Object-Based Unequal Error Protection

Madhavi Marka and James E. Fowler

Engineering Research Center

Mississippi State University

A Report Prepared for the

High Performance Visualization Center Initiative (HPVCI)

February 25, 2002

Technical Report MSSU-COE-ERC-02-01, Engineering Research Center, Mississippi State University, February 2002

Contents

Abstract

This report investigates the application of forward-error-correcting codes to data organized as multiple, independent multimedia objects and encoded with modern embedded coders. Capitalizing on the strict importance-ordering characteristic of embedded encodings, the strength of the error protection is optimized such that data that is more important to the reconstructed quality of the dataset is assigned stronger protection. The focus of the investigation is on providing this optimization while maintaining the ability to independently access the individual multimedia objects. Experimental results are presented for still-image objects that illustrate that the desired independent-access ability comes at a cost in reconstruction quality. In addition, it is observed that this cost increases as the channel-loss conditions actually experienced degrade from those for which the optimal-protection arrangement was designed.

1. Introduction

Multimedia information is inherently data intensive and usually requires large amounts of memory for storage and a large bandwidth for transmission. As a consequence, compression of multimedia data has become increasingly important to many applications. Since multimedia data can usually tolerate some degradation in quality, lossy compression, in which the reconstructed data is close, but not the same as the original pre-compressed data, is the preferred paradigm. Lossy data compression has a long history of success in the coding of still images and video and has thus been recently applied to multimedia data. In a multimedia scene, different regions of the dataset usually vary in importance. In object-based coding, a multimedia scene is viewed as a composition of multiple, distinct objects which are independently compressed. As a result, object-based coders provide independent access to the individual multimedia objects, allowing a user to access and manipulate an object independently of other objects. Objects in a multimedia dataset are encoded independently along with information on size and position within the dataset, and, at the decoder, provisions are provided for users to both decode an object of interest and also to interactively manipulate object composition without the need to decode the entire scene [1]. MPEG-4 [2], a recent video-compression and multimedia-coding standard, relies heavily on object-based representation of scenes. MPEG-4 objects include arbitrarily shaped video objects and arbitrarily shaped still-image texture objects, among others. Recent efforts at remote visualization, namely [3], also employ the object-coding paradigm whose progressive-transmission display permits users to "browse" through extremely large datasets and whose random-access capabilities allow selection of certain "regions of interest" for further refined visualization.

Object-based compression can alleviate many problems associated with the storage and transmission of multimedia datasets. However, when communication of multimedia data takes place over networks, the multimedia data is divided into packets which are transmitted individually, and some of these packets are often lost in transit across the network. The most prevalent cause of this packet loss is that packets are randomly discarded due to network failure and congestion. Conventional methods for handling this data loss require identification and retransmission of the lost packets which can cause significant network delays, waste of network bandwidth, and exacerbation of the situation that caused the loss in the first place. Methods that avoid retransmission include error-protection algorithms which assign forward-error-correcting (FEC) codes to the data. This FEC information is, in essence, a controlled form of redundancy that allows lost data to be recovered from correctly received data in many instances. The amount of lost data that can be recovered in this manner depends on the strength of the FEC codes, i.e., how much redundancy is added. The most straightforward approach to FEC protection is that of Equal Loss Protection (ELP) in which the strength of error protection is applied equally to all portions of the data to be transmitted. Alternatively, Unequal Error Protection (UEP) algorithms assign unequal amounts of FEC protection to the data in an effort to vary protection strength according to the importance of various portions of the data. When UEP FEC codes are applied to data that has been subjected to lossy compression, the measure of "importance" in determining the application of the UEP redundancy is usually the quality of the reconstruction. In general terms, the UEP approach tends to perform better than ELP in terms of quality [4], since one can better optimize the UEP placement so as to maximize the quality of the reconstruction of the data. This project investigates the FEC protection of object-based codings. Specifically, we explore the assignment of UEP to objects of a multimedia dataset with the goals that each object 1) is protected within itself according to the importance of each of its bits to the reconstruction quality of the object, 2) has an object-level amount of error protection proportional to the object's importance to the reconstruction quality of the scene, and 3) can be accessed independently of other objects of the image. Hereafter, only arbitrarily shaped still-image objects, which are a special case of multimedia data, are considered. The general approaches considered, though, apply to other multimedia forms.

In the remainder of this manuscript, we first review concepts of embedded image coding and the protection of

embedded bitstreams against packet losses using UEP algorithms. We then introduce two general approaches for applying UEP FEC codes to protect objects and consider the issue of independent access to the protected objects. Experimental results exploring the performance of these two methods are presented, and then some concluding remarks are made.

2. Background

SPIHT, an embedded image coding technique, and UEP, a loss-protection algorithm, are the building blocks for object-based UEP. This chapter provides an overview of embedded-coding techniques and also explains the different ways by which protection of data over lossy channels can be achieved.

2.1 Embedded Coding

In an embedded code, the bits of a bitstream are arranged in the order of importance. This is similar to binary finite-precision representations of real numbers—for each extra bit added to the right in a representation of a real number, the precision increases. For example, if an embedded encoder produces two files with size N and M bits in an embedded fashion, where $M > N$, then the file with N bits is exactly same as the first N bits of the file with size M. Using an embedded code, the encoding process can stop at any point once the desired target bit count is met. Similarly at the decoder, decoding can be stopped at any point and reconstructions for lower-rate encodings can be produced [5]. Progressive transmission of the data is thus possible in that the decoder can produce multiple reconstructions of increasing fidelity as it receives more and more of the embedded bitstream.

2.2 Set Partitioning in Hierarchical Trees (SPIHT)

SPIHT is a fully embedded image-compression technique with precise rate control and low complexity [6]. SPIHT outperforms other image-compression techniques such as JPEG [7], vector quantization, and stack run [8]. SPIHT is a refinement of the Embedded Zerotree Wavelet algorithm [5]. It orders data progressively; i.e., globally important data is sent first. The decoder can stop decoding at any point in the decoding process and a lower-quality image can be decompressed and reconstructed.

SPIHT employs a discrete wavelet transform to transform the image, and then encodes zerotrees found in the wavelet-coefficient subbands. The organization of the wavelet coefficients resulting from the transform is in the form of a hierarchical subband system. The lower-frequency components, where most of the image energy is concentrated, are in higher levels and the frequency increases when moving from higher levels to lower levels. The highest-level component, or baseband, is treated as the tree root. In this tree structure, each node consists of either four direct descendants (offspring) or no offspring (leaves). A significance test is performed on wavelet coefficients with respect to a given threshold; i.e., a coefficient is considered to be significant if it is greater than the threshold. If a coefficient is insignificant, then all the coefficients at a finer scale of the same spatial orientation are assumed to be insignificant.

SPIHT encoding forms sets of the wavelet coefficients according to spatial orientation and performs magnitude tests to order them. The magnitude-ordered coefficients are coded using a set-partitioning algorithm and are transmitted in bitplanes with the most-significant bitplane first. There are two major passes in the SPIHT algorithm. In the first pass, called the sorting pass, coefficients which are significant with respect to the current threshold are identified and their pixel coordinates are sent. In the second pass, called the refinement pass, the precision of the previously sent coefficients is increased by sending the next most-significant bit from their binary representations. The output is a fully embedded code which allows the encoder to meet an exact target bit rate.

2.3 Shape-Adaptive SPIHT Encoding

In object-based image coding, an image is partitioned into various objects of arbitrary shapes, and shape-adaptive encoding is used for compression of these arbitrarily shaped objects. The SPIHT algorithm can be rendered shape-adaptive by incorporating a shape-adaptive wavelet transform (SA-DWT) such as that of [9]. In a SA-DWT, the number of coefficients is exactly equal to the number of pixels in the objects and is achieved by using a mask that is opaque for only object pixels and transparent everywhere else. In shape-adaptive SPIHT encoding, each time a coefficient is to be encoded, its position with respect to the mask is taken into consideration. If a coefficient is within the opaque region of the mask, it is encoded as normal. All transparent coefficients are considered to be insignificant at all times and thus encoding is avoided. Similarly, shape information is also used to guide the decoding process.

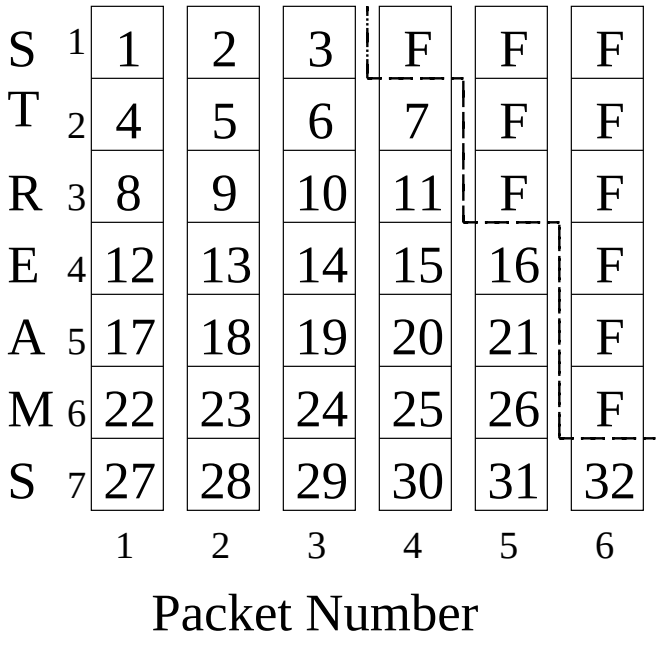


Figure 1: Packet arrangement for UEP—F represents an FEC code byte, while data bytes are numbered according to their occurrence in the embedded bitstream; packets are columns and streams are rows (adapted from [4]).

2.4 Reed-Solomon Codes

Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications. In these erasure codes, k blocks of source data are encoded to produce n blocks to be transmitted such that any subset of k encoded blocks is sufficient to reconstruct the k blocks of the source data [10]. That is, a (n, k) Reed-Solomon code operates on k source blocks, producing n total blocks for transmission such that $(n - k)$ is the number of redundancy blocks added. Such a code allows the receiver to lose up to $(n - k)$ blocks in the total of n blocks and still recover all k blocks of the source. Often, the block size for the code is a single byte.

2.5 Unequal Error Protection (UEP)

UEP is a framework that assigns FEC codes to bitstreams such that the most important information receives the greatest protection. UEP attempts to provide graceful degradation of image quality as the packet losses increase [4]. There are several techniques that have been proposed for UEP [4, 11, 12]. In this report, we focus on the algorithm due to Mohr *et al*. [4] which is designed to protect embedded bitstreams with UEP FEC assignments so as to optimize the reconstructed-image quality for a given probabilistic packet-loss model. In the algorithm of [4], FECs and data bytes form a "stream," with the number of streams equal to the number of bytes in each packet to be transmitted. In other words, a stream contains one byte from every packet transmitted. This arrangement is illustrated in Fig. 1.

Fig. 1 shows a bitstream arranged into 7 streams and 6 packets. The number of streams is equal to the packet length (7 bytes in this example). The tenet central to the algorithm of [4] is that all the bytes of a stream can be decoded if the number of packets lost is less than or equal to the number of FEC bytes in that stream; such is the case when Reed-Solomon codes are applied to each stream to generate its FEC bytes. Since the bitstream is embedded in that earlier parts of the bitstream are more important than latter parts, a greater number of FECs must be assigned to earlier streams than to latter streams [4, 13]. The algorithm of [4] attempts to find an optimal arrangement of these FECs in order to maximize the expected reconstructed-image quality.

The following example further explains the process of FEC arrangement and data recovery. Fig. 1 shows the arrangement for sending 32 bytes of data with 10 FEC bytes over a lossy channel. Fig. 2 shows the situation at the decoder in the case that packet 4 is lost while the other five packets are received correctly. Given that a stream can be retrieved intact if the number of packets lost is less than or equal to the number of FECs in that stream, the initial 26

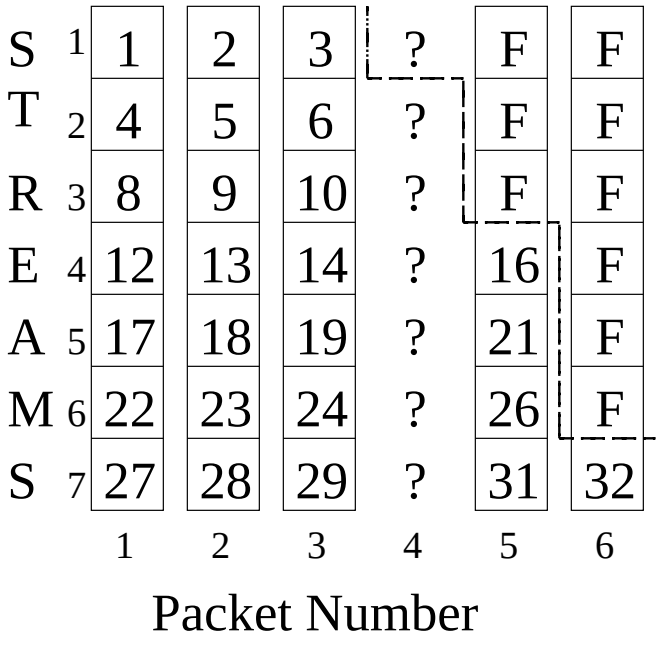


Figure 2: Packets received with packet 4 lost (adapted from [4]).

bytes of data can be recovered after inverting the FEC code. Because the packet 4 is lost in this case, data bytes 27, 28, 29, 31, and 32 are received correctly while data byte 30 is lost. In an embedded bitstream, a byte cannot be decoded unless the previous byte is decoded. In this case, bytes 31 and 32 are not useful because byte 30 is lost and thus a total of 29 bytes are used to decode and reconstruct the image. Fig. 3 shows data recovery for this case. Similarly, if packet 5 instead of packet 4 was the lost packet, 30 data bytes would be decodable. On the other hand, if two packets are lost out of the six packets, the first 11 bytes of data can be retrieved and recovery of bytes 12-15 depends on which packets are lost. In an embedded bitstream, each extra byte recovered improves the quality of the image. Hence this arrangement leads to graceful degradation of image quality with increasing packet losses [4].

In the UEP-assignment algorithm due to Mohr *et al.* [4], if f_i is the number of FEC bytes assigned to stream i, the FECs assigned to all streams can be represented as a vector,

$$\bar{f} = (f_1, f_2....f_N). \tag{1}$$

For a given $\bar{f}$, $M_i(\bar{f})$ is the sequence of data bytes in stream i. The data-byte sequence in all streams is given as a vector,

$$M(L, \bar{f}) = M_1(\bar{f})M_2(\bar{f})...M_L(\bar{f}). \tag{2}$$

The incremental PSNR of decoding stream i is the difference in PSNR between decoding stream i and stream $(i-1)$,

$$g_i(\bar{f}) = \mathrm{PSNR}[M_i(\bar{f})] - \mathrm{PSNR}[M_{i-1}(\bar{f})]. \tag{3}$$

Since the data is embedded, $f_i > f_{i-1}$, and, if data byte $(i+1)$ can be decoded, then byte i can also be decoded.

The number of FECs required for a message fragment depends on the packet-loss model. The packet loss model is defined in terms of a probability mass function (pmf), p_n, the probability of losing n packets, where $n = 0, 1, \ldots, N$. The probability of losing k packets is then

$$c(k) = \sum_{n=0}^{k} p_n. \tag{4}$$

The expected PSNR for a received message as a function of $\bar{f}$ is

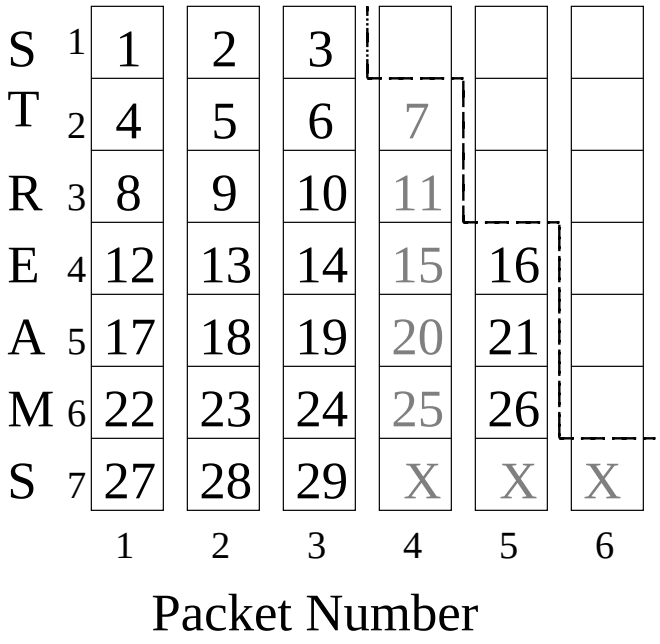


Figure 3: Data recovery—gray indicates those bytes recovered by inverting the FEC (adapted from [4]).

$$G(\bar{f}) = \sum_{i=1}^{L} c(f_i) g_i(\bar{f}). \tag{5}$$

In the algorithm of [4], $G(\bar{f})$ is maximized to get $\bar{f}$ for a packet-loss model given by the pmf; the details of this maximization can be found in [4]. Note that a profile of PSNR-vs-prefix length, $\mathrm{PSNR}[n]$, which gives the PSNR when a prefix of n data bytes is decoded, is needed in (3). Such a PSNR profile is easily obtainable during encoding in embedded algorithms such as SPIHT.

3. Object-Based Unequal Error Protection

The goal of the work described here is to protect object-based codings with UEP in order to transmit object-based data with optimal quality over lossy networks. In doing so, we would like to protect each byte of an object according to its importance as well as provide independent access to all objects.

We will consider still-image data as a special case of multimedia data, and we will assume that an image has already been partitioned into a set of objects. Such object partitioning can be done in practice manually or by an automated feature-detection algorithm. In either case, each object of the image is encoded individually, and, prior to transmission over a network, FECs are assigned to the object bitstreams to provide graceful degradation of image quality with increase in packet losses. Below we explore two approaches for providing such error protection for each object. We will assume that individual objects of the image are encoded in an embedded manner using the shape-adaptive SPIHT algorithm although any embedded image coder would suffice.

In an embedded coding, each object is coded as a sequence of enhancement layers. In SPIHT and other zerotree algorithms, these enhancement layers are bitplanes from the wavelet coefficients. We assume that the default representation of the global scene is organized such that object layers are arranged in an interleaved fashion. For example, assume that we have two objects in the image with M_1 layers from object 1 and M_2 layers from object 2. The individual layers of the objects are interleaved to form a single compressed bitstream B,

$$B = P_{11}, P_{21}, P_{12}, P_{22}, P_{13}, P_{23}, ..., P_{1M_1}, P_{2N}, P_{2N+1}, ...P_{2M_2}, \tag{6}$$

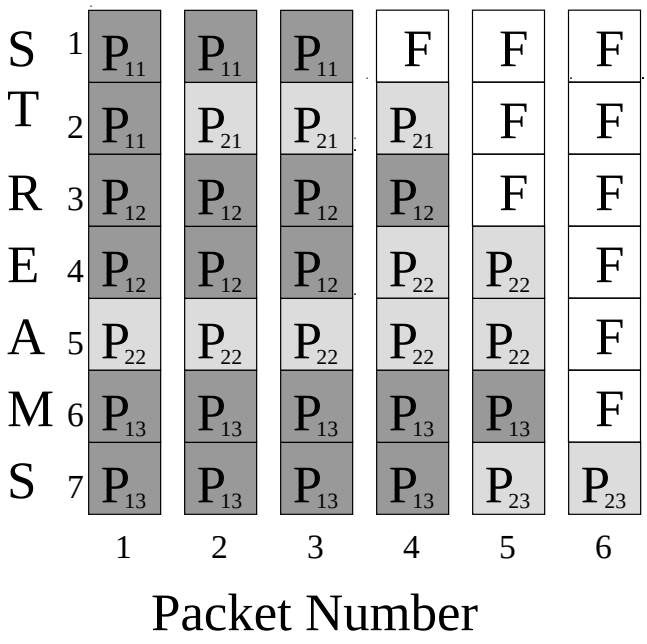


Figure 4: Packet arrangement for the CUEP approach—F represents an FEC byte, and P_{ij} represents a data byte from layer j of object i. Dark-shaded blocks indicate bytes from object 1 while light-shaded blocks indicate bytes from object 2.

where P_{ij} is layer j from object i. In this arrangement, both objects are refined equally fast, and bitstream B forms a globally embedded representation of the entire scene. The bitstream, $\tilde{B}$, that will be decodable after losing some packets in transmission will be some truncated version of B. The length of $\tilde{B}$ will depend on the nature of the FEC protection that is applied. In all cases, we measure scene quality with a whole-image PSNR. Specifically, suppose, for example, the decodable bitstream is $\tilde{B} = P_{11}, P_{12}, \tilde{P_{12}}$. In this case, we need to reconstruct the image by doing the following:

- reconstruct object 1 using the first layer, P_{11}, and the truncated version of the second layer $\tilde{P_{12}}$,
- reconstruct object 2 using the first object-2 layer, P_{21}, and
- composite the reconstructed object 1 and object 2 to form a reconstructed image.

The whole-image PSNR would be calculated as the PSNR between the composited reconstructed image and the original image.

3.1 Combined Unequal Error Protection (CUEP)

The most straightforward approach to providing UEP for the bitstream of (6) would be to apply the UEP algorithm of [4] to it directly. Given PSNR-vs-prefix profiles for the individual objects, a global PSNR profile for the interleaved bitstream as needed for the algorithm of [4] can be obtained as described in Appendix A.

Fig. 4 illustrates a packet arrangement typical to this approach. The layers of object 1 and 2 are interleaved, and the error-correcting codes are assigned to the individual object bytes according to their contribution to the entire-image PSNR, as determined from the global PSNR profile. Unfortunately, this arrangement does not provide independent access to the objects since the individual-object bytes are not in continuous packet locations. That is, any given packet might hold both object-1 and object-2 bytes. Hence, independent access to individual objects is possible only after the decoder inverts the entire UEP code matrix, which can occur only after all packets are sent by the encoder and the decoder determines which packets are missing.

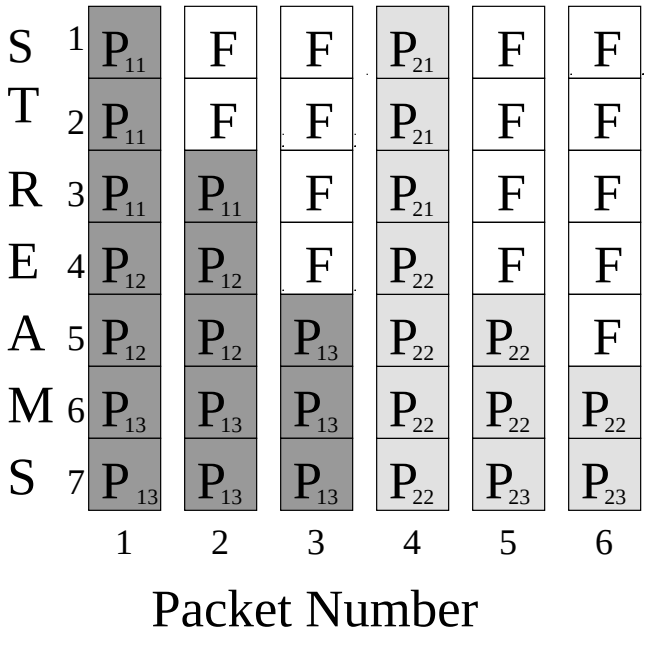


Figure 5: Packet arrangement for the IUEP approach—F represents an FEC byte, and P_{ij} represents a data byte from layer j of object i. Dark shaded blocks indicate bytes from object 1 while light shaded blocks indicate bytes from object 2.

The performance of this approach is measured by an expected PSNR. Let $p(x_i, N)$ be the probability that x_i packets are lost from a total of N packets, and $D(x_i)$ be the whole-image PSNR obtained in this case. Note that $D(x_i)$ depends on the FEC arrangement of the bitstream B as determined by the UEP algorithm of [4]. The expected PSNR is calculated as

$$\bar{D} = \sum_{i=0}^{N} p(x_i, N) D(x_i). \tag{7}$$

In subsequent experimental results, we use an exponential probability loss model for probability density $p(x_i, N)$.

3.2 Individual Unequal Error Protection (IUEP)

In order to ensure independent access to the individual objects in the bitstream of (6), an alternative to the previous UEP strategy would be to "deinterleave" the bitstream and then apply the algorithm of [4] to each object individually. That is, from the bitstream of (6), we would generate two bitstreams B_1 and B_2,

$$B_1 = P_{11}, P_{12}, P_{13}, .., P_{1M_1} \tag{8}$$

$$B_2 = P_{21}, P_{22}, P_{23}, .., P_{2M_2}. \tag{9}$$

In this case, we have two PSNR-versus-prefix profiles, one for each object. The algorithm of [4] is applied individually to bitstreams B_1 and B_2 to get the FEC configuration for each. Fig. 5 shows the packet arrangement for this approach. The first 3 packets are from object 1, and next 3 packets are from object 2. The data bytes of objects 1 and 2 are in consecutive packets, while a given packet contains bytes from only one object. Hence, this approach provides independent access to objects. For example, the decoder can invert the UEP code matrix for object 1 after only the first 3 packets are transmitted and accounted for—it does not have to receive packets 4 through 6 to do so. The advantage of this method over CUEP is that it provides independent access without the need to wait until all packets have been transmitted.

The performance of this approach is measured by an expected PSNR. Let $p(x_i, N_i)$ be the probability that x_i packets are lost from N_i packets in object i. Then $p(x_1, N_1)p(x_2, N_2)$ is the probability that x_1 packets are lost from

(a) (b)

Figure 6: Original images, (a) lenna, (b) coastguard.

object 1 and x_2 packets are lost from object 2. The expected whole-image PSNR is calculated as

$$\bar{D} = \sum_{x_1=0}^{N_1} \sum_{x_2=0}^{N_2} p(x_1, N_1) p(x_2, N_2) D(x_1, x_2) \tag{10}$$

where $D(x_1, x_2)$ is calculated from individual-object PSNR profiles as described in Appendix A. In subsequent experimental results, we assume an exponential probability loss model for probability densities $p(x_i, N_1)$ and $p(x_i, N_2)$.

4. Experimental Results

This chapter presents experimental results for the methods discussed above. All the experiments are conducted using the shape-adaptive SPIHT implementation in QccPack [14]. The 512×512 grayscale lenna and 352×288 coastguard images are used for results. The lenna image is manually partitioned into two objects—object 1 is lenna and object 2 is background. The coastguard image is also manually partitioned into two objects—object 1 is ship and object 2 is background. The original lenna image is shown in Fig. 6(a) and the two objects, lenna and the background, are shown in Figs. 7(a) and (b) respectively. The original coastguard image is shown in Fig. 6(b) and the two objects, ship and background, are shown in Fig. 7(c) and (d) respectively. In all the experiments, a bit rate of 0.5 bits per pixel (bpp) is assumed for the entire image. The tests are done assuming an exponential packet-loss model at a mean loss rate of 20%. We assume 47 bytes of data in each packet, as ATM packets have a payload length of 48 bytes of which one byte is required for a packet-sequence number.

4.1 CUEP Results

The packet distribution for CUEP applied to lenna is as follows. A total of 349 packets are sent of with a total payload length of 16403 bytes. The algorithm of [4] applied to the interleaved bitstream of (6) yields 9935 data bytes with the remaining 6468 being FEC codes, while in the total 9935 data bytes, 7992 bytes are from object 1 and 1343 are from object 2. The PSNR of the entire image with Combined UEP is 34.1 dB when all packets are received.

The PSNR-vs.-prefix profile for the interleaved bitstream is shown in Fig. 8(a). The FEC arrangement obtained with the algorithm of [4] for this bitstream is shown in Fig. 8(b). Clearly, unequal amounts of error codes are assigned to the data bytes, with the amount of FEC codes assigned to the data decreasing as the stream number increases; i.e., each byte is protected according to its importance.

4.2 IUEP Results

The packet distribution for IUEP applied to lenna is as follows. When the algorithm of [4] is applied to the deinterleaved bitstreams of (8) and (9), a total of 350 packets are sent of which 180 packets are from object 1 and 170

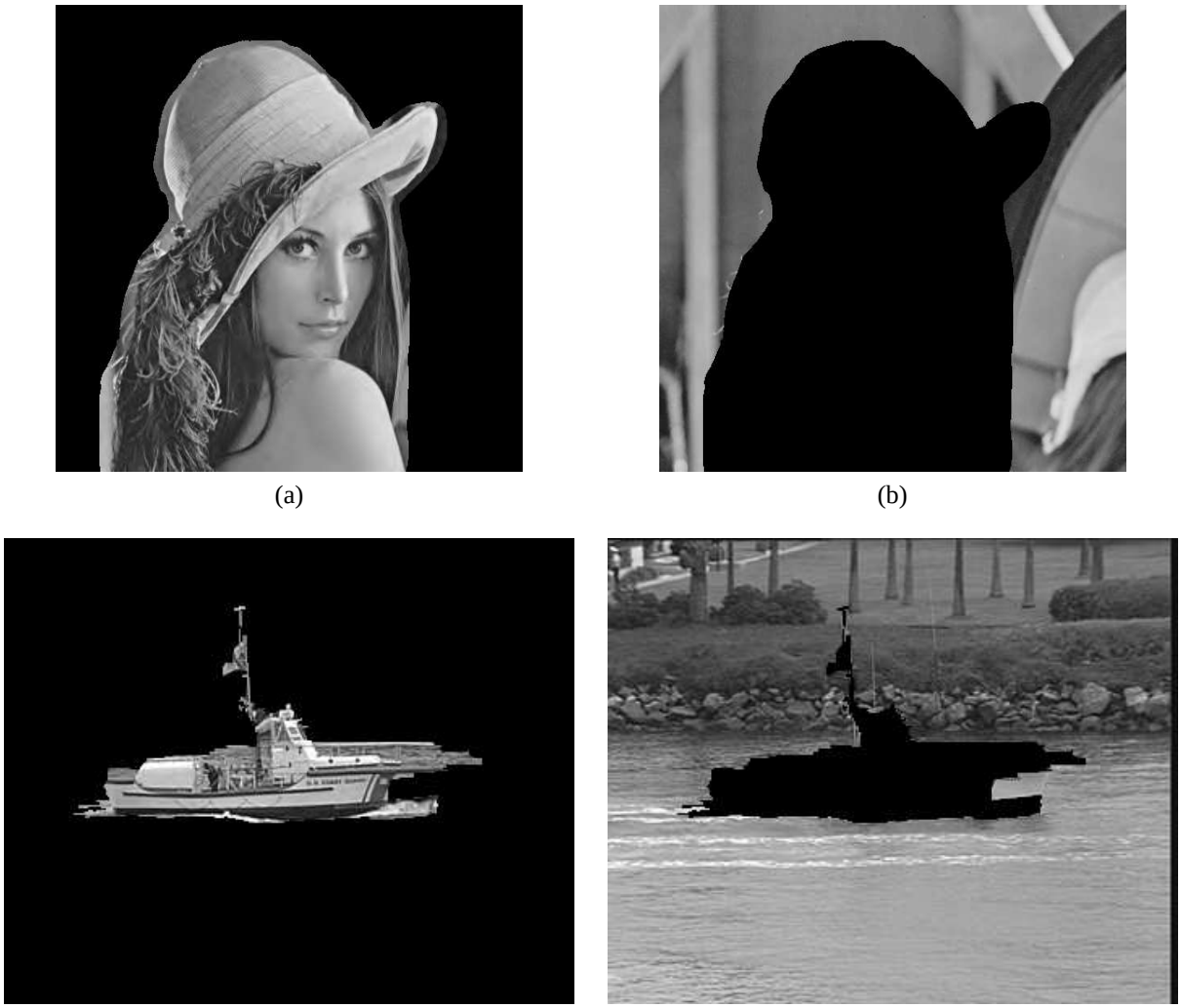
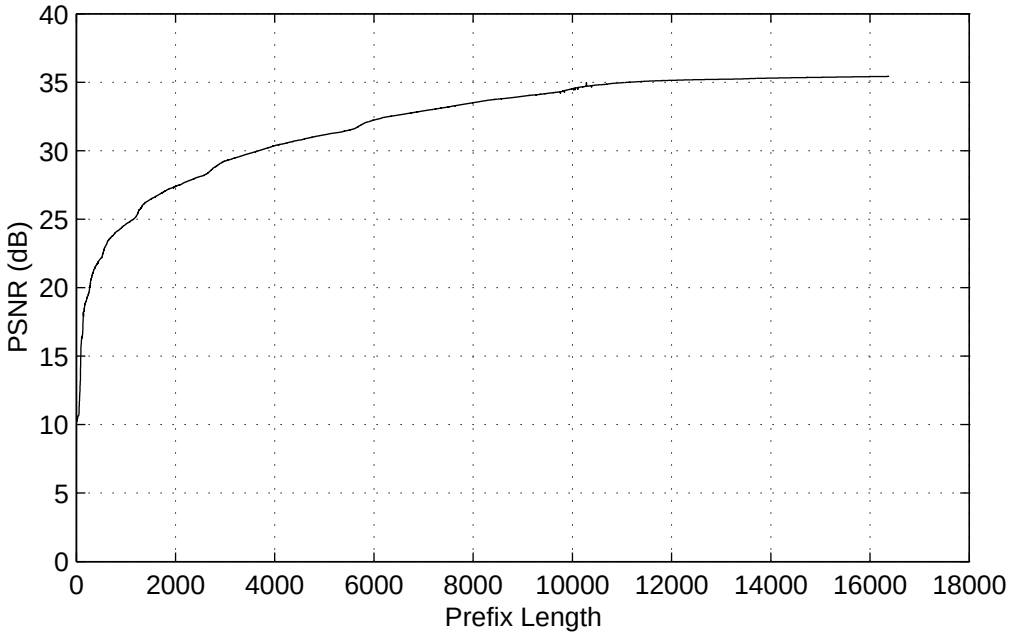
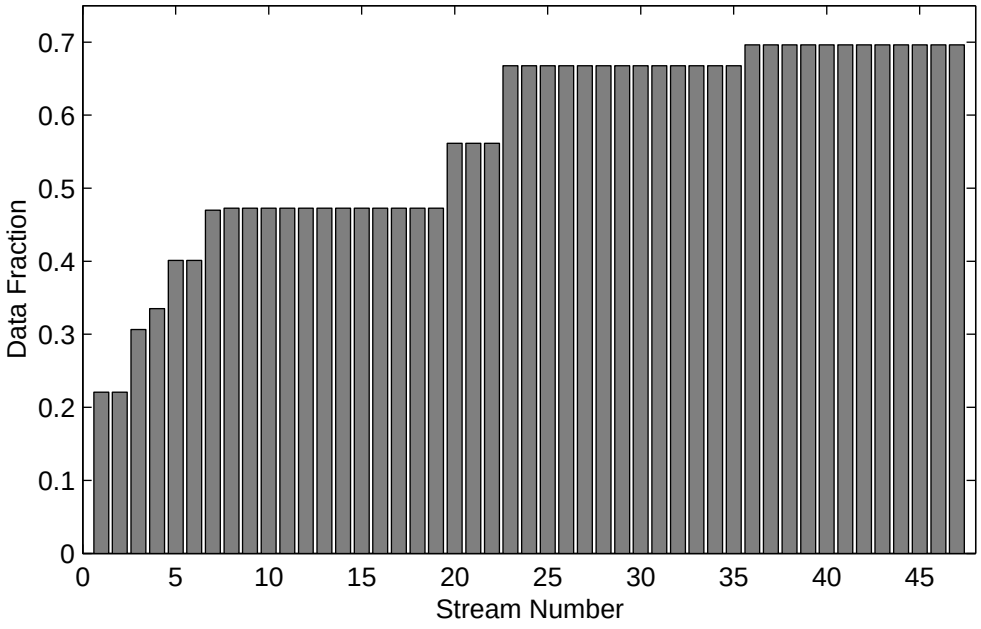


Figure 7: (a) Object 1, lenna, (b) object 2, background, (c) object 1, ship, (d), object 2, background.



(a)



(b)

Figure 8: CUEP for lenna, (a) PSNR-vs.-prefix profile, (b) the fraction of each stream devoted to data as opposed to FEC as obtained using the algorithm of [4] for the interleaved bitstream.

packets are from object 2. The total payload length in this case is 16450 bytes of which 8342 bytes are data bytes. In the total of 8342 data bytes, 4784 bytes are from object 1 and 3540 bytes are from object 2. The PSNR of the entire image is 32.9 dB when no packets are lost.

PSNR-vs.-prefix profile is shown for each object in Fig. 9. In Fig. 10, the FEC arrangements for the object 1 and object 2 bitstreams are shown where the amount of FEC codes assigned to the data decreases as the stream number increases. This shows that the error protection is of unequal nature, and each data byte is protected according to its importance.

4.3 Comparison

The two approaches described above assign FEC codes to the compressed bitstream using different organization of the bitstream. IUEP supports independent access to individual objects whereas CUEP does not support independent access. We now compare the performance of these approaches to determine a cost of providing this independent access.

In the Fig. 11, the expected PSNR is shown for the CUEP and IUEP approaches under a variety of channel conditions for lenna; i.e., for each approach, we design the FEC arrangement assuming a channel with an exponential-loss model with mean loss rate of 20% and then determine performance using an exponential-loss channel with a loss rate of $\lambda, 0 \leq \lambda < 100\%$. In this manner, we evaluate the performance of the UEP code arrangement for the situation that the channel encountered is different from that for which we designed the code.

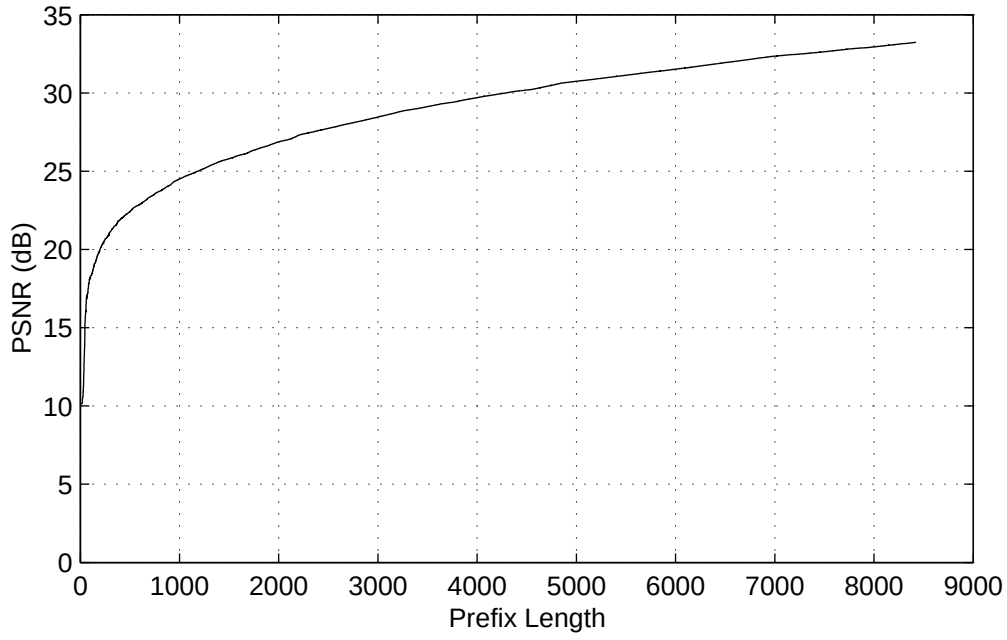
The expected PSNR performance is given by (7) for CUEP or (10) for IUEP. It is shown that the PSNR obtained using CUEP is around 1.25 dB higher than for IUEP for channels with loss at or below the designed-for loss rate (20%). For greater loss rates, the difference in PSNR between the two approaches gradually increases to about 7.5 dB at 100% packet loss. This difference in PSNR can be considered to be the cost of obtaining independent access to objects, and this cost increases as the mismatch between the actual and design channels increases. Actual reconstructed lenna images for both CUEP and IUEP at a variety packet losses are shown in Figs. 12 and 13. Until 40% packet losses, image quality is high in both methods. The image quality begins to degrade at about 50% packet loss.

The expected PSNR for the CUEP and IUEP approaches under a variety of channel conditions is shown in Fig. 14 for coastguard. It is shown that the PSNR obtained using CUEP is around 0.4 dB higher than the individual method for channels with loss at or below the designed-for loss rate (20%). For greater loss rates, the difference in the PSNR gradually increases up to 6 dB at 100% packet loss. As before, we conclude that the cost of providing independent access to the objects increases as the mismatch between the actual and design channels increases. Actual reconstructed images for both CUEP and IUEP at a variety of packet losses are shown in Figs. 15 and 16. Until 25% packet loss, image quality is high in both methods, degrading thereafter.

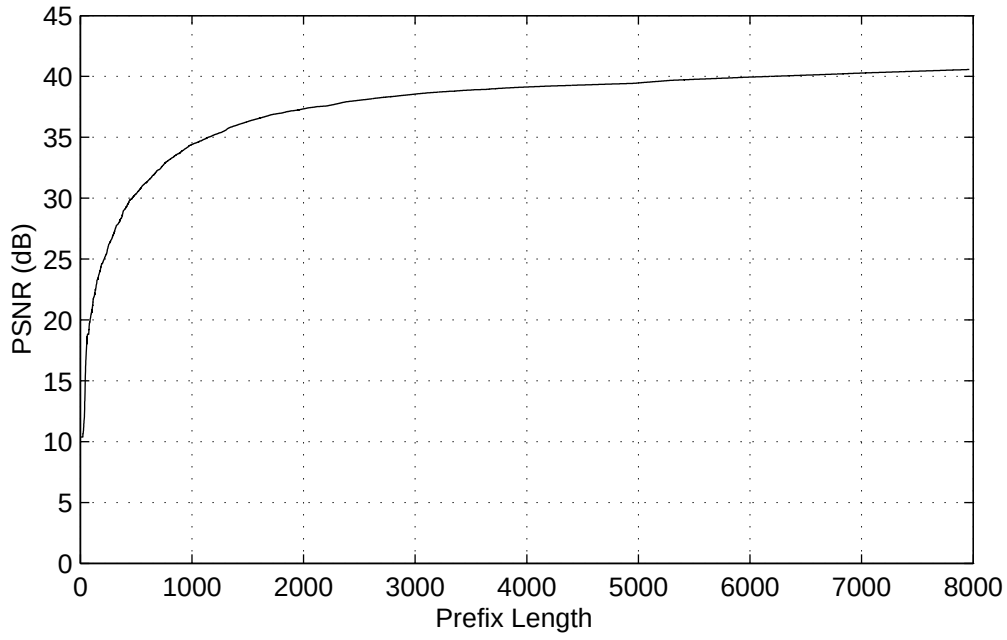
5. Conclusions and Future Work

Two general approaches to provide object-based UEP have been investigated in this report. These two methods assign FEC codes to the image data according to its importance using embedding encoding and a UEP algorithm. The CUEP approach outperforms the IUEP method in expected PSNR, but does not provide independent access to the objects. Hence, there is a cost associated with independent access; the experimental results reveal that this cost increases as the mismatch between the actual and design channels increases.

Potential future work in this regard would be to develop IUEP algorithm to improve its performance when the packet losses increase. It might be possible to determine an object-level of importance and assign more protection to objects according to their contribution to the entire image PSNR automatically. Further, this work is applicable to other multimedia types such as those arising in MPEG-4 data streams. Currently, still-image textures are coded in MPEG-4 in an embedded fashion using an algorithm similar in spirit to shape-adaptive SPIHT as used in this report; the work presented here will be applicable to other video/multimedia objects as embedded formats are adopted as extensions to MPEG-4 or in subsequent standards. On the other hand, embedded coding is fundamental to the object-based visualization of the system of [3]; indeed, it is anticipated that sophisticated embedded coding will be increasingly crucial to the success of remote-visualization systems deployed on extremely large-scale datasets. Consequently, optimal protection schemes such as those explored in this work will take on increasing importance in visualization and other multimedia-communication systems.

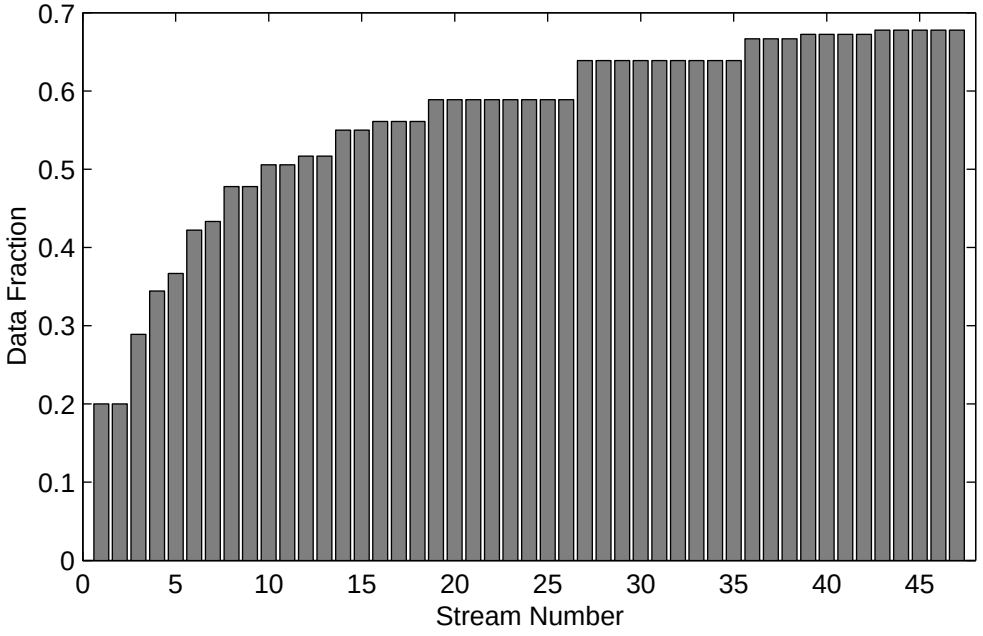


(a)

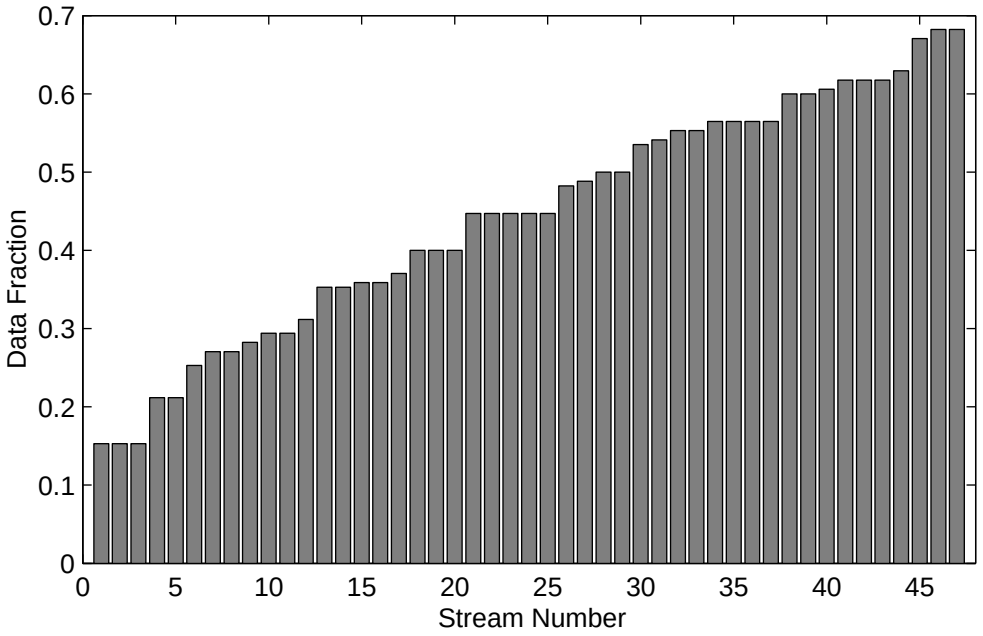


(b)

Figure 9: IUEP for lenna—PSNR-vs.-prefix profiles, (a) object 1, (b) object 2.



(a)



(b)

Figure 10: IUEP for lenna—fraction of each stream devoted to data, (a) object 1, (b) object 2.

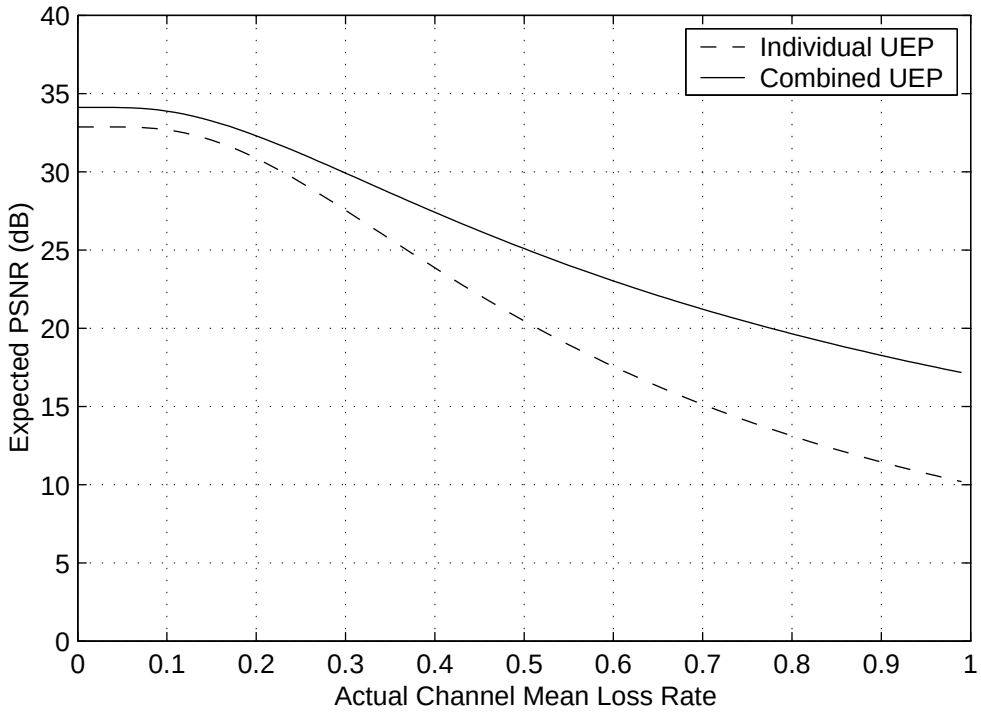


Figure 11: Expected PSNR vs. fraction of packets lost for lenna

(a) (b)

(c) (d)

Figure 12: Reconstructed lenna images at various packet losses, (a) 0% loss for CUEP, PSNR = 34.1 dB, (b) 0% loss for IUEP, PSNR = 32.8 dB, (c) 20% loss for CUEP, PSNR = 34.1 dB, (d) 20% loss for IUEP, PSNR = 32.8 dB.

(e) (f)

(g) (h)

Figure 13: Reconstructed lenna images at various packet losses, (a) 40% loss for CUEP, PSNR = 29.6 dB, (b) 40% loss for IUEP, PSNR = 27.6 dB, (c) 50% loss for CUEP, PSNR = 28.8 dB, (d) 50% loss for IUEP, PSNR = 25.5 dB.

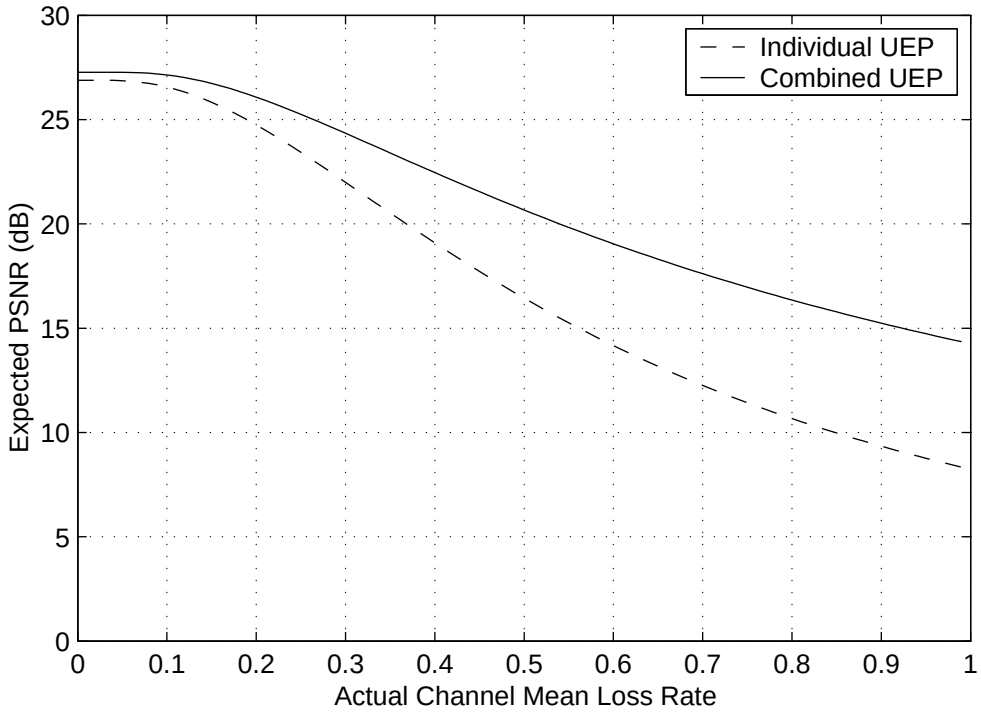
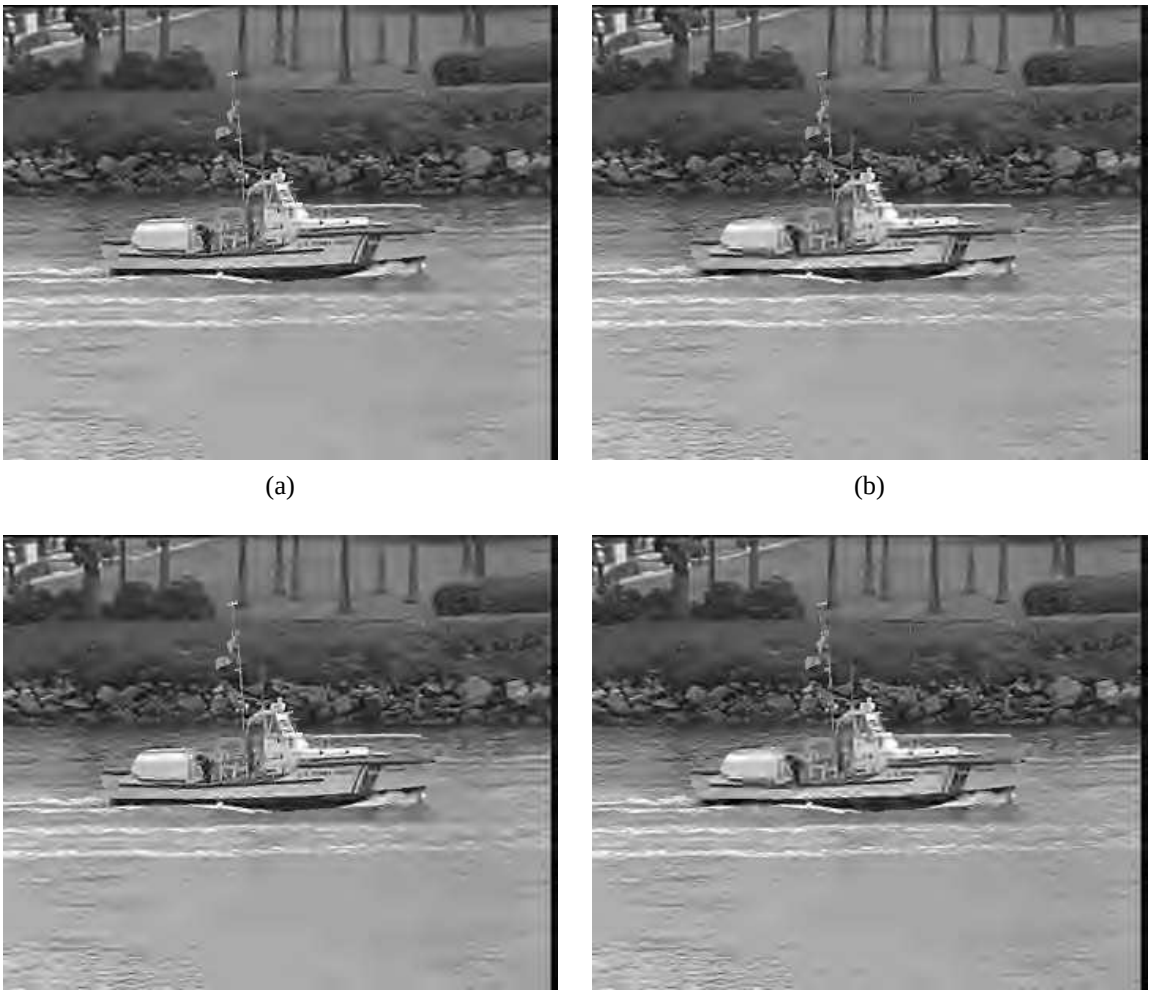


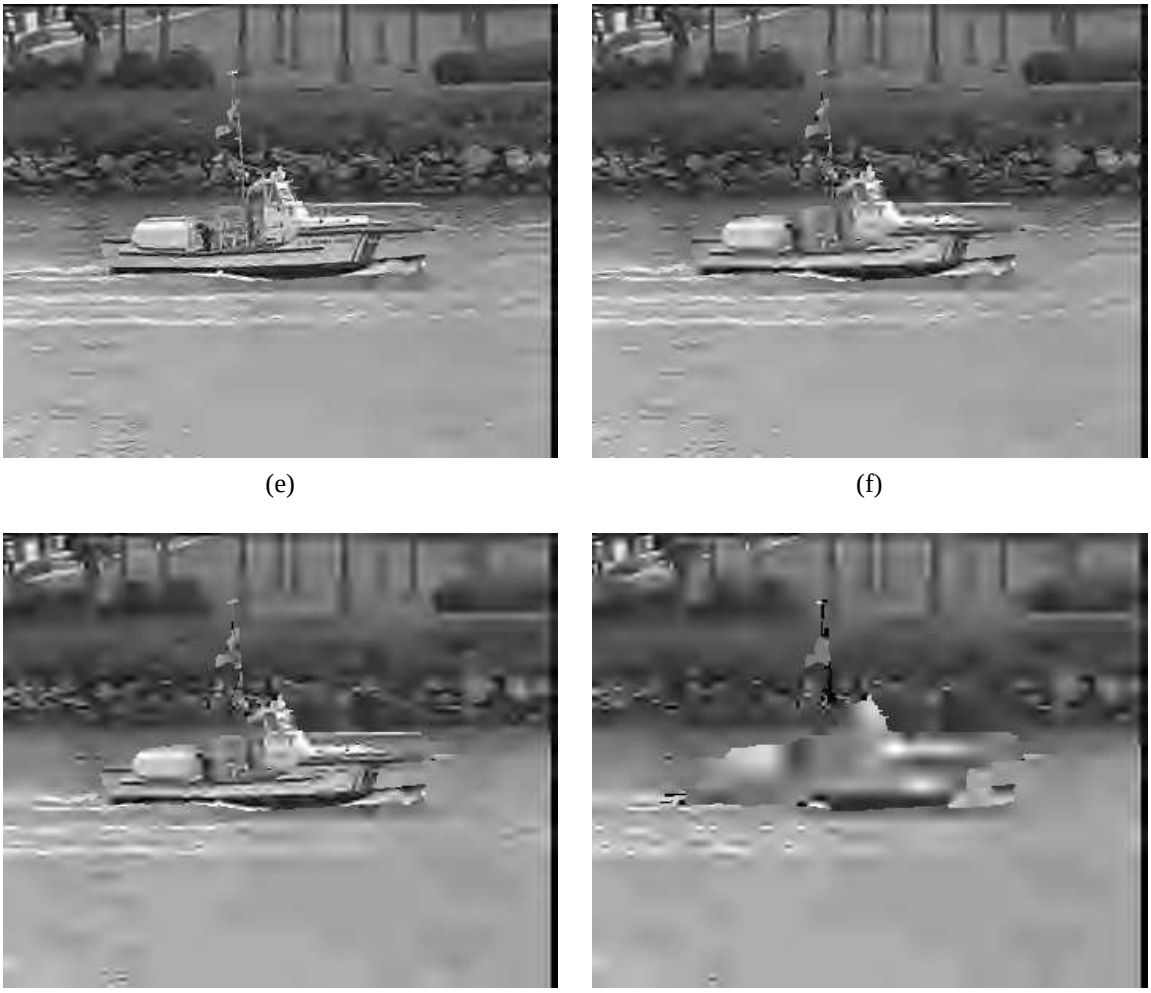
Figure 14: Expected PSNR-vs.-fraction of packets lost for coastguard.



(a) (b)

(c) (d)

Figure 15: Reconstructed coastguard images at various packet losses, (a) 0% loss for CUEP, PSNR = 27.2 dB, (b) 0% loss for IUEP, PSNR = 26.8 dB, (c) 25% loss for CUEP, PSNR = 27.2 dB, (d) 25% loss for IUEP, PSNR = 26.8 dB.



(e) (f)

(g) (h)

Figure 16: Reconstructed coastguard images at various packet losses, (a) 35% loss for CUEP, PSNR = 26.3 dB, (b) 35% loss for IUEP, PSNR = 24.5 dB, (c) 45% loss for CUEP, PSNR = 23.7 dB, (d) 45% loss for IUEP, PSNR = 21.0 dB.

Appendix A: Calculations with PSNR-vs.-Prefix Profiles

Let $D(x_1, x_2)$ be the PSNR of the composite image (i.e., whole-image PSNR) when x_1 bytes are decoded from object 1 and x_2 bytes are decoded from object 2. These x_i bytes may be the first x_i bytes of an embedded bitstream for object i, in which case, the following describes how to calculate a global PSNR profile from individual-object profiles for CUEP on an interleaved bitstream. That is, object 1 is reconstructed from bitstream B_1 with x_1 bytes and object 2 is reconstructed from bitstream B_2 with x_2 bytes. $D(x_1)$ and $D(x_2)$ are the single-object PSNR values taken from the individual PSNR profiles of object 1 and 2, respectively. The corresponding MSEs are

$$d(x_i) = 255^2 10^{\frac{-D(x_i)}{10}}. \tag{11}$$

Now, the whole-image PSNR $D(x_1, x_2)$ is calculated from the PSNR profiles of individual objects as

$$D(x_1, x_2) = 10 \log_{10} \frac{255^2}{d(x_1, x_2)},$$

where

$$d(x_1, x_2) = \frac{K_1 d(x_1) + K_2 d(x_2)}{K_1 + K_2},$$

and K_i is the total number of pixels in object i. Evaluating these equations for all combinations (x_1, x_2) produces the global PSNR-vs.-prefix profile.

Alternatively, these same equations can be used in IUEP to calculate the expected whole-image PSNR from individual-object PSNR profiles; in this case, x_i denotes the number of packets lost from object i such that $D(x_i)$ is the single-object PSNR that arises when x_i packets are lost (which can be obtained from the PSNR profile for object i by figuring out the length of the prefix available for decoding when x_i packets are lost). K_i would then be the total number of packets in object i.

References

[1] M. Ghanbari, *Video Coding: An Introduction to Standard Codecs*, The Institution of Electrical Engineers, London, United Kingdom, 1999.

[2] ISO/IEC 14496-2, *Information Technology—Coding of Audio-Visual Objects—Part 2: Visual*, 1999, MPEG-4 Coding Standard.

[3] R. Machiraju, J. E. Fowler, D. Thompson, B. Soni, and W. Schroeder, "EVITA - Efficient Visualization and Interrogation of Tera-Scale Data," in *Data Mining for Scientific and Engineering Applications*, R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. R. Namburu, Eds., chapter 15, pp. 257–279. Kluwer Academic Publishers, Norwell, MA, 2001.

[4] A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Unequal Loss Protection: Graceful Degradation of Image Quality Over Packet Erasure Channels Through Forward Error Correction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 819–828, June 2000.

[5] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, December 1993.

[6] A. Said and W. A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.

[7] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, April 1991.

[8] M.-J. Tsai, J. D. Villasenor, and F. Chen, "Stack-Run Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 5, pp. 519–521, October 1996.

[9] S. Li and W. Li, "Shape-Adaptive Discrete Wavelet Transforms for Arbitrary Shaped Visual Object Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 5, pp. 725–743, August 2000.

[10] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *ACM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, April 1997.

[11] A. Alanese, J. Blömer, J. Elmonds, M. Luby, and M. Sudan, "Priority Encoding Transmission," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1737–1744, November 1996.

[12] R. Puri and K. Ramchandran, "Multiple Description Source Coding through Forward Error Correction Codes," in *Proceedings of the* 33^{rd} *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 1999.

[13] A. E. Mohr, R. E. Ladner, and E. A. Riskin, "Approximately Optimal Assignment for Unequal Loss Protection," in *Proceedings of the International Conference on Image Processing*, Vancouver, Canada, September 2000, pp. 367–370.

[14] J. E. Fowler, "QccPack: An Open-Source Software Library for Quantization, Compression, and Coding," in *Applications of Digital Image Processing XXIII*, A. G. Tescher, Ed., San Diego, CA, August 2000, Proc. SPIE 4115, pp. 294–301.