ADAPTIVE MULTILAYER REVERSIBLE DATA HIDING USING THE MEAN-TO-PIXEL DIFFERENCE MODIFICATION

Han-Min Tsai, Long-Wen Chang

Department of Computer Science, National Tsing-Hua University, HsinChu, Taiwan

ABSTRACT

In this paper, we propose a new reversible image data hiding algorithm with high data embedding rate and low stego-image distortion. The proposed method is called as a multilayer algorithm. It can embed data in the image repeatedly in a non-overlapping block fashion. In the first layer, the image is divided into 2x2 blocks in a raster-scan order and at most 3 bits can be embedded in each block. In the second layer, the resulting stego-image from the first layer is also divided into 2x2 blocks in a raster-scan order from the second pixel to alleviate the stego-image distortion, and so on. Each layer has its threshold to control its data embedding rate. The threshold is adaptively adjusted according to the ratio of the size of the total data bitstream to the size of the already embedded bitstream. Our experimental results show the proposed method achieves good performance of high data embedding rate and high stego-image quality.

1. INTRODUCTION

Digital image data hiding is the technique that embeds the watermark or secret data into an original image to form a stego-image. The embedded data or watermark can be visible or invisible on a stego-image. If it is invisible, the embedded watermark on stego-image should not be easily perceived. Although the stego-image has only slight difference from the original image, in some specific images like medial, military, and legislation images, even the little distortion on these images can not be accepted. In these applications, the data hiding algorithm should be able to restore the original image from the stego-image. This is the so-called reversible, lossless, or distortion-free data hiding technique.

In [1], Kalker and Willems prove theoretical capacity limits of reversible data hiding schemes which based on lossless compression, and give a practical code construction. In [2], RS vectors are losslessly compressed to vacate space for data embedding. In [3], the authors create the space for data embedding by shifting the histogram between the peak point and the zero point. Ni et al. [4] further improved their method[3] by using more peak points for higher embedding capacity. However, the embedding capacity in [3, 4] is limited by the number of peak points. Celik et al. [5] proposed a lossless data embedding method, which losslessly compresses the little perceptible portions of the original image to produce the extra space for data embedding. In [6], Tian proposed a reversible data hiding using difference expansion, which embeds the data bits into the least significant bit(LSB) of the expanded difference between two adjacent pixels. Kamstra [7] uses the Sweldens' lifting scheme and predicts the LSB-plane using the seven most significant bit-planes to compress the LSB-plane to create space for data embedding.

The methods in [2, 5, 6, 7] losslessly compress the original image features as the side information. For example, the bitstream of lossless compression of the RS-vectors in [2], the bitstream of lossless compression of the least significant portions of an image in [5, 7], and the bitstream of lossless compression of the location map in [6] are side informations. The side information and watermark are together embedded in an image. Thus, if the side information is smaller, there would be more space for data embedding.

In this paper, we losslessly compress the location map of the possible underflow or overflow(PUO) pixels as the side information. In general, the PUO pixels are few in most images such that the side information is small and large data is embedded in the image.

2. REVERSIBLE DATA EMBEDDING AND EXTRACTION

The proposed algorithm(encoder/decoder) embeds and extracts data by modifying the differences between the pixel values of a 2x2 block and the mean value of this block pixels. As indicated in Fig.1, three pixels p_1, p_2 and p_3 are modified to embed data while the pixel p_4 is modified such that the mean value of these four pixels is unchanged after the modification.

The mean value m and the differences between m and p_1, p_2, and p_3 are given by

$$\begin{cases} m = \lfloor \frac{1}{4} \sum_{i=1}^{4} p_i \rfloor. \\ d_i = m - p_i, \text{ where } i = 1, 2, 3. \end{cases} \quad (1)$$

p_1	p_3
p_2	p_4

Fig. 1. A 2x2 block.

The notaion $\lfloor . \rfloor$ denotes the mathematical *floor* operation, for instance, $\lfloor 2.7 \rfloor = 2$, $\lfloor -1.7 \rfloor = -2$. The differences d_1, d_2, and d_3 in the equation (1) are then modified by

$$d_i' = \begin{cases} 2d_i + b_j & \text{if } -T < d_i < T; \\ d_i + sign(d_i) \cdot T & \text{else,} \end{cases} \quad (2)$$

where b_j is the j^{th} bit of the data payload, T is the given threshold, $sign(d_i) = 1$ if $d_i \geq 0$ and $sign(d_i) = -1$ if $d_i < 0$. Note that d_i, $-T < d_i < T$, is the difference that can be embedded data. With the new differences d_i', p_1, p_2, and p_3 are then modified by $p_i' = m - d_i'$, where $i = 1, 2, 3$.

To preserve the same mean value both at the encoder and the decoder, the three differences between p_i' and p_i are summed up and then p_4 is modified by,

$$\begin{cases} \delta & = \sum_{i=1}^{3} (p_i' - p_i) \,. \\ p_4' & = p_4 - \delta. \end{cases} \quad (3)$$

In the decoder, we first calculate the mean value m' using (1). Based on (3), we can easily verify that $m' = m$. Therefore, d_i' can be restored by $m' - p_i' = m - p_i' = d_i'$, where $i = 1, 2, 3$. With d_i', data bit b_j can be extracted by

$$b_j = |d_i'| \bmod 2 \quad \text{if } -2T < d_i' < 2T, \quad (4)$$

where $i = 1, 2, 3$ and $|.|$ represents the absolute value. Then, d_i can be restored by

$$d_i = \begin{cases} \lfloor d_i'/2 \rfloor & \text{if } -2T < d_i' < 2T \\ d_i' - sign(d_i') \cdot T & \text{else,} \end{cases} \quad (5)$$

where $i = 1, 2, 3$. With the original d_i, the original pixel values p_1, p_2, and p_3 are restored by $p_i = m - d_i$, With the original p_i, δ is calculated by $\delta = \sum_{i=1}^{3} (p_i' - p_i)$, and then p_4 is restored by $p_4 = p_4' + \delta$. At present, we have correctly extracted data bits and restored the original pixel values.

In most cases, we just modify pixel values to embed data. However, when pixel values are close to 0 or 255, modifying these pixels may cause underflow or overflow, and we call these possible underflow or overflow pixels as the PUO pixels and the the remaining pixels as the non-PUO pixels. To avoid the problem that pixel values are overflow or underflow after pixels modification, we must pre-process the image before the proposed method launches. To locate the PUO and non-PUO pixels, we generate a binary image where the PUO pixels denoted as '1' and the non-PUO pixels denoted as '0'. The binary image is then compressed by the losslessly compression algorithm JBIG as the side information. Besides, those PUO pixel values must be shifted by

$$p' = \begin{cases} p + 3T & \text{if } p < 3T; \\ p - 3T & \text{if } p > 255 - 3T, \end{cases} \quad (6)$$

where p the PUO pixel value of the original image.

3. FORMAT OF DATA PAYLOAD AND SYNCHRONIZATION INFORMATION

The proposed method is a multilayer embedding algorithm. The format of data payload P and synchronization information(SI) of each layer should be the same both at the encoder and decoder. The side information s must be embedded in the stego-image to restore the original image. Thus, the data payload P consists of s and watermark w of each layer, as shown in Fig. 2. For the decoder to distinguish between s and w, P contains a 18-bit $||s||$ denoting the size of s, a $||s||$-bit s denoting the side information bitstream, and a $||w||$-bit w denoting the watermark bitstream, where $||.||$ represents the size of the bitstream,

18	$\|\|s\|\|$	$\|\|w\|\|$
$\|\|s\|\|$	s	w

Fig. 2. The format of data payload P.

The format of SI is shown in Fig. 3. It contains a 3-bit n denoting the n^{th} layer, a 4-bit T denoting the threshold of each layer, a 1-bit F signaling the decoder whether decoding is necessary after extracting the last embedded bit. and a m-bit P_s denoting the size of the data payload P, where m = $\lceil \log_2(\frac{N}{2} \times \frac{N}{2} \times 3) \rceil$ assuming the dimension of the image is $N \times N$.

3	4	1	m
n	T	F	P_s

Fig. 3. The format of synchronization information SI.

In our method, the size of data embedded in each layer is not known till the encoder finishes in each layer. That is to say, P_s is not known till the end of this layer embedding. Therefore, the encoder must reserves enough 2x2 blocks for SI before embedding P in each layer. The encoder uses the fixed threshold T_f, (1), and (2) to count

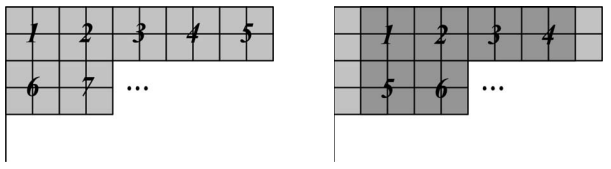
the number of bits that SI needs from the most left-top 2x2 block of the image in a raster-scan order, but without embedding data. The remaining blocks of the image will be used to embed P.

Both the encoder and decoder use the fixed threshold T_f to embed and extract SI. With SI, the decoder can extract P followed by separating s from w. The side information s is then decompressed as the location map to indicate which pixels are shifted before embedding data. Then, the decoder can shift back those shifted pixel values to restore the original pixel values.

4. ADAPTIVE MULTILAYER EMBEDDING

The encoder embeds data in the non-overlapped 2x2 blocks of the original image in a raster-scan order as shown in Fig.4(a). After embedding data in the most right-bottom block of the image, the encoder finishes the first layer embedding and forms the stego-image.

If there exists data payload not embedded, the encoder launches the second layer embedding. It embeds the rest of the data in the stego-image resulted from the first layer. The block embedding order in the second layer is right shifted horizontally with a pixel as shown in Fig. 4(b). With the right shifted operation, it can alleviate the distortion introduced from the first layer. The block embedding order in the third layer is the same as that in the first layer and so on.



(a) The first layer. (b) The second layer.

Fig. 4. The block embedding order for the first and second layer.

In addition, the threshold of each layer is adaptively adjusted according to the ratio of the size of the total data bitstream to the size of the already embedded bitstream. For example, let W be the watermark bitstream that user wants to embed, S and E be the total side information and the already embedded bitstream till this layer, respectively. The threshold T in current layer is adjusted by

$$T' = T + \lfloor (||W|| + ||S||)/||E|| \rfloor, \qquad (7)$$

where T' is the threshold for the next layer. We summarize the embedding steps for each layer as follows.

1. *Reserve enough* 2x2 *blocks for SI in a raster scan order from the left-top of the image.*

2. *Pre-process the image to avoid overflow/underflow using (6).* The encoder uses both T_f and T of each layer to shift the PUO pixels in the reserved blocks for SI and in the remaining blocks for P, respectively.

3. *Embed P in the remaining blocks.* Let e be the embedded bitstream and s be the side information in this layer. When $||e|| < ||s|| + 18$, the encoder must quit embedding because s can not be fully embedded in this layer.

4. *Embed SI in the reserved blocks.* After embedding P in a single layer, the size of P is known such that the synchronization information SI can be embedded in the reserved blocks.

5. *Adaptively adjust the threshold using (7) and go to step 1 if there exists data not embedded.*

Note that, in the reserved blocks for SI, the encoder must use T_f to embed data and to shift the PUO pixels, and the decoder must use T_f to extract data and to shift back those originally PUO pixels. In the remaining blocks for P, both the encoder and decoder use T to do pixels shifting, data embedding, and data extraction.

5. EXPERIMENTAL RESULTS

The watermark bitstreams were pseudo randomly generated with different sizes, the threshold T_f was set as 5, and the threshold T was set as 3 in the first layer. The proposed method was applied to the 512x512 8-bit gray scale images, called Lena, F16, and Mandrill. Table 1 lists the Peak Signal to Noise Ratio(PSNR) under the various watermark sizes measured in bit per pixel(bpp). When the data embedding rate is 1 bpp, the stego-images of Lena and F16 still have high PSNR above 31 dB. Additionally, the proposed scheme can automatically determine the maximum embedding rates for different images. The maximum data embedding rates for Lena, F16, and Mandrill are 1.40, 1.70, and 0.55 bpp, respectively.

Fig.5 shows three stego-images with PSNRs under 0.2bpp, 0.4bpp, and 0.6bpp. Their image quality are quite good. Fig.6 compares the proposed method with the high capacity reversible watermarking schemes [5, 6, 7] in PSNR(dB) vs. Capacity(bpp). When the embedding rate is above 0.5 bpp, the proposed method outperforms all of them more than 3 dB. The proposed method is only slightly worse than [7] below 0.35 bpp. However, the difference is so small that it can be omitted. In general, the PUO pixels are few in most images. Therefore, the PUO and non-PUO pixels are so skewly distributed that the side information can be small. Further, the performance can be improved by a better lossless compres-

sion algorithm like JBIG2. For security, the watermark can be encrypted before embedding.

Capacity(bpp)	Lena	F16	Mandrill
0.05	52.53	51.24	42.01
0.10	48.73	49.10	40.22
0.20	44.83	46.94	34.69
0.40	41.37	42.19	27.65
0.60	37.67	38.66	-
0.80	35.41	36.04	-
1.00	31.82	33.67	-

Table 1. PSNRs for various data embedding rates.

44.83 / 0.2bpp 41.37dB / 0.4bpp 37.67dB / 0.6bpp

Fig. 5. The stego-images for various PSNRs and Capacity.

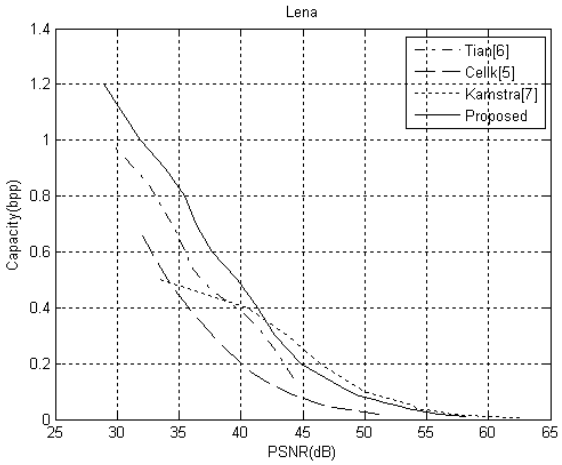


Fig. 6. Capacity(bpp) versus PSNR(dB).

When the size of the watermark was 262144 bits, we applied the proposed method to the Lena image. In Table 2, e is the embedded bitstream in each layer, it contains side information s and the embedded watermark bitstream w for each layer. It shows that the larger threshold T is, the more data can be embedded in the image. However, it also generates more PUO pixels, which increases the overhead of side information.

Layer	1	2	3
e(bits)	119113	136052	25633
s(bits)	1434	2090	15130
T	3	5	6

Table 2. The embedded bistream e, the side information s, and the threshold T in each layer.

6. CONCLUSION

In this paper, the proposed method utilizes the correlation between the mean of the 2x2 block and the block pixels to embed more data. If the data is large, multilayer embedding and adaptively adjusting threshold scheme are used to alleviate the stego-image distortion. In the experimental results, we show that the stego-image has little artifact and the proposed method achieves better performance compared with existing high capacity reversible watermarking schemes [5, 6, 7].

7. REFERENCES

[1] T. Kalker and F. M. J. Willems, "Capacity bounds and constructions for reversible data-hiding," 2002, vol. 1, pp. 71–76 vol.1.

[2] J. Fridrich, M. Goljan, and R. Du, "Lossless data embeddingvnew paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 2, pp. 185–196, 2002.

[3] Zhicheng Ni, Y. Q. Shi, N. Ansari, and Wei Su, "Reversible data hiding," in *Circuits and Systems, 2003.ISCAS '03.Proceedings of the 2003 International Symposium on*, 2003, vol. 2, pp. II–912; II–915 vol.2.

[4] Zhicheng Ni, Yun-Qing Shi, N. Ansari, and Wei Su, "Reversible data hiding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 3, pp. 354–362, 2006.

[5] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-lsb data embedding," *Image Processing, IEEE Transactions on*, vol. 14, no. 2, pp. 253–266, 2005.

[6] J. Tian, "Reversible data embedding using a difference expansion," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 8, pp. 890–896, 2003.

[7] L. Kamstra and H. J. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2082–2090, Dec 2005.