ERROR CONCEALMENT USING DATA HIDING

Peng Yin *Bede Liu*

Information Science and System
Electrical Engineering Department
Princeton University, Princeton, NJ 08544

Hong Heather Yu

Panasonic Information &
Networking Technology Laboratory
Princeton, NJ 08540

ABSTRACT

Error concealment plays an important role in combating transmission errors. Methods of error concealment that produce better quality are generally of higher complexity, thus making some of the more sophisticated algorithms not suitable for real-time applications or are restricted to client devices with limited capability. In this paper, we propose an approach to use data hiding to facilitate the error concealment at the decoder. A set of features are extracted at the encoder and embedded imperceptibly into the host media. If some part of the media data is damaged during the transmission, the embedded features can be extracted and used for recovery of lost data. The use of data hiding leads to reduced complexity at the decoder. Simulation shows that our approach has better image quality than some well-known conventional error concealment methods.

1. INTRODUCTION

Transmission over networks of digital multimedia data is increasingly popular. Media data, especially in compressed form can be quite vulnerable to imperfect channels, because a single error bit may lead to objectionable visual distortion at the decoder [1], making the combating of transmission errors an important problem. There are two general approaches to error resilient communication: error control and error concealment. Error control aims at lossless recovery, treating the media stream as any other types of data. Error concealment, on the other hand, makes use of inherent characteristics of the data such as spatial or temporal correlations, and attempts to obtain a close approximation of the original signal that is least objectionable to human perception [1]. In this paper, we shall only discuss the problems of error concealment.

Various approaches have been proposed for error concealment with different tradeoff between complexity and quality [1]. Algorithm based on linear interpolation is computationally simple, but may produce severe blocky artifacts [2]. More powerful algorithms, such as those proposed in [3] and [4], can alleviate blocky effects. But they are computationally intensive, thus are not suitable for real-time applications and for some capability-limited client devices.

Performing error concealment at the decoder generally consists of two steps. First, the decoder estimates some features of lost information. The features may be edge information to help spatial interpolation [3] [4], or motion mode and motion vectors for temporal error concealment [5]. Second, the decoder interpolates lost information from the estimated features using spatial, transform-domain, or temporal interpolation. The first step is essential to recovery and is computationally costly. It is thus desirable to shift the burden of the first step to the encoder, so that the decoder needs only to perform the second step using features already extracted by the encoder. In addition, performing feature extraction at the encoder is more effective as the encoder usually have access to more information of the data [6]. Though the complexity of the encoder is increased, the encoder usually has more computational resources and often can perform the tasks off-line.

The extracted features are side information to the media data and must be sent to the decoder. One way is to attach it in a separate header, but this will increase bit-rate. For compressed bit stream, this increase can be avoided by careful bit allocation using complicated rate control [7]. Another approach is to use data hiding, which can embed the features into the media without increasing the bit rate and without introducing perceptible artifacts [6] [7]. The additional computation for data hiding can be made minimal by using such approaches as odd-even embedding [8].

In this paper, we propose a general scheme of using data hiding to facilitate error concealment at the decoder. The scheme is outlined in Section 2. A detailed example is then given in Section 3.

This work was performed while the first author was a summer intern at PINTL and partially supported by a NJ State R & D Excellence Grant. The authors can be contacted at {pengyin,liu}@ee.princeton.edu,heathery@research.panasonic.com

2. GENERAL ARCHITECTURE

The general system layout for our scheme is shown in Figure 1.

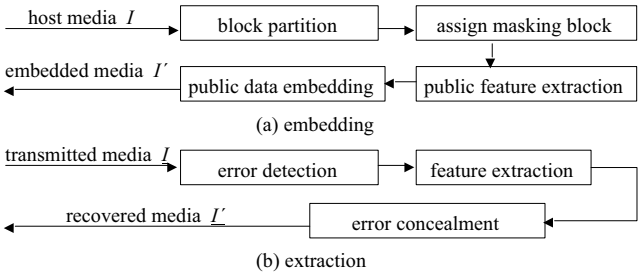

Fig. 1. General Architecture

We shall use images to discuss our approach and shall use block-based embedding instead of global embedding. This is to prevent having both original data and feature data lost in the transmission [6]. To be compliant with most standard codecs, the host image is first divided into 8×8 blocks. Each block A has associated with a designated companion block Ac into which the features of block A is embedded. At the decoder, if part or all of block A is lost, the features in the companion block Ac and the neighboring blocks of A are used to recover A. In what follows, we will refer block A as the host block. We note that by extracting the features at the encoder, the computation burden of the decoder is reduced.

There are two general approaches to data hiding: private data hiding and public data hiding. Private data hiding requires the original media to extract the embedded information while public data hiding does not. Only public hiding schemes are applicable for error concealment, since the original media may not be available at the decoder.

Since security is generally not involved, the embedded data is also public. In order to avoid visual artifacts caused by the embedded data, a human visual model will be used. Another consideration is that we need the data hiding scheme to have reasonably high hiding capacity, while robustness against attack is not an important consideration [6].

In the next session, we will illustrate by a detailed example how the scheme of Figure 1 actually works using multi-directional interpolation method from [3] and [4].

3. ERROR CONCEALMENT

3.1. Edge Detection and Bilinear Interpolation

Multi-directional interpolation method in [3] and [4] attempts to derive local geometric information of lost or damaged data from surrounding pixels, and use this information for "content dependent" interpolation. Here we assume that damaged image regions can be correctly detected and corrupted area is block-based .

A multi-directional interpolation method generally has two steps:

1. Estimate local geometrical structure, such as edge directions, of damaged block from surrounding correctly received blocks;
2. Interpolate damaged block along edges by surrounding correctly received blocks.

How good the interpolation of Step 2 will produce is critically dependent on Step 1. In [3], Step 1 consumes roughly 30% of total computations; it consumes more in [4], as iterations are required to estimate edge directions.

Unlike in [3] and [4], the encoder of our proposed scheme extracts edge directions from the content block itself instead of estimating them at the decoder from the surrounding blocks. It is assumed that the edge can be approximated as a straight line in a small block. To determine the edge directions, we employ Robert gradient operators [9]. The gradient component of local edge for the pixel are first computed by

$$g_y = u_{i+1,j-1} - u_{i-1,j-1} + 2u_{i+1,j} - 2u_{i-1,j} + u_{i+1,j+1} - u_{i-1,j+1}, \quad (1)$$

$$g_x = u_{i-1,j+1} - u_{i-1,j-1} + 2u_{i,j+1} - 2u_{i,j-1} + u_{i+1,j+1} - u_{i+1,j-1}. \quad (2)$$

where g_x and g_y denote the horizontal and vertical gradients respectively. The gradient vector magnitude and direction at coordinate (i, j) are given by

$$g(i,j) = \sqrt{g_x^2 + g_y^2}, \quad (3)$$

$$\theta(i,j) = \tan^{-1}(\frac{g_y}{g_x}). \quad (4)$$

The pixel location (i, j) is declared as an edge point if $g_{i,j}$ exceeds some threshold t. The content block is denoted as a *smooth block* if there is no edge point in it. Otherwise, it is denoted as an *edge block*. For an edge block, the edge orientation for those edge points is quantized to one of m equally spaced directions in the range $0° - 180°$,

$$q\theta_{ind}(i,j) = \lfloor \frac{\theta(i,j) + \pi/(2m)}{\pi/m} \rfloor \quad (5)$$

where $q\theta_{ind}(i,j) \in \{0, \ldots, m-1\}$.

Due to limited embedding capacity, only one edge is embedded as side information even if the content block has more than one edge. We select the one edge by using majority voting

$$q\theta_{ind} = \{k | \sum_{q\theta_{ind}(p,q)=k} g(p,q) \geq \sum_{q\theta_{ind}(p,q)=l} g(p,q), \quad l = 0, \ldots, m-1, \quad l \neq k\}, \quad (6)$$

$$q\theta = q\theta_{ind} \times \pi/m + \pi/2m. \quad (7)$$

where (p, q) is the coordinate of edge points in the content block, $q\theta$ is the angular of the edge and $q\theta_{ind}$ is the corresponding index. We enlarge the block size for edge detection at the encoder by adding two nearest surrounding layers of the content block to reduce the blocky artifact of error concealment. In embedding, we use 1 bit to denote whether the content block is a smooth block or an edge block and $b = \lceil log_2^m \rceil$ bits to denote the edge direction index $q\theta_{ind}$. Thus 1 bit embedding is required for smooth blocks and $1 + b$ bits for edge blocks. The features of the content block are embedded into its companion block.

If a lost block is detected at the decoder, its block type and edge features are extracted from its companion block and then bilinear interpolation is performed to reconstruct the missing block as in Figure 2. If the lost block is smooth, the bilinear interpolation of pixel P is as in 2(a) .

$$P = \frac{[\frac{d_2}{d_1+d_2}P_1 + \frac{d_1}{d_1+d_2}P_2 + \frac{d_4}{d_3+d_4}P_3 + \frac{d_3}{d_3+d_4}P_4]}{2} \quad (8)$$

where P_1, P_2, P_3 and P_4 are the neighboring pixels on the inner layer of surrounding blocks [2]. If the lost block is an edge block, the bilinear interpolation of pixel P along edge direction θ is as in 2(b).

$$P = \frac{d_2}{d_1 + d_2}P_1 + \frac{d_1}{d_1 + d_2}P_2 \quad (9)$$

where P_1 and P_2 are linearly interpolated from their two nearest neighboring pixels on the inner layer of surrounding blocks [3], and $d_i = |P - P_i|$.

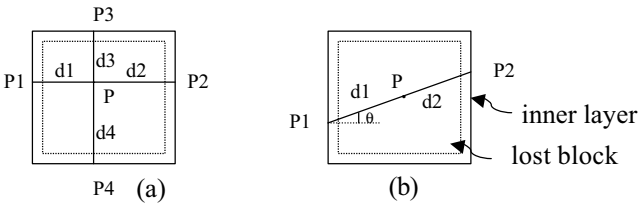

Fig. 2. Bilinear Interpolation (a)smooth block (b)edge block

3.2. Embedding and Extraction

Multimedia data is often sent in compressed form. Here we illustrate our approach using JPEG or MPEG like standards. In particular, we embed the features of a content block into the DCT coefficients of its companion block. For simplicity, we adopt odd-even embedding where a coefficient value is forced to be even to embed a '0' and to be odd to embed a '1' [8]. Such enforcement is performed after quantization. Embedding is done in low frequency DCT coefficients. However, to avoid visual artifact, DC coefficients and the first two lowest coefficients are left unchanged. Extraction can be done by reversing the processing of embedding. In order to avoid the situation that both the host and its companion block are lost, the two blocks should be separated as far as possible. In addition, we do not embed features of block A into block B and features of block B into block A. Instead, we use circular embedding [6]. That is, we embed features of A into B and features of B into C, and features of C into A.

3.3. Block Interleaving

For the proposed error concealment scheme to be effective, the corrupted blocks must be isolated. But in the case of sending compressed data, the concealment of contiguously damaged blocks is harder than that for isolated blocks. To minimize the likelihood of loss of contiguous blocks and to facilitate recovery, a simple 2-way and a simple 4-way block interleaving as illustrated in Figure 3 have been incorporated into JPEG like coding [1], with only a slight penalty in compression efficiency and processing delay.

X O X O X O X O	A B A B A B A B
O X O X O X O X	C D C D C D C D
X O X O X O X O	A B A B A B A B
O X O X O X O X	C D C D C D C D
(a) 2-way interleaving transmit all X's followed by all O's	(b) 4-way interleaving transmit all A's, followed by all B's, then all C's, then all D's.

Fig. 3. Block Interleaving

Figure 4 shows the main processing flow of our scheme. An image is first partitioned into blocks, and features are extracted for each block. The blocks are then interleaved, and the DCT of each block is taken and the coefficients quantized. Circular embedding is applied, followed by entropy encoding to form a JPEG image. When generating a JPEG image, we use self-synchronization codeword at the beginning of each scan row of blocks, so a transmission error in one block will only cause damage of a single row. At the decoder, error detection is first performed to find the damaged blocks and the features are then extracted from their companion blocks. After decompression and de-interleaving, bilinear interpolation is employed to recover the lost blocks.

3.4. Experimental Result

The 512×512 "Lenna" image is used as an example in our simulation. The block size is 8×8, and the quality factor for JPEG compression is 75. Edge direction is classified into 16 sets and 4-way interleaving is employed. We compare our result with geometrical-structure-based error concealment (GSBEC) of [3]. We assume a quarter of the blocks are lost, i.e., one out of every four blocks is missing. Figure 5 shows

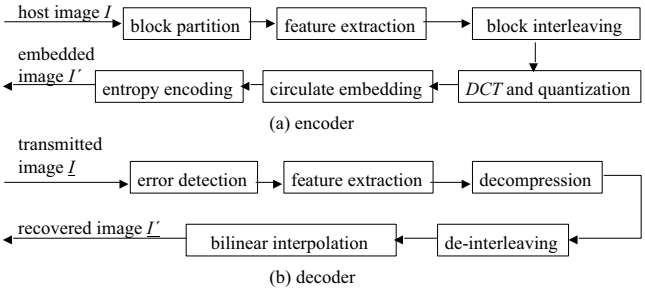

Fig. 4. Main Processing Flow

our result is comparable to [3], with 0.4dB improvement in PSNR. There is no visual difference between original JPEG image and our marked image which has embedded features. Due to the interleaving, the bit rate increases about 1%. If we attach the features in the header, we increase the bit rate about 5%.

Due to limited embedding capacity, we can only embed one edge. When the lost block has a complex geometric structure, such as corner or streaks, our result is not as good as in [3]. With our scheme, however, we can reduce 30% computations at the decoder than [3].

4. CONCLUSION

We propose in this paper a method to improve the speed of error concealment at the decoder by using data hiding. Throughout the paper, we use images to illustrate our approach. However, the same approach can be applied to video and audio. Issues such as how to fine tune the tradeoff between capacity and quality for data hiding need further investigation. Also, more tests need to be performed on a larger data set.

5. ACKNOWLEDGEMENT

The authors would like to thank Dr. Wenjun Zeng of Sharp Labs and Min Wu from Princeton University for enlightening discussion. In addition, Dr. Wenjun Zeng provides the source code of error concealment algorithm [3].

6. REFERENCES

[1] Yao Wang and Qinfan Zhu, "Error control and concealment for video Communication: an overview", *Proceedings of IEEE*, vol. 86, NO. 5, pp.974-997, May 1998.

[2] S. Aign and K. Fazel, "Temporal and spatial error concealment techniques for hierachical MPEG-2 video codec", *Globlecom'95*, pp.1778-1783

[3] Wenjun Zeng and Bede Liu, "Geometric-structure-based error concealment with novel applications in block-based low bit rate coding," *IEEE Transaction on Circuit System and Video Technology*, pp. 648-665, June 1999.

Fig. 5. Experimental Result:(1)upper: left:original; right:corrupted (2)middle: left:JPEG image, PSNR=36.38dB; right:marked JPEG, PSNR=36.06dB (3)lower:reconstructed image, left:using GSBEC, PSNR=31.95dB; right:using proposed scheme, PSNR=32.32dB

[4] H. Sun and W. Kwork , "Concealment of damaged block transform coded images using projections onto convex sets," *IEEE Transaction on Image Processing*, vol. 4, pp.470-477, April 1995.

[5] W.Lam, A.Reibman, and B.Liu, "Recovery of lost or erroneously received motion vectors," *ICASSP'93*, pp.304-315.

[6] H. Yu and P. Yin, "Multimedia data recovery using information hiding," *Globecom'00*.

[7] P. Yin, M. Wu and B. Liu, "Video Transcoding by Reducing Spatial Resolution", *ICIP'00*.

[8] M. Wu, H. Yu and A. Gelman, "Multi-level data hiding for digital image and video," *SPIE*, vol. 3854, 1999

[9] Anil K. Jain, "Fundamentals of digital image processing." Prentice Hall, 1989