HALFTONE IMAGE DATA HIDING WITH BLOCK-OVERLAPPING PARITY CHECK

Richard Y.M. Li, Oscar C. Au, Carman K. M. Yuk, Shu-Kei Yip, Sui-Yuk Lam

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
E-mail: {ymli, eeau, yukkaman, sukiyip, sylace}@ust.hk

ABSTRACT

In this paper, we propose an innovative algorithm, namely block-overlapping parity check (BOPC), that can be applied to the existing halftone image data hiding algorithms Data Hiding Smart Pair Toggling (DHSPT) to achieve improvement in visual quality. The proposed algorithm utilizes the properties of block-overlapping parity check to reduce the number of pair toggling required in DHSPT. Our experiments suggest that the proposed algorithm reduces the number of pixel pair toggling significantly and has a better performance in terms of Modified Peak-Signal-to-Noise Ratio (MPSNR), especially when the watermark payload is high.

Index Terms— Watermarking, Data hiding, Halftone, Block-Overlapping, Parity Check.

1. INTRODUCTION

Nowadays, multimedia applications become more and more popular. With the advanced technology development in the Internet and wireless technology, numerous images appear widely on accessible web pages. They can be easily downloaded by some unauthorized users for unexpected purposes. Digital watermarking [1] is one of the solutions to prove the ownership and the authenticity by hiding information such as a random sequence or logo into the digital media.

In applications like books, magazines, newspapers, printer outputs and fax documents, the output is constrained to be strictly black-and-white. Halftoning [2] is a process to convert multitone images into black-and-white halftone images, which look like the original images when viewed from a distance. It is often desirable to hide value-adding invisible data within the halftone images such as company logo for copyright protection and authentication purposes.

Most of the existing data hiding algorithms are designed for multitone images, which are not suitable for halftone signals. Some of the existing data hiding algorithms for halftone images modify the halftoning process of the original multitone images [3]. For other existing data hiding algorithms, some hides data by pattern replacing [4] and some performs data hiding by toggling the pixels [5] on the halftone images without the help of the original grayscale images. An algorithm named Data Hiding Smart Toggling (DHSPT) is proposed in [5]. It hides data in a set of pseudo-random position. If DHSPT causes a pixel to toggle, one of the 3 x 3 neighboring 'slave' pixels with opposite halftone value is forced to toggle so as to preserve the local intensity and thus provide good visual quality of the image.

In this paper, we proposed a new data hiding algorithm for halftone images named Block-Overlapping Parity Check (BOPC). This paper is organized as follows. Section 2 describes the proposed watermarking algorithm in details. Section 3 shows the experimental results, such as MPSNR of the watermarked images and the number of pair toggling required by the embedding process under the same payload. Section 4 presents the conclusion.

2. THE PROPOSED BOPC ALGORITHM

The propose BOPC algorithm borrows the strength of DHSPT and adds features to improve performance. DHSPT achieves good visual quality by the smart pair toggling. Artifacts are found, although reduced, in the location of pair toggling. Block-Overlapping Parity Check (BOPC) achieves better visual quality than DHSPT by reducing the number of smart pair toggling under the same payload situation.

In BOPC, a digital logo L with the size of L_X x L_Y is embedded into a host image S with the size of S_X x S_Y to form a watermarked image W. The flow diagram of the embedding process is shown in Fig. 1. BOPC uses several data structure, which we will call the master map, parity map and toggle map. A procedure called bubble formation is applied on the toggle map to choose the locations at which smart pair toggling will be applied. Section 2.1 describes the master map formation and the parity map formation. Section 2.2 presents the details of the toggle map formation and bubble formation. Lastly, the process of watermark insertion is described in Section 2.3.

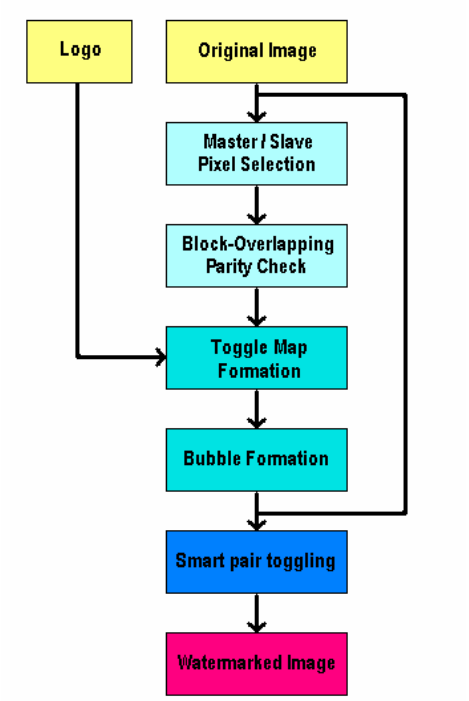

Fig. 1 The Flow Diagram of Watermark Embedding

2.1. Master map and parity map formation

In BOPC, pixels in S are divided into two groups, namely master pixels and slave pixels. Master pixels will be toggled, if necessary, to store the embedded logo data. When a master pixel is toggled, a neighboring slave pixel is typically toggled in a complementary way to preserve the local intensity, similar to DHSPT. A pseudo-random number generator with a known seed, K, is used to generate a set of $(2L_X+1)$ x $(2L_Y+1)$ pseudo-random locations on S. These are our master pixels which we put together to form a master map M as shown in Fig 2, and the corresponding location are stored in a table which we call the location mapping table.

In smart pair toggling process that we will discuss in section 2.3, a pair of neighboring master pixel and slave pixel is involved in pair toggling. It is recommended that a master pixel should be surrounded by eight slave pixels in the 3 x 3 neighborhood. The elements in the master map M are divided into overlapping blocks with the size of 3 x 3 as shown in Fig. 3. BOPC hides one bit in the parity of a block in M. A parity map, P, with the size of L_X x L_Y, is formed by block-overlapping parity check with equation (1).

$$I_{ij} = \left(\sum_{x=2i-1}^{2i+1} \sum_{y=2j-1}^{2j+1} M_{xy} \right) \bmod 2 \qquad (1)$$

During the watermark extraction process, the master map is built by the pseudo-random number generator with the same seed K. A parity map is formed once again by equation (1). Without any attack, the parity map should appear the same as the watermark input at the encoder.

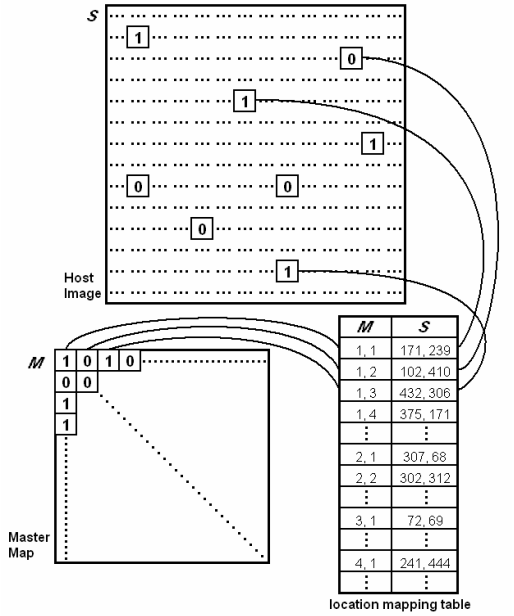

Fig. 2 The master map formation with corresponding location mapping table

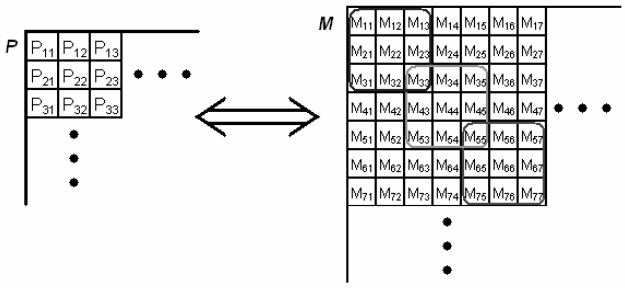

Fig. 3 The corresponding location of an element in P and a block in M

2.2. Toggle map formation and bubble formation

In the locations where P_{ij} is different from L_{ij}, the value of P_{ij} should be toggled. A toggle map, T, is formed by comparing performing pixelwise logical exclusive-OR (XOR) between P & L, using equation (2).

$$T_{ij} = P_{ij} \oplus L_{ij} \qquad (2)$$

If P_{ij} and L_{ij} happen to be the same, T_{ij} will be zero and no toggling is needed. If they are different and T_{ij} is equal to '1', one of pixels in the block centered at $M_{(2i)(2j)}$ are toggled so that P_{ij} will be equal to L_{ij}. In the master map, all blocks overlap with their neighbors. In other words, some master pixels are shared by more than one block. For $1 \le i \le L_X$ and $1 \le j \le L_Y$, the number of blocks sharing a master pixel at a given location is shown in table 1. With the property shown above, BOPC achieves better visual quality by choosing a pixel to toggle in a smart way. As shown in table 1, a single

toggling in M can result in one, two, or four information bits toggling in P. To minimize the total number of toggling in the host image, a process called bubble formation is used.

Table 1. The number of blocks sharing a master pixel

Pixel location	Number of blocks
(2i, 2j)	1
(2i+1, 2j) / (2i, 2j+1)	2
(2i+1, 2j+1)	4

Table 2. Allowable bubble arrangement groups

Group	1	2	3	4
arrangement of bubble	1	2 2	3 3	4 4 4 4

In the bubble formation process, every '1' in T is called an elementary 'bubble' with the size of 1 x 1. Neighboring bubbles can be merged together to form a complex bubble. The allowable bubble arrangements are illustrated in table 2. The goal of the process is to enclose all the '1' in T by using the least number of bubbles. During the process, the elements of T are updated to be the group number of the bubble. The process can be illustrated by an example shown in Fig. 4.

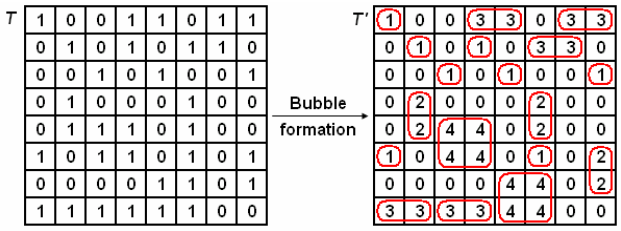

Fig. 4 The bubble formation

The complexity of finding the smallest number of bubbles is very high. To achieve a good solution with much lower computational power, the simulation results in this paper are achieved by performing bubble formation in simply two passes.

T is firstly scanned in raster-scan order. Any group 4 bubbles (2 x 2 neighboring '1') are marked with the value '4'. T is then scanned in raster-scan order again. If an element T_{ij} is '1' and can be grouped into either group 2 or group 3 bubble, we check the number of the consecutive '1' from T_{ij} downward and rightward. If the vertical run is even and horizontal run is odd, we choose group 2 bubble. Otherwise, we choose group 3 bubble.

In bubble formation, the expected number of bubbles per embedded bit is related to the size of the logo. The expected number decreases in two conditions (C1-2):

C1) the dimensions increase in either or both directions.

C2) for the same payload, the difference between the dimensions decreases.

The reason for (C1) is that the more elements in T, the higher chance for small bubbles merging together to form a larger one. (C2) favors the merging of bubbles by providing more interior edges. Recall that every bubble represents a pair toggling in the host image, which led to a drop in visual quality in W. In DHSPT, the expected number of toggling per embedded bit is 0.5. In BOPC, the expected number of bubbles (toggling) per embedded bit for a 128 x 128 logo is 0.26.

2.3. Watermark insertion

After the bubble formation process, the number of bubbles represents the number of pixel toggling required in M. Let T_{ij} be the first element in raster-scan order in a bubble in T, the master pixel chosen by the proposed algorithm for different bubble groups is shown in table 3.

Table 3. The master pixel location decision

Group	Pixel location
1	$M_{2i, 2j}$
2	$M_{2i+1, 2j}$
3	$M_{2i, 2j+1}$
4	$M_{2i+1, 2j+1}$

An example of the process of finding the master pixel to toggle is illustrated in Fig. 5. After finding all the master pixel locations to be toggled, smart pair toggling is performed on the host image. The only constraint is that the complementary toggling partner must be appeared in the slave pixel group defined in section 2.1.

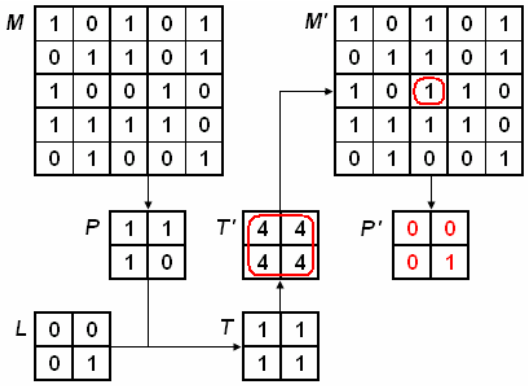

Fig. 5 The flow of master pixel location decision

3. EXPERIMENTAL RESULTS

The BOPC is applied to halftone images generated with error diffusion using Jarvis kernel. The testing images used are Lena, Baboon, Peppers, Barbara, Pentagon, Fishingboat and F16 which are 512 x 512 in size. A typical error diffused image, Lena and the watermark logo are shown in Fig. 6 and 7, respectively. Figs. 8-11 show a portion of

watermarked Lena using DHSPT or BOPC, while 4096 bits are embedded in Figs. 8-9 and 16384 bits are embedded in Figs. 10-11.

Fig. 6 Error diffused Lena

Fig. 7 Watermark logo

Fig. 8 DHSPT with 4096 bits

Fig. 9 BOPC with 4096 bits

Fig. 10 DHSPT with 16384 bits

Fig. 11 BOPC with 16384 bits

From Figs. 8-11, we can find fewer artifacts in BOPC than that in DHSPT. Visual artifacts in both algorithms are mainly caused by pair toggling. The number of total toggling directly affects the visual quality of the watermarked image. BOPC reduces the number of toggling required and thus produces watermarked image with fewer artifacts.

Table 4. The MPSNR of different methods (dB)

	Payload :	4096 bits		16384 bits	
Image	Baseline	DHSPT	BOPC	DHSPT	BOPC
Lena	26.96	26.58	26.77	25.72	26.22
Baboon	22.76	22.60	22.66	22.15	22.39
Peppers	26.56	26.22	26.38	25.39	25.86
Barbara	24.38	24.12	24.25	23.52	23.86
Pentagon	24.46	24.24	24.34	23.66	23.97
Fishingboat	25.73	25.40	25.57	24.66	25.09
F16	25.98	25.68	25.81	24.96	25.36
Average	25.26	24.98	25.11	24.29	24.68

Table 5. The number of pair toggling of different methods

Payload :	4096 bits		16384 bits	
Image	DHSPT	BOPC	DHSPT	BOPC
Lena	2113	1079	8436	4372
Baboon	1971	1088	7914	4304
Peppers	2081	1100	8465	4310
Barbara	2137	1112	8503	4376
Pentagon	2035	1075	8148	4345
Fishingboat	2205	1082	8718	4328
F16	2021	1084	8364	4337
Average	2080	1089	8364	4339

In the experiments, the objective quality measure used is the modified peak signal-to-noise ratio (MPSNR), which is the PSNR between the original multitone image and the lowpass filtered watermarked halftone image. The MPSNR of different images using DHSPT and BOPC are shown in table 4. The comparison of the total pair toggling performed on Lena by the two algorithms is shown in table 5.

From tables 4-5, we can see that BOPC obtains the highest MPSNR under the situation of same payload. We can also find that the number of pair toggling in BOPC is reduced by half when comparing with that of DHSPT.

4. CONCLUSION

In this paper, a new data hiding algorithm for halftone images named Block-Overlapping Parity Check (BOPC) is proposed. Even without the help of original multitone images, it can hide a large amount of data in halftone images. Comparing with DHSPT, BOPC can reduce half of the pair toggling required. The proposed algorithm has a high MPSNR and good visual quality as shown in the experimental results.

5. REFERENCES

[1] F. Mintzer et al., "Effective and ineffective digital watermarks," *in Proc. IEEE Int. Conf. Image Processing, vol. 3*, pp. 9–13, Oct. 1997.

[2] R. A. Ulichney, *Digital Halftoning*. Cambridge, MA: MIT Press, 1987.

[3] J.M. Guo, S.C. Pei, H. Lee, "Watermarking in Halftone Images with Parity-Matched Error Diffusion" *Proc IEEE Int. Conf. On Acoustics, Speech, and Signal Processing Proceeding, vol. 2*, pp. 825 – 828, March 2005

[4] S.K. Yip, O.C. Au, C.W. Ho, H.M. Wong, "PI-Preserve Data Hiding for Halftone Image" *Intelligent Signal Processing and Communication Systems*, pp. 125-128, Dec. 2005

[5] M.S. Fu, Au, O.C, "Data hiding watermarking for halftone images" *Proc. IEEE Transactions on Image Processing, Vo 11, Issue 4*, pp.:477 – 484, April 2002