2009 International Conference on Signal Processing Systems

Invisible and High Capacity Data Hiding in Binary Text Images Based on Use of Edge Pixels

H. Tirandaz, R. Davarzani
Department of Electrical Engineering
Semnan University
Semnan, Iran
{hamedtirandaz, reza.davarzanii}@gmail.com

M. Monemizadeh, J. Haddadnia
Department of Electrical Engineering
Ferdowsi University of Mashhad
Tarbiat Moalem University
mostafamonemizadeh@gmail.com
haddadnia@sttu.ac.ir

***Abstract*—In this paper, a novel blind method for hiding data in binary text images is presented. In this method, the embedding process is limited to the edge pixels of all connected components. In the embedding process, the secret data are located in locations so that to introduce minimal visual distortion. Unlike existing block-based methods, we embed the secret bits directly into closed outer boundary pixels of all connected components in the text image. Experimental results show the high potentials of the introduced method in terms of data capacity and image quality.**

Keywords-binary text image; data hiding; edge pixel; watermarking.

I. INTRODUCTION

Data hiding represents a class of processes used to embed data, such as copyright information into various forms of media such as image, audio, or text with a minimum amount of perceivable degradation to the "host" signal; its goal is not to restrict or regulate access to the host signal, but rather to ensure that embedded data remains inviolate and recoverable.

A watermarking technique makes use of a data hiding scheme to insert some information in the host image, in order to make an assertion about the image later. In this paper, data hiding scheme simply means the technique to embed a sequence of bits in a still image and to extract it afterwards.

Watermarking techniques can be classified as either "robust" or "fragile." Robust watermarks are useful for copyright and ownership assertion purposes. They cannot be easily removed and should resist common image manipulation procedures. On the other hand, fragile watermarks (or authentication watermarks) are easily corrupted by any image processing procedure. However, watermarks for checking the image integrity and authenticity can be fragile because if the watermark is removed, the watermark detection algorithm will correctly report the corruption of the image.

Data hiding and authentication of digital text documents have aroused great interest due to their wide application areas such as legal documents, business documents, certificates, digital books and engineering drawings [1]-[6]. The principles of data hiding in color image, video and audio are similar in that they make use of the redundant information of their host media as well as the characteristics of human visual or auditorial system. Among these media text documents show very peculiar properties: binary nature, block/line/word patterning, and clear separation between foreground and background areas. In other words, in contrast to image, video and audio there is no redundant information in text documents. Therefore, data hiding in binary text images techniques are basically different from that other media.

There are a few of important methods for high capacity data hiding in binary images in references [1]-[6].

Wu [1] and [2] referred a predefined flippability lookup table and flipped some number most flippable pixels to embed one bit in one block. Wu's scheme took watermarked image quality in to account while the capacity is limited for only one hidden bit for one block. In order to enhance the capacity, another method based on using secret keys and weight matrix has also been suggested [3]. This method succeeds in increasing the capacity but the penalty is poor quality of the watermarked image especially when it tries to achieve high capacities. Further improvements on visual quality are obtained by using the edge pixels [4]. Choosing suitable pixels to carry watermark data in a blind data hiding method based on preserving the connectivity of pixels in a local neighborhood is proposed in [5]. In [5], the "flippability" of a pixel is determined by imposing three transition criteria in a 3×3 moving window centered at the pixel. Another new method is proposed which involves data hiding in binary images by using optimal block pattern coding and dynamic programming. Employing this technique, up to three message data bits can be embedded into one 2×2 pixel block of the input binary image [6]. This method achieves fairly high capacity but the data embedding process is very time consuming.

In this paper, we present a high-capacity blind data-hiding scheme for binary text images based on hiding data in the edge pixels. In the proposed data hiding method, the flipping of pixels and embedding process is only limited to the edge pixels of connected components, therefore, alterations in binary text image are hardly noticeable. In the current implementation, we only used the outer boundary of a character to embed data.

978-0-7695-3654-5/09 $25.00 © 2009 IEEE
DOI 10.1109/ICSPS.2009.45

II. Proposed Data Hiding Method for Binary Text Image

A. Definition of Three Types of Edge Lines

In the proposed data hiding method, the flipping of pixels and embedding process is only limited to the edge pixels of all connected components, which correspond to characters or other symbols in a binary text image. Therefore, we should first, have a clear vision of the different edge lines in a binary image. We define an *edge line* as the common sharing 'line' between two neighboring pixels where the pixel values for the two pixels are different. The *edge pixel* refers to pixels (either white or black) on the edge. For example, the edge lines in Fig. 1(a) are those black lines shown in Fig. 1(b).

The scan direction and start point of a binary connected component are two important points to specify the type of edge line, and both of them play an important role in data embedding and extraction processes.

In the embedding or extraction processes, the input image is scanned in a left-to-right, and top-to-bottom manner to extract all connected components. For each connected component, the first upper-left foreground (black) pixel encountered in the scanning process is used as the starting pixel and the outer boundary of a character is then traversed in a *clockwise* direction. An 8-connected boundary following algorithm is then used to obtain the closed outer boundary of a connected component. There are two common edge lines in a binary image, "*vertical edge line (VE line)*" and "*horizontal edge line (HE line)*", as shown in Fig. 2. In this paper we define an imaginary edge line which is derived from the intersection of a vertical edge line and a horizontal edge line and call it "*diagonal edge line (DE line)*". In other words, when the direction of a vertical or horizontal edge line changes by 90 degrees, it forms a diagonal edge line. There are eight different types of diagonal edge lines as shown in Fig. 3 that make up all possible diagonal edge lines in a 2×2 pixel block. The pixels depicted as chequer, are "don't care" pixels which can be black or white pixel.

One important issue in binary images data hiding is how to find "suitable" locations to hide the watermark data so that the visual distortion is low. Flipping some of the edge pixels introduce more visual distortion than the other edge pixels. For example, in Fig. 4, flipping pixels A and B cause larger distortion than flipping pixel C. Based on the visual distortion table in [1] and [2], the flippability score for pixels A, B, and C are 0, 0, and 0.625, respectively.

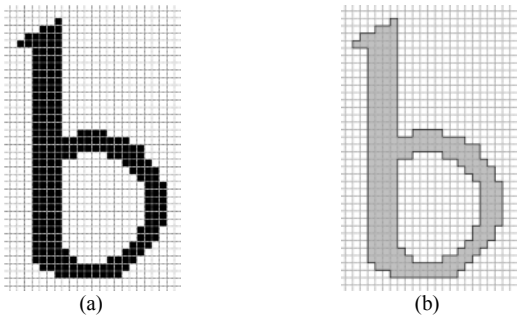

Figure 1. Illustration of edges: (a) enlarged "b" and (b) edge line of "b"

Figure 2. (a) Vertical edge line (VE line), (b) horizontal edge line (HE line)

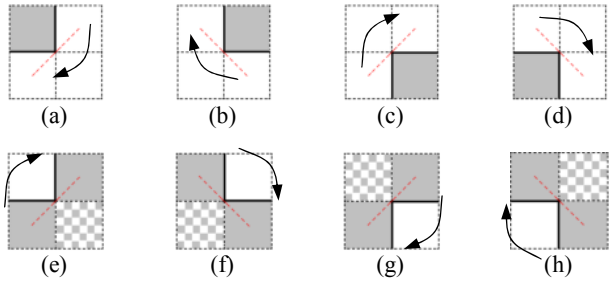

Figure 3. Scan direction and eight different types of diagonal edge lines (DE lines); figures (e), (f), (g) and (h) are considered as embeddable diagonal edge lines (EDE lines).

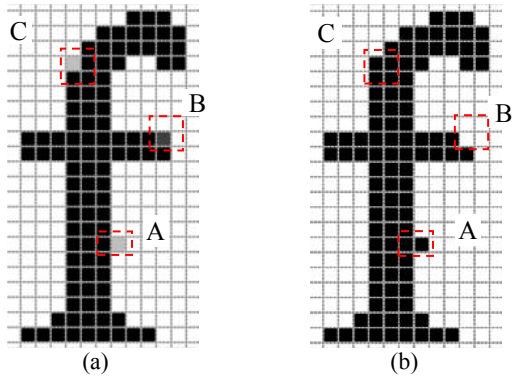

Figure 4. Comparison of visual distortion introduced by flipping different edge pixels. (a) before flipping, (b) after flipping.

B. Embedding Algorithm

In the proposed method, we use the total number of vertical and horizontal edge lines that are between two consecutive embeddable diagonal edge lines as a parameter for blind data hiding. The following example explains the process.

Consider Fig. 5 in which a part of the edge lines in character "b" is magnified. In this figure the starting point of the edge line is an EDE line (EDE line I) and the EDE lines are numbered consecutively with Roman numerals. Also between each two consecutive EDE lines, just VE lines and HE lines are considered and numbered with Arabic numerals, and NEDE lines are ignored. For example, there are two HE lines between EDE line I and EDE line II or there are overall two VE and HE lines between EDE lines II and III, and finally, there is no VE or HE line between EDE lines III and IV. Note that the scan direction of edge is always *clockwise*. In data embedding process, at first, we define the parameter $S(i, i+1)$ as the summation of vertical and horizontal edge lines that are between two consecutive EDE lines i and $(i+1)$. Therefore, for Fig.5, we have:

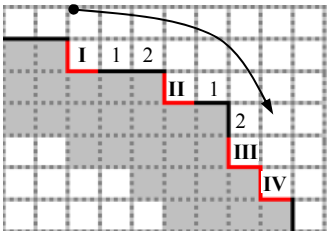

Figure 5. Scan direction, numbering and illustration of EDE, HE and VE lines

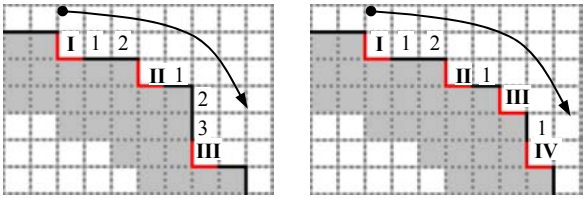

Figure 6. Embedding bit 1 between EDE lines II and III by flipping: (a) black, (b) white edge pixel associated with EDE line III.

$$S(I,II)=2,\quad S(II,III)=2,\quad S(III,IV)=0.$$

For embedding the watermark bits, we use the summation $S(i,i+1)$ as a parameter for data embedding. If this summation is odd, we take it as bit 1 and if this summation is even, we take it as bit 0. For example, suppose that we want to embed two data bits $(01)_2$ in the edge of Fig. 5. Since $S(I,II)=2$, then for embedding the first bit, the condition is satisfied and there is no need to change in the edge pixels between EDE lines I and II. Now, for embedding the second bit, we consider the summation of VE lines and HE lines that are between EDE lines II and III, i.e. $S(II,III)$. As illustrated in Fig. 5, $S(II,III)=2$. In this case, this summation is even while the bit that is to be embedded is 1. Hence, we have to make a change in the edge pixels. For making this change we should shift the EDE line III one pixel. Note that we should not shift the EDE line II at all, because it causes the error in extraction of the first bit which is embedded between EDE lines I and II. For shifting the EDE line III, we can flip one of the associated edge pixels with the EDE line III from black to white or vice versa that both of them are illustrated in Fig. 6(a) and Fig. 6(b). After the shifting the EDE line III, we have $S(II,III)=3$ and $S(II,III)=1$ for Fig. 6(a) and Fig. 6(b), respectively. Therefore, with making a simple and invisible change in the edge pixels of connected components we can embed the desired bits easily.

The main idea of embedding algorithm is described in above, but for perfect implementation of this method, there are some important points which should be considered:

1) As stated above, for modifying the summation $S(i,i+1)$ or equivalently, shifting the EDE line $(i+1)$, we should flip an edge pixel associated with the EDE line $(i+1)$. The flipping an edge pixel from black to white may cause discontinuity in the boundary of a connected component, therefore, except for four special cases (which be discussed later) for modifying the summation $S(i,i+1)$, we only flip the white edge pixel associated with the EDE line

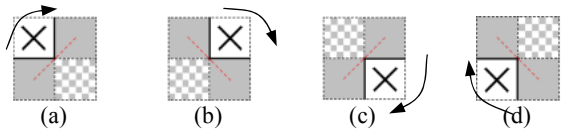

Figure 7. Illustration of flippable white edge pixels for four types of EDE lines.

$(i+1)$. Corresponding to four types of EDE lines for a 2×2 block, flippable white edge pixels are illustrated (with "· " sign) in Fig. 7.

2) There are four special cases that flipping the white edge pixel associated with EDE line $(i+1)$ causes error in both data embedding and extraction processes. These special cases are illustrated in Fig. 8.

In Fig. 8, it is clear that flipping the white edge pixel which is between two consecutive EDE lines, remove both EDE lines i and $(i+1)$. In these special cases for flipping the edge pixels (if needed) we use flipping the black edge pixel associated with the EDE line $(i+1)$. For four special cases in Fig. 8, the flippable black edge pixels which can be used for flipping and modifying $S(i,i+1)$ are illustrated (with "· " sign) in Fig. 9.

3) Some special characters, such as the characters "O" and 'B', contain one or more inner boundaries and their inner boundaries are not used in our current implementation.

4) In some cases, it is likely that by flipping a flippable edge pixel, the desired result is not achieved for $S(i,i+1)$. In such cases, the flipping of the edge pixel should be repeated again. For example, suppose that we want to embed data bit '1' between the EDE lines I and II in Fig. 10. Since $S(I,II)=0$, then we should flip the flippable edge pixel associated with the EDE line II, (Fig. 11(a)). In Fig. 11(a), the summation of VE and HE lines is 2, i.e. $S(I,II)=2$. Therefore, we should repeat flipping of the flippable edge pixel associated with the EDE line II for Fig. 11(a). As shown in Fig.11(b), after second flipping, the desired result is achieved. In other words, after each edge pixel flipping, we should check the result and if desired result was not achieved, the flipping of the flippable edge pixel will be repeated again.

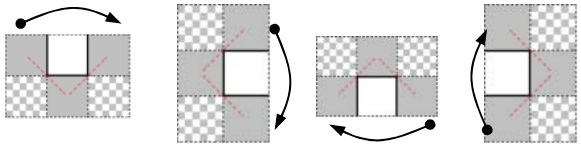

Figure 8. Scan direction and four special cases that flipping the white edge pixel causes error in both data embedding and extraction processes.

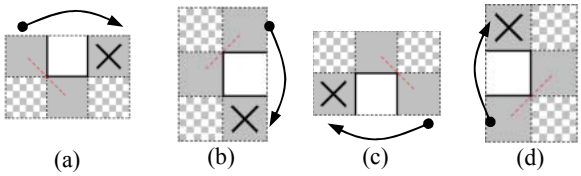

Figure 9. Flippable black edge pixels (with " × " sign) for four special cases.

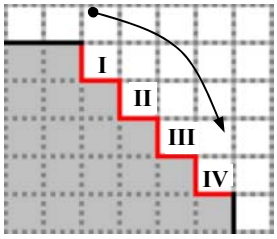

Figure 10. Scan direction and edge pixels before flipping, $S(I,II)$ =0.

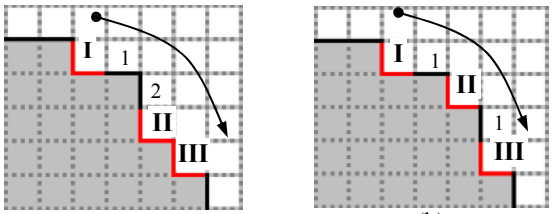

Figure 11. Edge pixels after: (a) first flipping, $S(I,II) = 2$ and (b) second flipping, $S(I,II) = 1$.

C. Extracting Algorithm

Data extraction is actually the reverse process of data embedding and is much simpler than data embedding. In the extraction process, the same procedure as used in the embedding process is used to extract all connected components and outer boundaries of them. For each connected component, embedded bits can be extracted by computing the total number of vertical and horizontal edge lines which are between two consecutive embeddable diagonal edge lines.

III. Experimental Results

A. Capacity an Visual Quality

Some quantitative methods to evaluate the visual distortion caused by flipping pixels in binary images are available in the literature [1] and [2]. The visual distortion table proposed by Wu *et al.* [1], [2] is employed here. The flippability score (*FS*) is obtained by matching the 3×3 pattern centered at the flipped pixel in the look up table. In order to be consistent with the observation that the smaller the number of pixels being flipped, the lower the visual distortion, the complement of flippability score with reference to value "1" is used, namely "distortion score (*DS*)" and given by:

$$DS(i) = 1 - FS(i) \tag{1}$$

where i represents the ith flipped pixel. Obviously, the lower the distortion score, the less the distortion that means the better the visual quality of the watermarked image. The total distortion (*TD*) and average per pixel distortion (*APPD*) introduced by flipping pixels in binary images are defined as

$$TD = \sum_{i=1}^{n} DS(i) \tag{2}$$

$$APPD = TD/n \tag{3}$$

where n denotes the total number of pixels being flipped to embed the data [5]. Three types of text images, including English, Persian, and Chinese text images are used to test the capacity and visual quality of the proposed method. The results are shown in Fig. 12 and Table I. In Table I, the total distortion (TD) and average per pixel distortion (APPD) introduced by flipping pixels in binary text images are indicated. It can be seen from the results that the presented method has proved to offer high capacity while maintaining the consistency and quality of the original image.

B. Comparisons

In order to evaluate the capacity and invisibility of the proposed method and compare the results with that of other known methods (including Wu *et al.* [1], Tseng *et al.* [4] and Yang and Kot [5]) several experiments were conducted. The capacity and the corresponding average per pixel distortion are shown in Tables II and III, respectively. In the tables, the block sizes for other methods are chosen such that larger capacity is obtained under the constraints that the watermarked image is of acceptable visual quality. It can be noticed from Table III that with our proposed method, the capacity achieved is larger than that of Wu *et al.*'s method, Yang and Kot's method and Tseng *et al.*'s method. As observed from Tables II and III, our proposed method slightly degrades the average per pixel distortion of the watermarked images compared those obtained in Wu *et al.*'s approach and Yang and Kot's approach. However, the gain in capacity is significant. Our proposed method achieves better average per pixel distortion than Tseng *et al.*'s approach.

IV. Conclusions

In this paper, we present a high-capacity blind data-hiding scheme for binary text images based on edge pixels. In the proposed data hiding method, the flipping of pixels and embedding process is only limited to the edge pixels of connected components, therefore, alterations in binary text image are hardly noticeable. In the current implementation, we only used the outer boundary of a character to embed data. By using the inner boundary, the data hiding capacity is expected to be further increased. Results of comparative experiments with other methods reinforce the proposed scheme's superiority in being able to attain larger capacity while maintaining acceptable visual distortion.

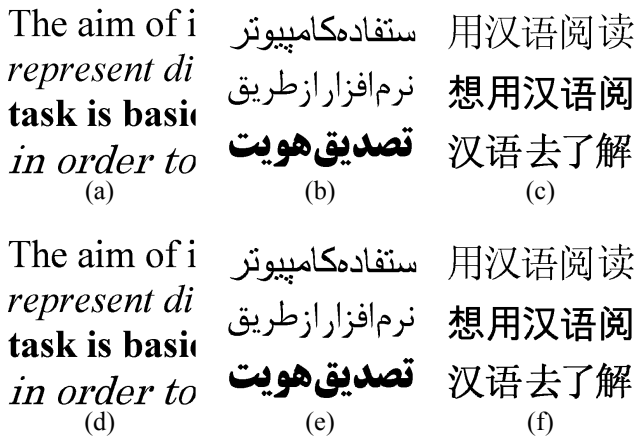

Figure 12. Comparison of hiding effects in different text images. (a), (b), and (c) are original text images of size 485×425. (d), (e), and (f) are after embedding with proposed method.

TABLE I. CAPACITY AND VISUAL QUALITY FOR DIFFERENT TEXT IMAGES

Text image	Capacity	Total Distortion	NO. flipped pixels	APPD
English	2238	666	1356	0.49
Persian	1768	559	1174	0.47
Chinese	2042	668	1292	0.51

TABLE II. CAPACITY FOR DIFFERENT METHODS

Text image	Proposed	Wu [1] 12×12	Tseng [4] 18×18	Yang & Kot [5] *IB* 4×4
English 1	2053	712	1421	1124
English 2	1296	460	846	701
Persian 1	3314	975	2187	1996
Persian 2	1619	630	1215	995
Chinese 1	2105	687	1587	1106
Chinese 2	8315	2473	5791	4300

TABLE III. AVERAGE PER PIXEL DISTORTION FOR DIFFERENT METHODS

Text image	Proposed	Wu [1] 12×12	Tseng [4] 18×18	Yang & Kot [5] *IB* 4×4
English 1	0.5	0.44	0.87	0.47
English 2	0.49	0.45	0.84	0.48
Persian 1	0.47	0.43	0.85	0.46
Persian 2	0.47	0.45	0.86	0.46
Chinese 1	0.51	0.43	0.85	0.47
Chinese 2	0.51	0.42	0.85	0.47

REFERENCES

[1] M. Wu, and B. Liu, "Data hiding in binary images for authentication and annotation", IEEE Trans. Multimedia, vol. 6, no. 4, pp. 528–538, Aug. 2004.

[2] M.Wu, "Multimedia Data Hiding" Ph.D. dissertation, Princeton Univ., Princeton, NJ, Jun. 2001 [Online].Available: http://www.ece.umd.edu/~ minwu/ research/ phd_thesis.html

[3] Y. Tseng, Y. Chen, H. Pan, "A Secure Data Hiding Scheme for Binary Images", IEEE Trans on Communications, vol. 50, no. 8, pp. 1127–1231, August. 2002.

[4] Y. C. Tseng, and H.-K. Pan, "Data hiding in 2-color images", IEEE Trans. Comput., vol. 51, no. 7, pp. 873–878, Jul. 2002.

[5] H. Yang and A. C. Kot, "Pattern-based date hiding for binary images authentication by connectivity-preserving", IEEE Trans. Multimedia, vol. 9, no. 3, pp. 475–486, Apr. 2007.

[6] I.S. Lee and W.H. Tsai, "Data hiding in binary images with distortion-minimizing capabilities by optimal block pattern coding and dynamic programming techniques," IEICE TRANS. INF. &SYST., vol.E90-D, no.8, pp. 1142–1150, Aug. 2007.