This change has not been used in the standard system since it offers no advantage (unless the MBA bit map was coded separately) and lengthens the MBA codes by an average 0.1 bit. For use with the EREC, the bit

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.


(a)

(b)

Fig. 9. Standard system with no resynchronization. (a) Single error. (b) 0.1% BER.


(a)

(b)

Fig. 10. System with extra end of line synchronization code-words. (a) Single error. (b) 0.1% BER.

map approach ensures that a valid variable-length code of at least 1 bit exists for all MB's. Each MB is then assigned to an EREC slot, and the MB data is coded by applying the EREC algorithm. The second stage is to code the block layer in a similar fashion. In order to do this, each block is assigned to an EREC slot by use of the EREC placement algorithm to find unoccupied slots. Block data is then coded by further use of the EREC algorithm.

The number of slots N is chosen to be at least the number of blocks containing prediction error information with a minimum of total number of MB's. Suitable values of N and T must be transmitted to the decoder as protected header information with each frame.

H.261 specifies an extra sync word at the start of each frame. This has been kept to provide the EREC decoder with some framing information. Note that during normal operation, the EREC knows the size of each frame and can predict the location of this code. Thus, its correct transmission is not as important as for the H.261 system.

B. Simulation Results

We have simulated both the slightly adapted H.261 scheme and the EREC with the 'Claire' sequence[3] at about 80 Kbits (8000 bits/frame.)

[3] 'Claire' is a monochrome 68 frame 256 by 256 pixel 8-bit sequence at 10 frames/s.

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.

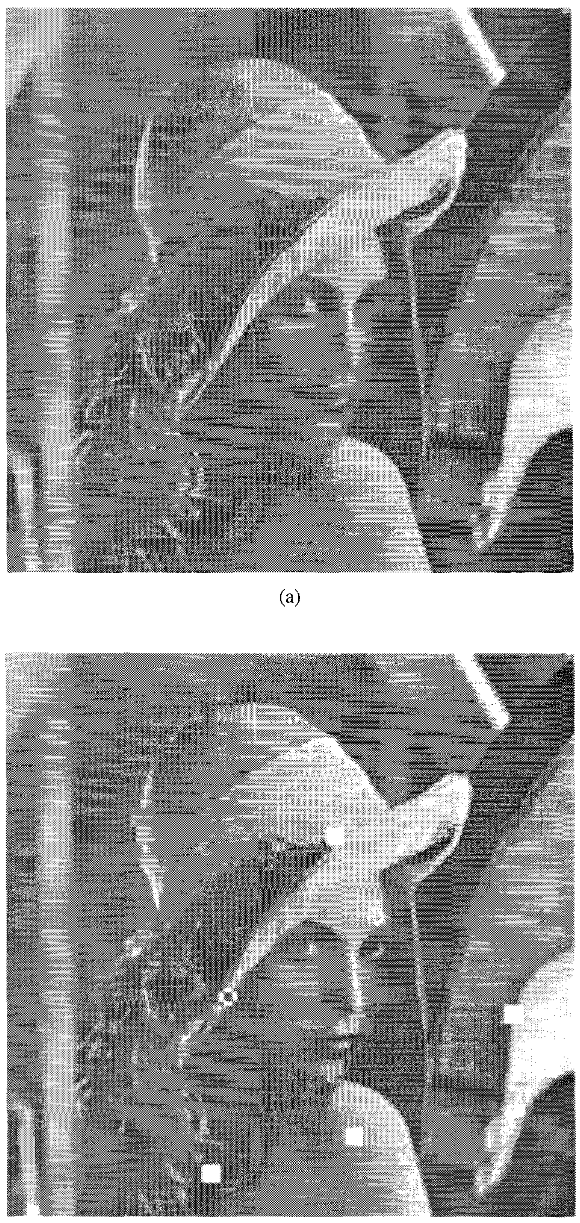


Fig. 11. System coded with the EREC. (a) Single error. (b) 0.1% BER.

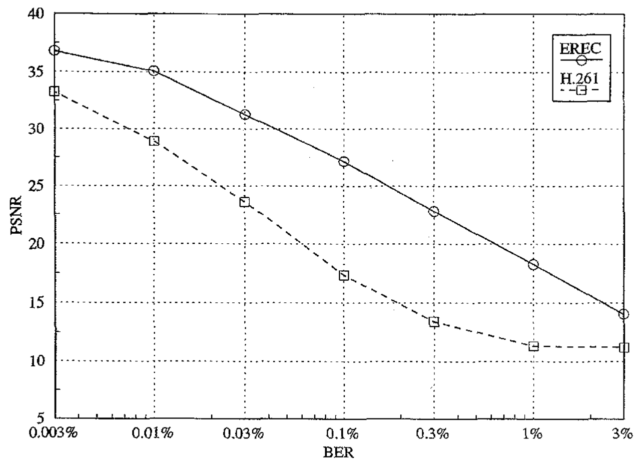


Fig. 12. Variation of PSNR with BER.

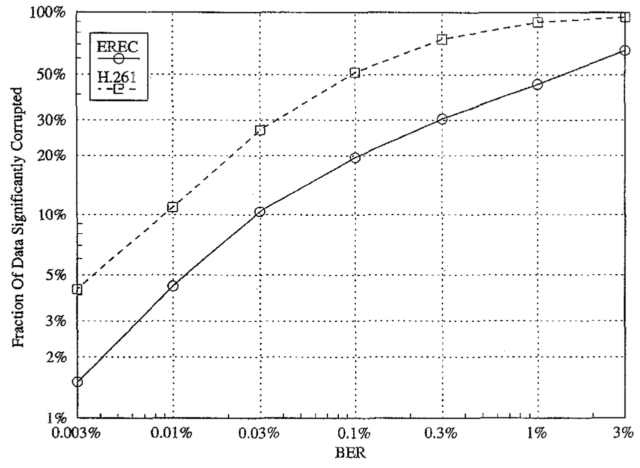


Fig. 13. Variation of fraction of corrupted data with BER.

The modification to the MBA code introduced by the EREC introduces a redundancy of 20.7 bits/frame. The necessary side information for the EREC is about 100 bits, depending on the error correction used. This is more than canceled by the removal of the GOB sync word, which represents 160 bits/frame.

Both schemes have been simulated for a range of bit error rates. Fig. 12 shows the peak signal-to-noise ratio (PSNR) at various bit error rates. For high BER, the PSNR appears to level off at about 12 dB. This is found to correspond to a complete loss of signal. The misleading value is due to the incorrect assumption of additive noise used to calculate PSNR. Due to the potential dominance of isolated errors, PSNR is not a reliable measure of image quality. Fig. 13 shows the fraction of blocks that are significantly corrupted at various bit error rates A block is deemed to be significantly corrupted if its PSNR with respect to the encoded version is less than 40 dB. This latter measure is also dubious as small errors may be counted as similar to large errors. However, a combination of the above measures shows that the EREC allows an increase of between 3 and 10 times in BER.

Fig. 15(a) and (b) show examples of frame 40 at a BER of 3×10^{-4}, and Fig. 16(a) and (b) show the same frame at BER $= 3 \times 10^{-3}$. Note the reduced distortion in the facial representation of Fig. 15(b) compared with that of Fig. 15(a).

In order to verify the theoretical results of Fig. 5, we have further simulated the immediate effects of various numbers of errors on both schemes. These results ignore temporal propagation of errors (from one frame to the next) by only measuring the immediate fraction of the corrupted picture. Fig. 14 shows how the fraction of current frame corrupted varies with the number of errors introduced. Comparison of Figs. 5 and 14 show that they have similar shape and that

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.

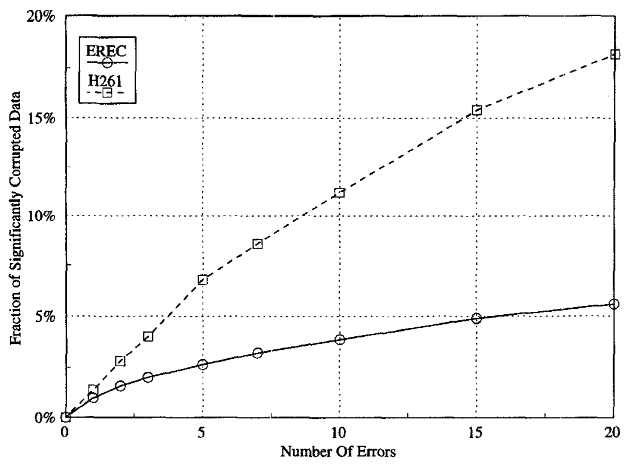


Fig. 14. Variation of fraction of current frame corrupted against number of errors per frame.

the EREC performance degrades only slowly with increasing numbers of errors, as demonstrated in Fig. 16(b).

VII. Conclusions

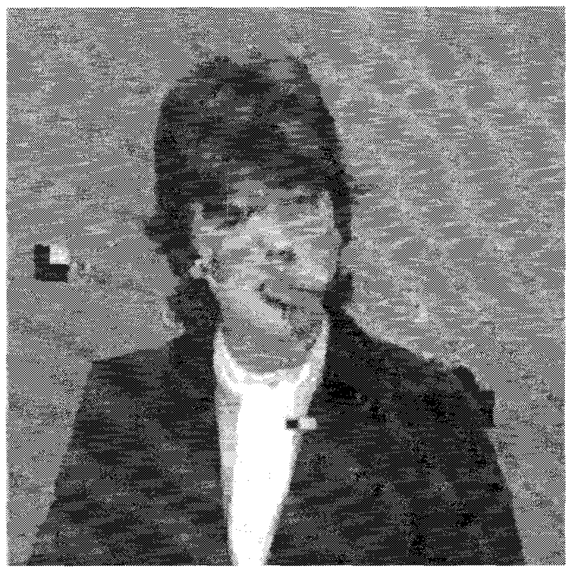
In this paper, we have introduced the error-resilient entropy code (EREC) as a method for providing error resilience for many compression coding methods that use variable-length blocks of data.

We have shown that the EREC can significantly reduce the channel error propagation effects and that the remaining channel error propagation is most likely to affect data from the end of longer blocks. For typical transform-based image and video coding schemes, the propagation affects high-frequency information from more active blocks. These errors are subjectively less visible [16] than errors in inactive regions or errors in low frequencies and motion vectors.

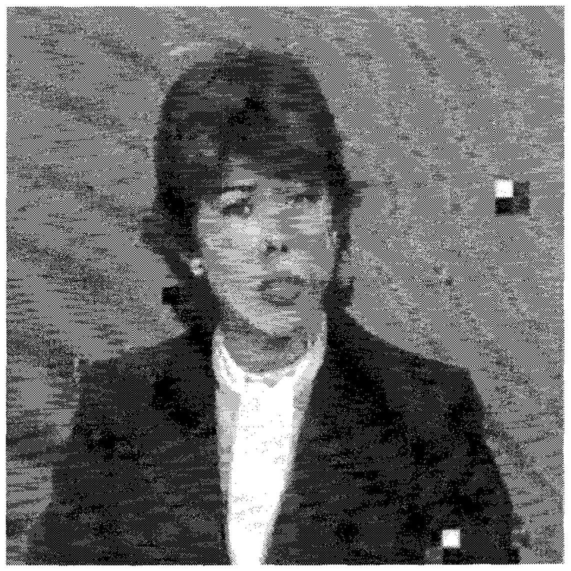
It is also found that with multiple errors, the increase in propagation effects is less significant than with other schemes. This is due to the propagation of subsequent channel errors affecting largely similar data to that of the first error. We have shown that the redundancy involved in using the EREC is less than that of using sync words, unless they are very widely spaced, giving large error propagation.

We have demonstrated the effectiveness of the EREC by simulations of both image and video coding using schemes based on transform and Huffman coding. The EREC has been found to give a significant improvement in quality in the presence of channel errors with virtually no added redundant information. It is found that the majority of the channel error propagation effects are in high-frequency coefficients in high activity regions of the image. These are subjectively much less noticeable than the effects of errors in low-frequency information and intraframe regions.

It can be seen in Fig. 15(b) that the most obvious errors arise from the immediate effects of errors in MBA and MTYPE information. Consequently, in a well-designed error-resilient scheme, it may be useful to give added protection to these bits



(a)



(b)

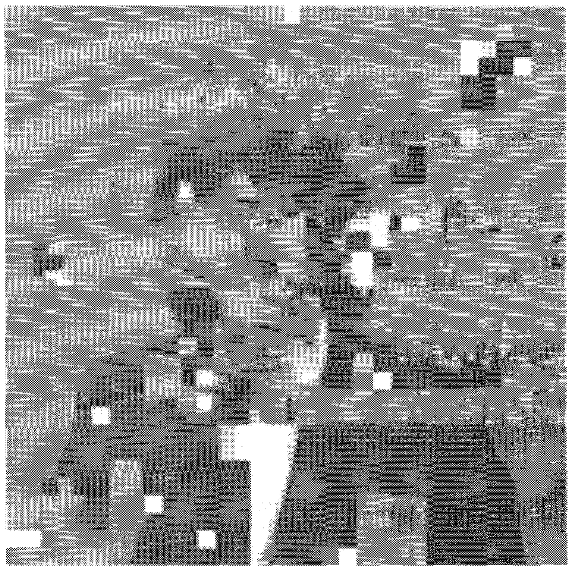
Fig. 15. Frame 40 of "Claire" with BER= 3×10^{4}(a) Standard H.261. (b) EREC.

by using error correction coding on the bits at the beginning of EREC slots.

It is noted that an EREC encoder and decoder could be designed to add on to a standard H.261 codec system to improve the error resilience. The complexity of the EREC codec would be quite high as the encoder and decoder both need to effectively decode and recode the data as far as distinguishing data for each block. It is also noted that for optimal performance in a noisy channel, some control over the fraction of intracoded MB's is desired. Hence, if possible, the EREC should be integrated into the video codec system.

The EREC should be considered to be one of a selection of techniques for error-resilient coding as it can be used in

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.


(a)

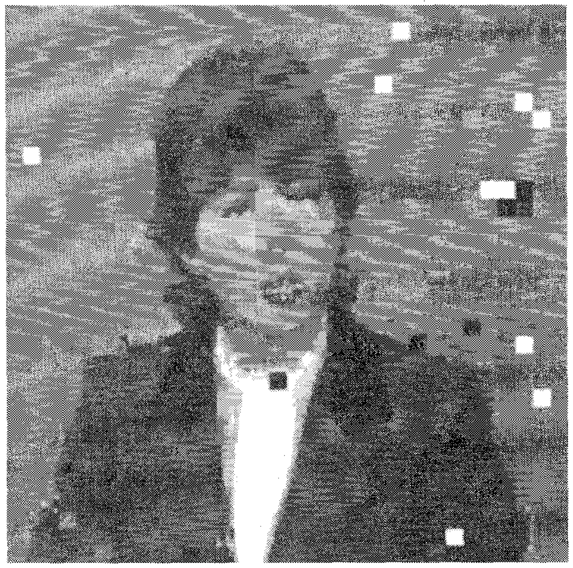

(b)

Fig. 16. Frame 40 of "Claire" with BER= 3×10^3. (a) Standard H.261. (b) EREC.

conjunction with error correction coding, error concealment, synchronization code words, and multilevel coding schemes. In particular, the use of synchronization codes may be essential for systems that suffer from bit insertion or deletion errors as the EREC provides no protection against these.

REFERENCES

[1] International Organization for Standardization (ISO) "JPEG digital compression and coding of continuous-tone still images," volume Draft ISO 10918, 1991.

[2] ———, "MPEG video committee draft," vol. ISO-IEC/JTC1/SC2/WG8 /MPEG, Dec. 18, 1990.

[3] CCITT Study Group XV, "Draft revision of recomendation H.261. Video codec for audiovisual services at p x 64 kbit/s," 1990.

[4] J. C. Maxted and J. P. Robinson, "Error recovery for variable-length codes," *IEEE Trans. Inform. Theory,* vol. IT-31, pp. 794-801, 1985.

[5] T. J. Ferguson and J. H. Rabinowitz, "Self-synchronizing Huffman codes.," *IEEE Trans. Inform. Theory*, vol. 30, pp. 687–693, 1984.

[6] B. L. Montgomery and J. Abrahams, "Synchronizing of binary source codes.," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 849-854, 1986.

[7] W. M. Lam and A. R. Reibman, "Self-synchronization variable-length codes for image transmission," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1992, pp. III-477–III-480.

[8] N. MacDonald, "Transmission of compressed video over radio links," *SPIE Visual Commun. Image Processing*, pp. 1484-1488, 1992.

[9] N. T. Cheng and N. G. Kingsbury, "The ERPC: An efficient error-resilient technique for encoding positional information of sparse data," *IEEE Trans. Commun.*, vol. COM-40, pp. 140–148, 1992.

[10] N. T. Cheng, "error-resilient video coding for noisy channels," PhD thesis, Cambridge Univ. Eng. Dept., 1991.

[11] J. W. Modestino and D. G. Daut, "Combined source-channel coding of images," *IEEE Trans. Commun.*, vol. 27, pp. 1644–1659, 1979.

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

[13] J. Rosebrock and P. W. Besslich, "Shape-gain vector quantization for noisy channels with applications to image coding.," *IEEE Trans. Selected Areas Commun.*, vol. 10, pp. 918–925, 1992.

[14] D. W. Redmill and N. G. Kingsbury, "Improving the error resilience of entropy coded video," in *Image Processing: Theory and Applications, Proc. IPTA 93*, San Remo, Italy, June 14–16, 1993.

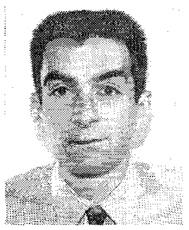
[15] G. G. Langdon and Jun, "an introduction to arithmetic coding," *IBM J. Res. Develop.*, vol. 28, pp. 135–149, 1984.

[16] A. N. Netravali., "On quantizers for DPCM coding of picture signals.," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 360–370, 1977.

[17] D. W. Redmill, "Image and video coding for noisy channels," Ph.D. thesis, Cambridge Univ. Eng. Dept., 1994.

[18] D. W. Redmill and N. G. Kingsbury, "Still image coding for noisy channels," in *IEEE Int. Conf. Image Processing*, vol. 1, Austin, TX, 1994, pp. 95–99.

[19] D. W. Redmill, "Robust architectures for image and video coding," in *Proc. 2nd Int. Workshop Mobile Multimedia Commun.*, Bristol, England, Apr. 11–14, 1995.



David W. Redmill received the B.A. honours degree in electrical and informational sciences in 1991 and the Ph.D. degree in 1995, both from the Department of Engineering, University of Cambridge, Cambridge, UK.

Since then, he has been a research assistant with the Centre for Communications Research, University of Bristol, Bristol, UK. His research intersts include low bit-rate image and video coding for noisy channels.

Nick G. Kingsbury (M'87) received the honours degree in 1970 and the Ph.D. degree in 1974, both in electrical engineering from the University of Cambridge, Cambridge, UK.

From 1973 to 1983, he was a design engineer and, subsequently, a group leader with Marconi Space and Defence Systems, Portsmouth, UK, specializing in digital signal processing and coding as applied to speech coders, spread spectrum satcomms, and advanced radio systems. Since 1983, he has been a Lecturer in Communications Systems at the University of Cambridge. His current research interests include image compression, eror-robust source coding techniques, and combined coding and modulation.

Authorized licensed use limited to: UNIVERSITY OF BRISTOL. Downloaded on September 30, 2009 at 10:50 from IEEE Xplore. Restrictions apply.