HIGH CAPACITY DATA EMBEDDING IN THE WAVELET DOMAIN

Anshul Sehgal, Ashish Jagmohan, Narendra Ahuja

Beckman Institute, University of Illinois, Urbana - 60801
{asehgal,jagmohan,nahuja}@uiuc.edu

ABSTRACT

A novel solution to the problem of data embedding in images is proposed in this paper. The proposed algorithm allows high capacity data embedding and is robust to JPEG image compression. Data is embedded in the wavelet domain which provides better perceptual masking compared to the DCT domain. Set Partitioning in Hierarchical Trees (SPIHT) is used to control the distortion (in the sense of PSNR) in the embedded host. Unlike other data embedding algorithms available in literature, the proposed algorithm provides control over the BER of the embedded data by appropriately choosing the JPEG quantization matrix. Preliminary results of an implementation of the algorithm are also presented.

1. INTRODUCTION

Data embedding/hiding refers to the process of inserting digital data in multimedia signals such as audio and video in an imperceptible manner. Data embedding has numerous applications such as embedding of meta-data for image/video indexing, multilingual subtitles in video, covert communications etc.

Data embedding is closely related to the problem of watermarking or image steganography which aims at embedding imperceptible signatures in multimedia streams for validation of ownership rights. Steganography requires the embedding of small amounts of signature data in the host stream. Data embedding aims at embedding comparatively larger amounts of data. Moreover, watermarking algorithms aim at embedding the signature in a manner robust to attacks specifically aimed at removing the signature information from the embedded stream. On the other hand, data hiding algorithms need only be robust to image manipulations like image compression

2. DATA EMBEDDING REQUIREMENTS

In this paper, we address the issues involved in designing an algorithm that facilitates high capacity data embedding in images while ensuring that the extra information is as imperceptible as possible. While embedding data, one needs to keep in mind the fact that the embedded host needs to be compressed for distribution purposes. Hence, it is desirable that any embedding algorithm be robust to lossy image compression. Data embedding and lossy image compression have conflicting interests. The aim of data embedding is to hide data in an imperceptible manner. Lossy compression aims at removing all perceptually irrelevant information from the image in order to maximize compression. Thus, embedding involves either addition of perceptible data (thereby degrading perceptual quality of the host), or the making of imperceptible changes to the original image requiring reduced compression and therefore an increase in the transmission rate. In our view, any data embedding algorithm should allow this tradeoff to be made in a controlled fashion. To summarize, the desired properties of an embedding algorithm include:

1. *The embedded data should be robust to image compression using the JPEG algorithm:* JPEG is, currently, the most popular compression standard for images. Hence for widespread applicability, it is advisable to design an algorithm that is robust to JPEG compression [1].
2. *The embedding algorithm should allow control over the quality of the embedded host image with fine granularity:* For a fixed amount of data to be embedded, the algorithm should provide a flexible way of controlling the tradeoff between the degradation of the embedded host and the compressed image size. More specifically, given the desired PSNR of the embedded host and the amount of data to be embedded, it should be possible (and practical) for the algorithm to judiciously decide the JPEG quantization matrix.
3. *The alteration in the original host due to embedding should be imperceptible:* It is well-known that PSNR is not a very good measure of perceptual quality of an image. Thus, ensuring a high PSNR of the host image (after embedding) is not enough to guarantee that the perceptual distortion is small.

Numerous algorithm have been proposed in literature for data embedding. Some of these algorithms embed data in the domain in which the image is finally compressed (for eg. DCT domain for JPEG compression) [1] [2] [3] [4], others embed data in different domains [5] [6]. Irrespective of the domain in which data is embedded, all algorithms aim to embed data in regions of the host image which are robust (in the sense of perceptual image degradation) to small perturbations. Data embedding algorithms can be broadly classified into three types - those using frequency masking, contrast masking or texture masking. Data embedding is carried out in regions of high frequency, high contrast and high texture content, respectively, by these three classes of algorithms. The definitions of these three types of regions are open to interpretation and overlapping, to a certain degree. Typically, embedding algorithms identify those image coefficients (in various transform domains) which are robust to perturbations and embed data by either replacing the least significant bits of these coefficients or by modulating the data onto their LSBs. To our knowledge, none of these algorithms explicitly provide a solution to the problem of judiciously controlling the tradeoff between the host degradation and compression efficiency (the second desired property outlined above).

[1] It should be noted here that the embedded data will be resilient to other compression algorithms too, however the degree of robustness will be lesser as compared to JPEG compression.

0-7803-6725-1/01/$10.00 ©2001 IEEE

In this paper, we propose an algorithm which satisfies all the three desirable properties outlined above. The proposed algorithm identifies coefficients which are robust to perturbation in a novel and efficient manner, which provides good perceptual masking for the embedded data. In addition, the algorithm provides a method to control the host PSNR-compression efficiency tradeoff.

3. PROPOSED ALGORITHM

First, we address the issue of the transform domain that should be used for embedding purposes. *Irrespective of which domain the data is embedded in, owing to the wide-spread use of the JPEG compression standards, it is advisable to finally convert the embedded image to the JPEG format.* The JPEG compression algorithm takes the Block DCT transform of an image and quantizes the resulting coefficients before applying lossless coding techniques. A key issue to be taken into account while embedding data is the loss of embedded data during quantization. One way around this problem is to embed data in the DCT domain (since JPEG uses DCT) after quantization. However, aggressive embedding of data in the DCT domain after quantization leads to checker board patterns in parts of the image. This is due to the poor space localization of the DCT coefficients. The wavelet domain offers good space-frequency localization and models the HVS (which consists of oriented band-pass filters) better than the DCT. Hence intelligent embedding in the wavelet domain would avoid the artifacts characteristic of embedding in the DCT domain and provide better masking of the embedded data. The validity of this observation has been demonstrated in [6].There is very little literature on embedding data in the wavelet domain [6],[5]. Both of these algorithms embed data by altering the least significant bits of the wavelet transform of the host image. However, one drawback of both these approaches is the lack of control over the quality of the decoded host image, since the exact effect of altering a particular coefficient of the host image cannot be precisely estimated. An elegant solution to this problem is to use the SPIHT encoding algorithm [7] for embedding. We emphasize that the SPIHT algorithm is used only for embedding purposes. After embedding, the embedded host is converted back into the JPEG format for distribution.

3.1. Overview of SPIHT

The SPIHT algorithm encodes an image into a priority encoded stream (PES) for transmission purposes, such that each bit in the stream is at least as important as the following bits. The chief characteristics of the SPIHT algorithm are as follows. The wavelet transform coefficients of the image are ordered in non-increasing order-of-magnitude. These ordered wavelet coefficients are then transmitted as a series of bit-planes. If the largest wavelet coefficient is represented by n_{max} bits, then the n^{th} bit-plane consists of the $n_{max} - n^{th}$ MSB of each wavelet coefficient. This implies that the n^{th} bit plane consists of coefficient whose value is greater than or equal to $2^{(n_{max}-n)}$. The ordering of bits within a bit-plane is the same as the order of the wavelet coefficients in which they occur.

The SPIHT stream (Figure 1) consists of significance bits followed by refinement bits. The significance bits define the ordering information based on coefficient magnitudes. The refinement bits code the bit-planes and thus progressively refine the value of the significant wavelet coefficients. In this manner, the bits (forming a binary representation of the image wavelet coefficients) are ordered depending on non-increasing order-of-importance, where the 'importance' of a bit is quantified by the amount by which the bit reduces the mean square distortion between the original and decoded images. As we show next, the structure of the SPIHT stream facilitates embedding of data in a controlled manner - i.e. the effect of the embedded data on the host image can be accurately estimated. Thus, embedding information in the SPIHT stream allows control over the quality of the decoded host image.

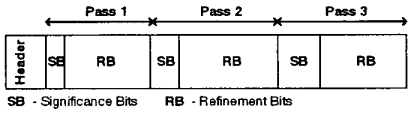

Fig. 1. Structure of the SPIHT stream

3.2. Overview of Proposed Algorithm

In this section, we briefly outline the details of the proposed algorithm. The host image is SPIHT encoded. The SPIHT stream of the host image is truncated at a position B^{mark} (this roughly corresponds discarding coefficients of magnitude smaller than 2^{n_b} where n_b corresponds to the bit-plane of B^{mark}). Thus, for example, if the truncation position B^{mark} corresponds to $n_B = 5$, then the truncated SPIHT stream represents the set of wavelet coefficients, of the host image, whose values are greater than or equal to 2^5. Moreover, this is an ordered set; the ordering is by non-increasing order-of-magnitude. Thus the SPIHT stream represents the ordered set of wavelet coefficients $\{c_1, c_2, ..c_x\}$ where c_1 etc. are greater than or equal to 2^5. It should be noted that these coefficients are effectively quantized by 2^5 since truncating the SPIHT stream removes their last 5 LSBs.

Next, data is embedded in the wavelet coefficients of the host in the order defined by the SPIHT stream. Thus, if b_i is the ith bit of data to be embedded, b_1 is embedded in c_1, b_2 in c_2 and so on. After x bits have been embedded in this fashion, b_{x+1} is again embedded in c_1 and so on. The embedding is done by placing the embedded bit in place of one of the discarded LSBs of the host coefficient (ie one of the LSBs that was lost during truncation). During decoding, the decoder follows the same order for extracting the embedded data.

To see why data is embedded only in coefficients whose values are greater than (or equal to) 2^{n_b}, consider that for proper decoding, the decoder requires knowledge of the embedding order. However, quantization in the DCT domain (during subsequent JPEG coding) would lead to alterations in the magnitude of the wavelet coefficients and could thus potentially change the ordering of the wavelet coefficients (since the ordering is based on their magnitudes). Such an occurrence can be avoided by embedding data only in suitably large coefficients and judiciously choosing the quantization matrix such that the ordering of the stream before B^{mark} remains unaltered after quantization.

3.3. Details of Embedding Algorithm

The embedding algorithm requires the position B^{mark} of the truncation position of the SPIHT stream and the size of the data to be embedded B^E. B^{mark} is set equal to the number of bits required

to SPIHT encode the host image to the desired PSNR. The embedding algorithm truncates the SPIHT stream at B^{mark}, decodes the stream to get the truncated wavelet coefficients and then proceeds to embed the data in the wavelet domain. The embedding algorithm proceeds as follows:

1. Take wavelet transform of left image $\mathcal{I}$ to give $W(\mathcal{I})$.
2. SPIHT encode $W(\mathcal{I})$ to give a priority encoded stream $\psi(\mathcal{I})$ until B^{mark}. While SPIHT encoding store the wavelet coefficients in a list of significant pixels (LSP) in the order in which they appear in the SPIHT stream. (The list LSP contains the ordering information of all the coefficients which become significant until the marker B^{mark})[2]. Let the size of LSP equal N. Also store the pass in which each of the wavelet coefficients in LSP becomes significant in a significance map (SM).
3. Insert a bit sequence M_o at position B^{mark} (M_o is the marker which the decoder subsequently uses to determine B^{mark}).
4. SPIHT decode $\psi(\mathcal{I})$ to give $\hat{W(\mathcal{I})}$, the wavelet coefficients corresponding to the truncated SPIHT stream.
5. Inverse wavelet transform $\hat{W(\mathcal{I})}$ to obtain $\mathcal{I}'$.
6. For $i = 1$ *to* B^E do:
   - Store $\phi(i)$ ($\phi(i)$ represents i^{th} bit of the data stream ϕ), in the $SM(LSP(i \bmod N))$ bit of $LSP(i \bmod N)$ coefficient of $\hat{W(\mathcal{I})}$.
   - Increment $SM(LSP(i \bmod N))$ by 1.
7. Reconstruct $\hat{\mathcal{I}}$ from $\hat{W(\mathcal{I})}$.
8. JPEG encode $\hat{\mathcal{I}}$ to give $J(\hat{\mathcal{I}})$. $J(\hat{\mathcal{I}})$ is transmitted to the decoder.

The decoding algorithm proceeds as follows:

1. Given $J(\hat{\mathcal{I}})$, decode it to give reconstructed image $\tilde{\mathcal{I}}$.[3]
2. Take wavelet transform $W(\tilde{\mathcal{I}})$.
3. SPIHT encode $W(\tilde{\mathcal{I}})$ to give $\psi(\tilde{\mathcal{I}})$ and search through stream for M_o. This recovers the position B^{mark}. While encoding, generate LSP and SM.
4. Using $W(\tilde{\mathcal{I}})$, LSP and SM, proceed in exactly the same manner as step 6 of the encoding algorithm. Instead of storing the embedded bits, read the embedded bits. This gives the extracted data-stream ϕ.

4. EFFECT OF QUANTIZATION

Correct decoding of the embedded data entails that the decoder knows the order in which the data was embedded in the wavelet coefficients at the encoder. As pointed out earlier, the process of quantization during JPEG encoding can potentially alter this order. which would lead to incorrect decoding of the embedded data. Decoding errors are of two types:

1. Errors resulting at the decoder due to the lack of knowledge of the order in which the encoder embedded the data.
2. Errors resulting from the corruption of the embedded data due to the process of JPEG quantization.

Errors of type (1) are far more serious than those of type (2) since they affect the entire stream as compared to a particular bit in embedded data. Hence, specific care must be taken to make sure that errors of type (1) do not occur. It is, of course, also desirable that errors of type (2) occur rarely. Before discussing the choice of the JPEG quantization matrix, we make some observations on the effect of quantization. Consider an *embedded* wavelet coefficient with binary value 111111 (ie decimal value 63), where 100000 (decimal 31) is the truncated wavelet coefficient and 11111 are the bits appended due to embedding. If during the process of quantization the value of this coefficient changes by just 1, the resultant value is 1000000 (decimal 64). Thus not only have all the embedded bits (11111) been altered (to 00000), but the magnitude of the wavelet coefficient (received by the decoder) itself has changed (from 100000 to 1000000). This would results in errors of type 2 and potentially type 1.

One way to avoid such a problem is to use 'cushioning'. This involves setting the LSBs of each coefficient to a string of the form 1000.. (ie a '1' followed by multiple '0's) during embedding. Thus, in the above case if the last 3 LSBs of each coefficient are set to '100' then a quantization error as large as ± 4 can be tolerated without errors to subsequent MSBs within the coefficient. While reducing the number of LSBs in which data can be embedded, cushioning was found to result in reliable performance.

4.1. Characterization of DCT Quantization in Wavelet Domain

In this section, we discuss how how to compute a JPG quantization matrix, given B^{mark}, such that the tradeoff between decoding errors and compression efficiency can be controlled. We model quantization as addition of noise in the DCT domain.

For good compression performance, the wavelet coefficients of the host image are weighted by a perceptual weighting matrix prior to SPIHT encoding and embedding. The weight matrix proposed in [8] is used. Let $\mathbf{P}$ denote the perceptual weighting matrix that is multiplied with the wavelet coefficients before SPIHT encoding and let $\hat{\mathbf{P}}$ denote the inverse perceptual matrix that is multiplied after SPIHT decoding. Since the wavelet transform and the DCT transform are both linear invertible operations, they can be represented by multiplying the image $\mathcal{I}$ (of size $M \times N$ where M, N are multiples of 8) with $\mathbf{T_{DCT}}$ and $\mathbf{T_W}$ respectively. Let B^{mark} denote the position where SPIHT stream of the image is truncated. Let n_b be the bit-plane [7] corresponding to B^{mark}. Let $\mathbf{Q}$ denote the product

$$\mathbf{Q} = \mathbf{T_{DCT}}\mathbf{T_W^T}\hat{\mathbf{P}}\mathbf{I}_{\sigma^2_{n_b}}\hat{\mathbf{P}}^{\mathbf{T}}\mathbf{T_W}\mathbf{T_{DCT}^T} \tag{1}$$

where $\mathbf{I}_{\sigma^2_{n_b}}$ is a diagonal matrix with entries $\sigma^2_{n_b}$, $\sigma^2_{n_b}$ is chosen as:

$$\sigma^2_{n_b} = \rho\frac{(2^{n_b})^2}{12} \tag{2}$$

where the factor ρ controls the probability that decoding errors occur versus the bit-rate required for compression. As ρ tends to 0, the probability of error (of type 1 and type 2) tend to 0. Let $\mathbf{q}$ be the 8×8 DCT quantization matrix with $q(i,j)$ denoting the $(i,j)^{th}$ entry of $\mathbf{q}$ and let $Q(i,j)$ be the $(i,j)^{th}$ entry of $\mathbf{Q}$. Then $q(i,j)$ is given by:

$$q(i,j) = \max_{k,l}\{Q(i+8k, j+8l\} \;\; 0 \le i,j \le 8 \tag{3}$$
$$0 \le k < \frac{M}{8} \;\; 0 \le l < \frac{N}{8}$$

The above result can be derived by an elementary application of the Central Limit Theorem.

[2]The list LSP is generated by the SPIHT encoding algorithm [7].

[3]Note that $\tilde{\mathcal{I}}$ is different from $\hat{\mathcal{I}}$ because of the quantization involved in JPEG encoding

Fig. 2. JPG Coded embedded-host with 7721 bits embedded and JPG Quality Factor=80.

5. RESULTS

The algorithm was implemented on a 256 × 256 8-bit grayscale *lena* image. Table 1 shows the results obtained by truncating the SPIHT stream at a position $B^{mark} = 6255$ bytes. This corresponded to $n^b = 4$ i.e. the 4 LSBs of all wavelet coefficients were discarded. A total of 7721 wavelet coefficients had values greater than 2^4 and were retained for embedding. The 3 LSBs of these coefficients were set to '100' (for cushioning), and data was embedded in the 4^{th} LSB of each of these 7721 coefficients. Thus a total of 7721 bits of data were embedded. Subsequently the embedded host image was JPEG coded at various different quality factors. Column 1 of Table 1 shows the JPEG Quality factor used, column 2 shows the PSNR of the JPEG coded embedded host image and column 3 shows the size of the resultant JPEG image. Columns 4 and 5 show the BER that was achieved with and without cushioning respectively. As can be seen cushioning results in a significant drop in the BER. Also, more than 5000 bits were correctly embedded even down to a quality factor of 70. Figure 2 shows the embedded image for $n^b = 4$ and a JPG quality factor of 80. As can be seen, the blocky artifacts typical in DCT embedding are absent.

Table 2 shows results for $B^{mark} = 3136$ bytes. This corresponded to $n^b = 5$. 3697 coefficients had values greater than 2^5 and bits were embedded in the 4th and 5th LSBs of these coefficients. Thus a total of 7394 bits were embedded and as can be seen from column 4 of Table 2, the BER was significantly lower than for Table 1. The tradeoff is the lower host PSNR caused by the discarding of additional bits.

The results presented here are preliminary. We are in the process of carrying out extensive simulations, especially regarding the computation of custom JPEG quantization matrices, which should improve results still further. Comparisons with other techniques in literature could not be made since complete information required for the comparison was not provided in literature. We are in the process of implementing some of these schemes to make this comparison.

6. CONCLUSIONS AND FUTURE WORK

A high capacity data embedding algorithm that controls the quality of the embedded host image and the BER of the embedded data is presented in this paper. Future work needs to be done on identifying more coefficients where data can be embedded without causing perceptual distortion to the host. One possible way to do this would be to use the fact that for natural images, the modulus maxima of the wavelet coefficients are bounded by an exponential decay with scale (with equality at isolated singularities).

JPG Q Factor	Host PSNR	JPG size (kB)	%BER cushioning	%BER no cushion
95	33.55	23.2	2.6	14.42
90	33.17	17.1	15.41	25.25
85	32.72	13.9	23.05	31.91
80	32.33	12.1	29.19	37.08
75	32.03	10.7	32.6	39.94
70	31.72	9.8	36.2	41.34

Table 1. Results with $B^{mark} = 6255$ bytes, $n^b = 4$

JPG Q Factor	Host PSNR	JPG size (kB)	%BER
95	29.84	20.1	2.65
90	29.74	15.5	8.81
85	29.59	12.78	14.78
80	29.47	11.23	19.43
75	29.33	9.98	23.44
70	29.23	9.2	26.98

Table 2. Results with $B^{mark} = 3136$ bytes, $n^b = 5$

7. REFERENCES

[1] M. Swanson, B. Zhu, A. Tewfik, Data Hiding for Video in Video, *Proc. Of the IEEE International Conference on Image Processing*, vol. 2, pages 676-679, October 1997.

[2] D. Mukherjee, J. Chae, S. Mitra, A Source-Channel Coding Framework for Vector-based Data Hiding in Video, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, Issue 4, June 2000, pages 630 - 645.

[3] F. Hartung, B. Girod, Watermarking of Uncompressed and Compressed Video, *Signal Processing*, vol. 66, no. 3, May 1998.

[4] I. Cox, J. Kilian, T. Leighton, T. Shamoon, Secure Spread Spectrum Watermarking for Multimedia, *NEC Research Institue*, Princeton, NJ, Technical Report95-10, 1995.

[5] W. Zhihui, X. Liang, An Evaluation for Watermarking Techniques,*Proc. of IEEE Int. Multimedia Expo*, Vol. 1, 2000, pages 373 - 376.

[6] M. Ramkumar, A. Akansu, A. Alatan, On the Choice of Transform for Data Hiding in Video, *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing*, Kobe, Japan, 1999, pages 3049 - 3052.

[7] A. Said, W. 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, Issue 3, 1996, pages 243 - 250.

[8] K. T. Lo, X. Zhang, J. Feng, D. Wang, Universal Perceptual Weighted Zero-Tree Coding for Image and Video Compression, it IEE Proc. on Vision, Image and Signal Processing, Vol. 147, Issue 3, June 2000.