# STEGANALYSIS OF BINARY TEXT IMAGES

*Jun Cheng, Alex C. Kot, Jun Liu, Hong Cao*

School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore 639798
cjun@pmail.ntu.edu.sg

## ABSTRACT

We present in this paper a technique for the steganalysis of electronic text documents. The proposed method averages the similar patterns together in a binary text image to estimate the original patterns. The proposed method can detect the existence of a secret message hidden by any boundary flipping techniques as well as estimate the message length and locate the flipped pixels.

## 1. INTRODUCTION

The increasing use of digital documents makes digital document image processing more and more useful. Data-hiding in document images have received much attention recently. One of the applications of data-hiding in document images is steganography. The purpose of steganography is to communicate information secretly so that others who inspect the objects being exchanged won't notice the existence of secret information hidden in the objects. As opposite to steganography, steganalysis is to detect the existence of the secret message in the objects and distinguish objects with secret message from objects without any secret message.

In recent years, some steganalysis techniques have been proposed for LSB embedding in color or gray image [1, 3, 7]. These techniques make use of the fact that the LSB embedding techniques would usually break certain kind of conditions such as smoothness to detect the existence of the hidden message. However, these techniques can not be applied to binary document images which do not have the same smoothness condition as color or gray images. In [2], Jiang *et al.* present a steganalysis technique for binary text images. In this method, the boundaries of characters or symbols in a document are modeled by a cubic polynomial and shaped by cubic splines. Then a boundary pixel is estimated from its neighbors in a window via cubic polynomials. The coefficients of the polynomials are determined using a large training set of unmarked document images. However, those coefficients are related with font size and font type, they are trained separately for characters and symbols of different sizes and fonts.

Many techniques have been developed for data hiding in binary document images. One type of techniques is to manipulate on the line, word or character space by shifting the lines, the words or the characters. However, for these techniques, the existence of secret message can be easily detected by examining the line, word or character space. Moreover, these techniques have relatively small embedding capacity hence the utility for steganography is limited. Another type of techniques for binary image data hiding is to change the value of individually selected pixels, such as the work in [4, 5, 6]. These techniques hide information in the image by flipping pixels. Here, flipping means change the pixel from white to black or vice versa. We call these techniques as pixel-flipping technique. Perceptual quality is controlled in these pixel-flipping techniques to avoid large visible distortions.

In this paper, we proposed a method to detect the existence of secret data in clean text images hidden by the pixel-flipping techniques. These clean images are directly converted from text files. Although digitization error may occur during the conversion, characters/letters/symbols from the same origin are identical after the conversion. We call these kinds of images as clean text images. However, the embedding process which flips some selected boundary pixels in the image will introduce noise to the image.

## 2. EMBEDDING PROCESS

The pixel-flipping technique would flip some pixels for the purpose of hiding information. In order to avoid large visible distortion, most of the pixel-flipping techniques [4, 5, 6] only flip the boundary pixels. Moreover, these techniques normally would not flip two 8-connected neighboring pixels simultaneously, which implies in any $2\times2$ square block in the image, no more than one pixel would be flipped. After data hiding, the same characters/letters/symbols usually no longer match pixel by pixel. However, they normally satisfy the following conditions when we compare them pixel by pixel:

(1) The number of mismatched pixel pairs would be at most 2 in any $2\times2$ square window when they are aligned together. It is because both of the $2\times2$ square blocks

which are from the originally same marks may contain one flipped pixel.

(2) The number of mismatched pixel pairs is always limited compared to the total number of pixel pairs, i.e., the percentage of mismatched pixel pairs is usually less than a threshold.

To detect the existence of secret message, we need to check whether the same characters/letters/symbols match pixel by pixel. In order to get confident detection result, we need to group those originally same symbols together while avoid grouping those originally different symbols together. A possible solution is to use OCR and computer recognition of fonts. However, this would require high computation load. Besides, the image may contain user created fonts, which possibly confuses the font recognition system

## 3. PROPOSED METHOD

The idea of soft pattern matching was proposed to compress binary image in JBIG2 [8, 9]. We use the similar idea here to detect the existence of a hidden message in text images as well as to estimate the message length and the locations of flipped pixels.

### 3.1. Segmentation

We first extract all the marks from the whole image. Here, we use the term 'marks' instead of the previously used 'characters/letters/symbols'. Marks [9] refer to letters, ligatures, figures and punctuation symbols and other symbols. These marks can be easily extracted by any standard segmentation technique. In our implementation, we segment the image into lines according to the horizontal profile and then segment each line into marks according to the vertical profile.

### 3.2. Grouping of marks

We sort all the marks according to their locations in the images and then carry out the following steps for each mark.

1. Start from the first mark and let it be the current mark.
2. Prescreen all the marks sorted after the current mark. Skip if the size is not equal to that of the current mark.
3. Compare all the potential matched marks with the current mark pixel by pixel and calculate the number of mismatched pixels. The two marks can be aligned perfectly as they have equal size. If the percentage of mismatched pixels is less than a predefined threshold $T$ and their matching satisfy condition (1) discussed in section 2, the potential matched mark is considered as matched to the current mark. In our implementation, $T$ is experimentally determined as 13% of the pixel number enclosed in the bounding box of the current mark. All the matched marks would be grouped together. The current mark is considered matched to itself if no other matched mark found, consequently, a group with only one mark formed.
4. Select the first mark of the remaining unmatched marks and let it be the new current mark. Repeat 2, 3 until all the marks have been grouped.

### 3.3. Estimation of the original marks

After the grouping of marks, we get some group of marks. Each group has at least one mark. The marks in one group have equal size in both dimensions; we average all the marks in each group to get the average mark. The average mark is considered as an estimation of the original mark. Some groups may have only one mark in it, which means the mark in this group can only match to itself. We called this kind of mark as unique mark.

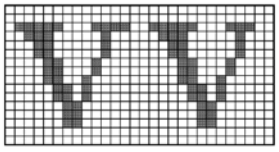
Suppose a group has $N$ marks $M_i$, where $i$=1, 2…$N$. The pixel value (1 or 0) is represented by $M_i(x, y)$, where ($x$, $y$) is the displacement of the pixel from the top-left corner of the mark.

Then the average pixel value at ($x$, $y$) is

$$avg(x,y) = Round\left(\sum_{i=1}^{N} M_i(x,y)/N\right). \quad (1)$$

where, $Round(\cdot)$ is the rounding operation to an integer.

We then use the average mark as a reference and compare it with all the marks in the group. The aggregate of mismatched pixel pairs $N_{MM}$ is counted for all the marks in all the groups. According to [5, 6], the embedding algorithms have 50% chance to flip a selected pixel for hiding a random bit of message. Thus, the message length is estimated by $2N_{MM}$. The above has assumed that the embedding process does not change the size of the marks. However, some embedding process may change it and the message length would be underestimated. For example, the embedding method proposed by Wu et al in [5] may flip the top-right pixel of the left "V" in Figure1 (a) and result in the right "V" in Figure 1 (b). In our previous grouping process, these two characters would not be grouped into one group because their size is not equal.



(a) (b)

Figure 1. Example of changing the mark size

### 3.4. Merging of similar groups

We study all the average marks obtained in each group by equation (1). The average mark for a unique mark is the unique mark itself. Since the size of the average marks may not be equal, we cannot compare the average marks directly. In our implementation, two average marks are aligned to achieve the minimum number of mismatched pixels. If there exist two average marks which satisfy all of the following conditions:

1. The two average marks have a size which differ no more than 2 in either dimension.
2. The number of mismatched pixels is less than a threshold $T^*$ and their matching satisfies condition (1) in section 2 when the two average marks are aligned to achieve the minimum number of mismatch pixels. We use $T^*$ as 5% of the pixel number enclosed in the bounding box of the mark. We use $T^*$ smaller than $T$ because the two average marks have less mismatched pixels if they are estimated from originally same marks.
3. There exists an individual mismatched pixel in the first row or last row if their sizes differ in row dimension. There exists an individual mismatched pixel in the first column or last column if their sizes differ in column dimension.

Then the two groups of marks are considered originally from same marks. We would adjust their size by adding or deleting rows or columns and make their size equal. Here we give an example to explain how we adjust it. For example, suppose the two "v" in Figure 1 are two average marks estimated from the two groups $G_1$ and $G_2$. We append a column of white pixels on the right side of the marks in $G_2$. Then, the marks in the two groups have equal sizes. We merge the two groups to one group. The average mark and the number of flipped pixels would be recalculated.

### 3.5. Detection and Recovery

The average mark for a group is used as the reference to that group. When we compare all the marks in a group with the reference for that group, most of mismatched pixels are the pixel being flipped. We may further recover most of the non-unique marks in an image by replacing the marks in each group with the average mark for that group. The accuracy of the detection and recovery would be discussed later.

## 4. ACCURACY

The accuracy of the estimation of the original mark will affect the accuracy of detection and recovery directly. From our observation, the accuracy of the estimation depends on the embedding algorithm, embedding rate as well as the hidden message. We have studied several approaches [5, 6]. We choose these two approaches as they are representative of pixel-flipping techniques. In our work, we assume the hidden message is randomly generated.

The embedding algorithm reported by Mei et al in [6] hides information using a set of dual five-pixel-long boundary pattern. According to the algorithm, it would choose the same locations in as the candidates for flipping in same contour, i.e., same characters or marks. It won't change the size of the mark and the process in section 3.4 is not necessary here. Wu et al in [5] used a shuffling key to provide random flipping. This method may affect the size of the marks, thus merging process discussed in section 3.4 is necessary. The accuracy of the estimation of pixel value of original marks is discussed below.

For a group of $N$ matched marks, suppose $n$ pixels of the total $N$ pixels at a location $(x, y)$ are used to hide information. Without losing the generality, suppose the $N$ pixels are $M_i(x, y)$, for $i$=1, 2…$n$. Then the probability to get the correct estimation at $(x, y)$ can be computed by:

$$p(x,y) = \begin{cases} 1, & \text{if } n/N < 1/2, \\ \sum_{k=0}^{n_0} \binom{n}{k} \cdot \left(\frac{1}{2}\right)^n, & \text{else}, \end{cases} \quad (2)$$

where, $n_0 = \begin{cases} \lfloor N/2 \rfloor & \text{if original pixel value is 1} \\ \lfloor (N-1)/2 \rfloor & \text{if original pixel vaule is 0} \end{cases}$

and $\lfloor \cdot \rfloor$ is the floor function.

Here we assume the embedding algorithm use odd/even or equivalent features to hide every bit. The number $n$ is related with the embedding scheme as well as the embedding rate. For Mei's scheme, $n/N$ is close or equal to the embedding rate if the pixel at $(x, y)$ is suitable for flipping. For Wu's scheme, $n/N$ is close or equal to the ratio of message length over the total number of pixels suitable for flipping, normally, it is smaller than 0.5.

## 5. EXPERIMENTS

We tested 60 images with different computer generated fonts. The size of these images varies from 500x500 pixels to A4 size. By using the soft pattern matching, marks with different fonts can be grouped into different groups as the sizes of the marks are not equal or they don't satisfy condition (1) and (2). Although the same marks always satisfy the two conditions discussed previously, there is no guarantee that any two marks which satisfy the two conditions must be originally identical. We did observe two such marks in our testing. For example, the comma (,) and the quotation mark (') differ by 1 pixel. We can exclude these marks as their size is smaller than alphabet.

We use the scheme in [5] and [6] to embed random bits in the images at different embedding rate $R$ and then use the

proposed method to estimate the embedding rate and get the estimated embedding rate $R^*$. The estimation error of the embedding rate is defined as $\Delta R = R^* - R$. The estimation errors at different embedding rate for the two embedding approaches are shown in Figure 3 and Figure 4 respectively. The solid line represents the mean error; the dotted line and the dash-dotted line represent the upper bound errors and the lower bound errors. The estimation error increases as embedding rate approach 1. The estimated message length is usually underestimated but we know there's a large amount of data embedded. One reason we get underestimated result is the existence of unique mark especially figures, equations. The other reason is the estimation error from equation (1). From Figure 3 and 4, we can see that the proposed method performs better to estimate the message length hidden by Wu's scheme than by Mei's scheme. This is because the estimation by equation (1) gives better result on image embedded by Wu's scheme than on image embedded by Mei's scheme. We also study the accuracy of locating the flipped pixels and original image. The accuracy is close to the estimation of message length.

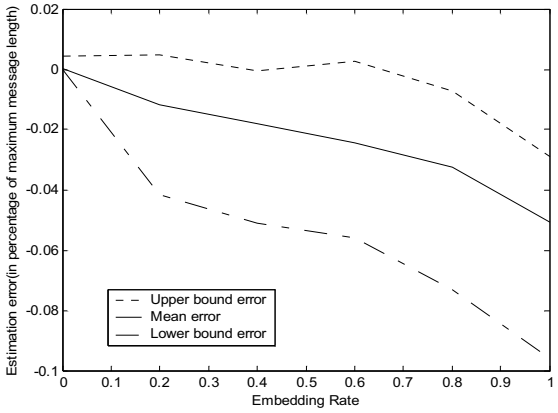



Figure 3. Estimation errors of 60 Images for Wu's scheme

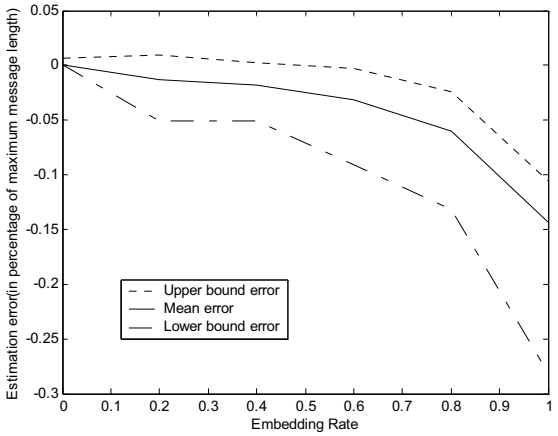



Figure 4. Estimation errors of 60 Images for Mei's scheme

## 6. DISCUSSIONS

Our proposed method can detect the existence of the hidden message as well as the message length. It can estimate the locations of flipped and the original pixel value at these locations. It does not require the knowledge of the details of the embedding algorithm. Hence, it is a general method and it can estimate the flipping rate for most of the boundary pixel-flipping techniques. The limitation of the method is that it only works for clean text image free from scanning errors. It cannot detect some message. For example, we hide several bits in one "s" and make all other "s" identical with the "s" where we hide the information. However, the information can be hidden would be quite lower compared with the method in [4, 5, 6]. Future work would be focused on detecting the existence of message in a noisy image such as a scanned image.

## 7. REFERENCE


[1] J. Fridrich, M.Goljan, and R.Du, "Reliable detection of lsb steganography in color and grayscale images", in *Porc. Of the ACM Workshiop on Multimedia Security,* Ottawa, CA, May 2001, pp.27-30.

[2].M. Jiang, X.Wu, E.K.Wong, and N. Memon, "Steganalysis of boundary-based steganography using autoregressive model of digital boundaries", in *IEEE ICME 2004*, June 2004.

[3]. J. Fridrich, M. Goljan, D. Hogea, and D. Soukal, "Quantitative steganalysis of digital images: Estimating the secret message length," *ACM Multimedia Systems Journal,* vol. 9, no.3, pp. 288-302, 2003.

[4] Haiping Lu, A. C. Kot., Jun Cheng; "Secure data hiding in binary document images for authentication", *ISCAS 2003*. vol. 3. May 2003, pp. III-806 - III-809

[5] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary image," *in IEEE ICME 2000*. New York City, NY, USA, July 2000.

[6] Q. Mei, E.K. Wong, and N.Memon, "Data hiding in binary text documents," *SPIE Proc Security and Watermarking of Multimedia Contents III*, Jan. 2001.

[7] Sorina Dumitrescu and Xiaolin Wu, "Steganalysis of LSB Embedding in Multimedia Signals", in *IEEE ICME 2002*. August 2002 pp:581 – 584. vol.1

[8] Paul G. Howard, "Text image compression using soft pattern matching," *The Computer Journal*, vol. 40, no. 2-3, 1997.

[9] P. Howard and F.Kossentini et al., "The emerging JBIG 2 standard," *IEEE Trans on Circuit and Systems for Video Technilogy*, vol. 8, pp. 838-848, 1998.

[10] J. Fridrich and M. Goljan, "Practical Steganalysis-State of the Art," *Proc. SPIE Photonics Imaging 2002, Security and Watermarking of Multimedia Contents*, vol. 4675, SPIE Press, 2002, pp.1-13.