

Improving Mutli-set Formatted Binary Text Watermarking Using Continuous Line Embedding

C. Culnane, H. Treharne, A.T.S. Ho
Department of Computing
University of Surrey
Guildford, Surrey, GU2 7XH
c.culnane@surrey.ac.uk

Abstract

Our aim is to develop a watermarking method for formatted text documents that is robust to printing and scanning, but with a greater capacity than has previously been achieved. Our end goal is to develop a system with sufficient capacity to embed a simple authenticating watermark. Our method is based on our previous technique of multi-set embedding but now makes use of all word spaces and treats the document as one long line. As part of the modifications we propose a new method of calculating the threshold between letter and word spaces, based on frequency distributions, and have modified the way we conduct threshold buffering. Experiments have been carried out at two different resolutions, 150dpi and 300dpi, on 9 different documents using three different watermarks. We have seen an average increase in capacity of 20 percent whilst also improving the level of robustness to printing and scanning.

1 Introduction

Formatted Binary Text Document watermarking is a specific branch of watermarking that looks to embed data into images of text documents [4]. These images are binary consisting of black and white. The methods used to embed in such documents are often very different to standard embedding techniques seen in image watermarking. This is because the nature of the image is very different, any changes made to a binary image are by definition high contrast changes. Flipping a pixel from black to white, or vice-versa, is as great a change as is possible to make in an image. As a result the capacity within these documents is extremely limited and changes have a tendency to be perceptible.

The other important nature of binary text documents is that they are likely to undergo the print and scan process.

Typically people want to print out their documents, but still want the protection a watermark can provide. By scanning the document back into the computer a digital representation can be easily obtained. It is therefore important that any watermarking method proposed can survive such a process.

Text Document Watermarking has applications in archiving and the legal sector where paper copies are still very common. We recognise that such a method should only be used as a method of authentication as opposed to copyright protection. There is little point attempting to use a watermark, in a binary text document, to copyright protect it. With modern OCR (Optical Character Recognition) a document image can be scanned in and converted to a formatted text document automatically, thus removing the watermark. However, if used as a method of authentication the watermark would not come under such attacks, since the removal of the watermark would result in a document not being able to be authenticated. Therefore the watermark requires protection from being changed as opposed to removal, which falls into the realms of established cryptographical methods.

Our aim is to develop a watermarking method for formatted text documents that is robust to printing and scanning, but with a greater capacity than has previously been achieved. Our end goal is to develop a system with sufficient capacity to embed a simple authenticating watermark. Due to the capacity limitations a suitable authenticating scheme will need to be designed. Before we begin this work we need to maximise the capacity available, so as to give us the best chance of creating a usable authenticating scheme.

Several approaches have been proposed for watermarking of binary documents. Generally they either involve flipping particular pixels or adjusting the feature set. There are several methods based on the flipping of pixels, including [1], [2] and [3]. These methods tend to offer a larger capacity than feature set ones, but are not robust to printing and

scanning. The advantage of feature set methods is that generally they are robust to printing and scanning. The features set methods are based on adjusting a feature of the image, for example the space between words, lines or letters. Zou and Shi [4] proposed a method of embedding using modulated word spaces. However, the proposed method only provided one bit of capacity per line of text. The method formed the basis of our previous work in [5] which aimed to increase the capacity by having multiple sets per line and therefore potentially gain up to two bits of capacity per line. The results in [6] provides further experiments and analysis to those provided in [5].

1.1 Background

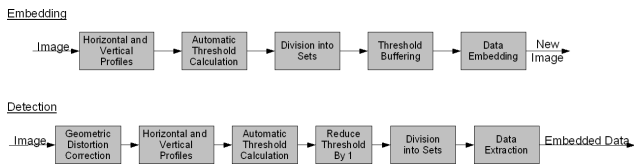


Figure 1: Overview of previous Embedding and Detection Method

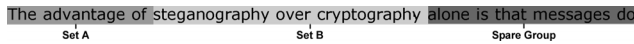


Figure 2: Illustration of Spare Group

Figure 1 shows the flow diagram of our previous work [5]. In it we used horizontal and vertical profiles to divide the document into lines and then each lines into letters and white space. For each line we calculated a threshold, using the mean and standard deviation of the sizes of the white space. This threshold was used to classify the white space as either a letter space, white space between adjacent letters, or a word space, white space between adjacent words. We would then take the word spaces in a line and divided them into pairs of sets. Each set contained three adjacent word spaces. If there were not enough word spaces to form a pair of sets the remaining word spaces would be assigned to a spare group. For example, a line with 12 word spaces would be divided into 2 pairs of sets (4 sets in all) and no spare group. A line with 11 word spaces would be divided into 1 pair of sets (2 sets in all) and a spare group containing 5 word spaces. The division into sets and the spare group are illustrated in Figure 2. Once the spare group is identified the contents are not adjusted in anyway.

For each pair of sets we would make the required adjustment to create the desired detectable difference between the two sets in the pair, the same principle as in [4]. During detection the horizontal and vertical profiles were again used to divide up the document. A threshold was calculated per line and the line divided into sets in the same way as detailed above. For each pair the difference between the total

widths of the sets was calculated and the relevant embedded bit outputted.

2 Method

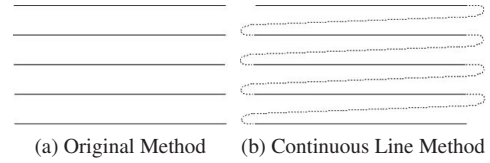


Figure 3: Illustration of the different embedding methods

2.1 Overview

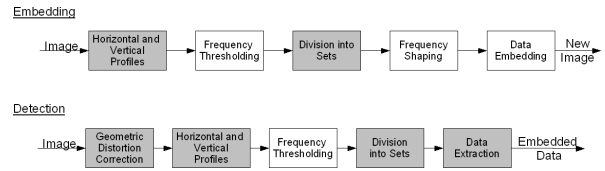


Figure 4: Overview of New Embedding and Detection Method

In this paper the principle of the above method has not changed, we are still aiming to create sets and then create a detectable difference between those sets to embed data. However, how we treat the document and the way we construct the sets has changed significantly. We identified in [6] that a further capacity increase could be gained by making use of the spare group, as discussed above. In order to make use of the spare group we allow sets to be formed from word spaces on multiple lines. In essence, we treat the document as one long line, as can be seen in Figure 3b, in contrast to Figure 3a showing how we previously treated each line independently. By treating the document as one long line we are able to utilise all available word spaces and therefore gain a greater capacity. As a result of this new approach the implementation of the methods shown in Figure 1 needed to be redesigned since they are no longer suitable. Figure 4 gives an overview of the new method. Boxes with a white background indicate new or modified methods. In this section we will detail the changes that have been made to the previous method and explain why.

2.2 Horizontal and Vertical Profiles

We use the same method of Horizontal and Vertical Profiles as in [5] in order to divide the document into lines, words and letters. This process is undertaken during both embedding and detection.

2.3 Frequency Thresholding

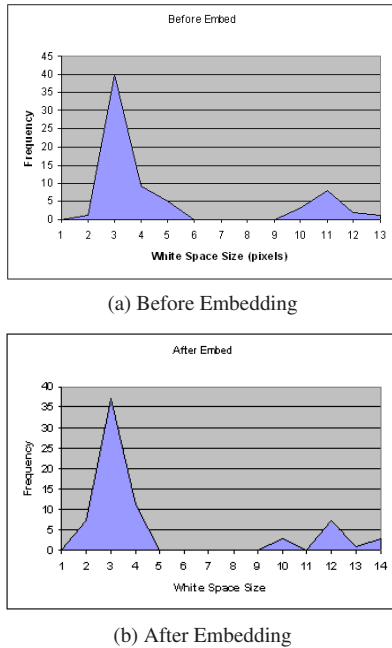


Figure 5: Typical White Space Frequency Distribution Before and After Embedding

The original method of Automatic Threshold Calculation in [5] proved to be no longer adequate with the new embedding strategy. The threshold is required to allow us to distinguish between letter and word spaces. We only want to embed in word spaces since they are more robust to the distortion introduced during printing and scanning and changes to them are less perceptible. The results in [5] and [6] indicated that the method was not perfect and once we had utilised the spare group for embedding, which had previously acted as a control group, we saw a drop in successful zero bit error detections to 44 percent. We therefore had to take a different approach to calculating a threshold. It should be noted that the threshold would still be calculated on a line by line basis.

Figure 5a shows a typical frequency distribution for the white space in a typical line at 150dpi. As can be seen, there are two clear peaks, complete with surrounding values. The peak, and adjacent values, at 3 pixels wide is the peak of letter spaces, whilst the peak and adjacent values, at 11 pixels refers to the word spaces peak. This frequency distribution is typical amongst different fonts at 150dpi. There may be variations in the exact location of the peaks, but through experimentation, it is normal to see two distinct groups, each with a peak in the distribution. By calculating a point between the two groups the ideal threshold can be calculated.

Unfortunately, it is not sufficient to just examine the two peaks and work from these points to calculate the threshold.

Figure 5b shows the effects of embedding on the distribution. The letter space group has narrowed by 1 pixel and become more symmetrical. This is not a problem and if anything would improve robustness by reducing the chances of stray letter spaces being misinterpreted. However, the word space group has changed more significantly. It has been divided into two groups, the second of which contains two peaks. This is to be expected, the word spaces are being made bigger, smaller or left unchanged. As such we would expect to see three different groups. We initially considered taking the mid-point between the two highest peaks as the threshold, but this proved to not be sufficient. In certain circumstances it was possible for the distribution to be such that word spaces could be incorrectly classified, particularly if they were small and fell between the two groups. Misclassification was more likely to happen during detection and after printing and scanning. The print and scan process introduced distortion to the line and to the distribution of white space.

A more sophisticated method of calculation the threshold was developed. We observed that the letter space distribution tended to stay as a connected group around a peak whereas the word space distribution tended to become more fractured, sometimes with significant gaps between groups. This was noted more at 300dpi where there was more scope for changes. We refer to the new profiling method as Directional Weighted Nearest Neighbour Frequency Thresholding.

2.3.1 Directional Weighted Nearest Neighbour Frequency Thresholding

The following is the new procedure for calculating the threshold.

```

Calculate Frequency Distribution
Create list groups of adjacent positive values  $l$ 
Find two groups with the highest peaks in  $l$ 
  Label two groups as  $g_1$  and  $g_2$ 
  Remove  $g_1$  and  $g_2$  from  $l$ 
Order  $l$  in right to left order
For each  $l$  calculate shortest distance to  $g_1$  and  $g_2$ 
  Expand whichever  $g$  is closer to encompass  $l$ 
Threshold is right edge of  $g_1$  + half distance to left
edge of  $g_2$ 

```

The goal of the approach is to create two groups, one representing letter spaces and one representing word spaces. By then finding the point between these two groups a threshold can be calculated. The reason for using a combination of nearest neighbour and right dominance is to better handle the type of distribution often seen in word spaces. Generally we noted that the letter spaces were tightly distributed around a peak, where as the word spaces could become quite spread out. By going from right to left, the word space group will grow more rapidly than the letter space

group, and therefore provide a greater chance of the space being classified as a word space.

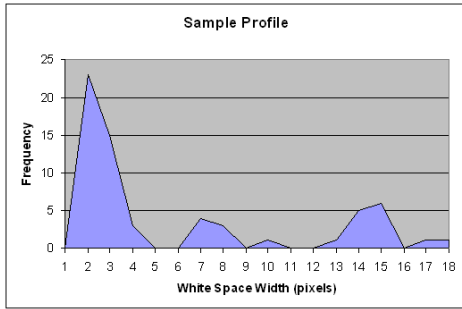


Figure 6: Typical White Space Frequency Distribution Before Embedding

Figure 6 shows an example profile from a document at Times New Roman 150 dpi. This illustrates how the right dominance nearest neighbour works. g_1 is from 1 to 5, whilst g_2 is from 12 to 16. The remaining groups are 16 to 18, 9 to 11 and 6 to 9, all of which are in fact word spaces. If we had simply taken the mid-point between the two peaks or even the two peak groups, we would have incorrectly classified some word spaces. If we had calculated the nearest neighbour from left to right, the group from 6 to 9 would have been classified as letter spaces, since they are only 2 away. However, by using the right dominated nearest neighbour we can see that g_2 is redefined as 12 to 18 in step one. In step 2 it is redefined as 9 to 18. Thus 6 to 9 is now closer to g_2 than g_1 , and therefore the group finally grows to be from 6 to 18 and the threshold is calculated as 4.5 and rounded to 5.

2.4 Division into Sets

The only change in this process was that sets could consist of word spaces from multiple lines. This was a straight forward change that occurred naturally once we treated the document as one continuous line.

2.5 Frequency Shaping

As was mentioned earlier, the methods defined in [5] were no longer suitable for the new embedding method. This included the thresholdBuffering technique which aimed to create a buffer around the threshold to mitigate against the chances of misclassification after printing and scanning. In [5] the threshold was buffered as much as possible before embedding. This was done by adjusting both letter and word spaces to make them less than or more than the threshold \pm thresholdBuffer respectively.

In essence the idea remains the same, although the implementation is different. All word spaces are adjusted to

be at least threshold + thresholdBuffer. In the new method no changes are made to the letter spaces, but a maximum size for a letter space is set to be equal to threshold minus thresholdBuffer. This value is used during the correcting of the line length detailed in Section 2.7.

2.6 Data Embedding

Again, this is virtually identical the previous method, except that now the changes are made without concern for matching changes between sets. This is because the line width correction process will be conducted after embedding and it would be infeasible to maintain the line length during embedding, now that sets can be spread over multiple lines.

2.7 Line Length Correction

We have to make the line width adjustment last because we can no longer match adjustments between sets. When pairs of sets were on the same line we could ensure that any changes made to one were cancelled out by the other, thus ensuring the length of the line was maintained. However, with sets now comprising of word spaces from multiple lines this is no longer possible. It is only possible to correct the line width once all changes have been made, making any changes before this would be inefficient.

We take the opportunity to also shape the profile and thus try to make the thresholding process more robust during the detection phase. Effectively, this step finishes off the thresholdBuffering undertaken during the Frequency Shaping detailed in Section 2.5. As was mentioned in Section 2.5 the letter spaces have a maximum size set, but no adjustments are actually made to the letter space at that point. It is during the following process that the changes are made and that the previously set maximum value comes into play. The following are the steps undertaken during the process to correct the overall line width and shape the profile:

1. Set all letter spaces to be 1 pixel wide
2. Calculate adjustment required to make the line the same width as originally (this should be positive since the act of reducing all letter spaces over compensates for any changes made during embedding)
3. Distribute required changes evenly across letter spaces (whilst still abiding by the maximum size set originally)

The results of this process on the profile can be seen in Figure 7 which shows the profile after embedding. This is in contrast to the profile prior to embedding that can be seen in Figure 6. Figure 7 shows how the letter space peak becomes just a single point. The advantage of doing this

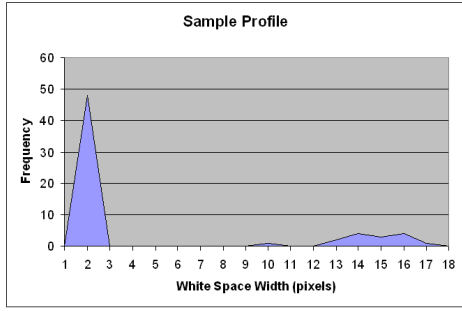


Figure 7: Typical White Space Frequency Distribution After Embedding

is that it reduces the chance of misinterpretation by forcing the letter spaces to all be in the close vicinity to one peak. The disadvantage of this system is that it does not guarantee the line length will remain identical to the original. There is a chance that if there is not enough spare word space a line may extend beyond its original point. We feel it is more important to maintain the embedding as opposed to the line width, since the line width is only of perceptual importance. It is therefore better to maintain the embedded data at the cost of perceptibility than maintain reduced perceptual impact at the cost of the embedded data. During our experiments, excluding the Script font, the overall line width was only incorrect on three occasions from 68 documents consisting of a total of 1149 lines.

3 Experimentation

We used the same test document, as in [5], and conducted the test on one scanner at both 150 and 300 dpi. Table 1 shows the sample watermarks used, whilst Table 2 and Figure 8 show our results. The sample watermarks were shortened for each document, to the nearest letter, to match the relevant capacity. The experiments were conducted on an HP PSC 2110.

Watermark ID	Watermark
A	WATERMARKED EXAMPLE
B	THE EXAMPLE WATERMARK
C	////////////////////

Table 1: Watermarks and ID's

4 Analysis

The proposed method has successfully increased the capacity by on average 20 percent and recovered the watermark with zero bit errors in several different fonts and at different resolutions. In [5] we achieved five zero bit error results in the 150 and 300dpi categories. The new method

Font	Old Cap	Change	Bit Error Rate							
			150 dpi				300dpi			
			Cap	A	B	C	Cap	A	B	C
Arial	88	118%	102	0	6	7	105	0	0	0
Arial Narrow	99	120%	115	0	0	0	122	0	0	0
Comic Sans	69	119%	82	0	0	0	82	0	0	0
Courier New	52	144%	75	0	0	0	75	5	29	30
MS Sans Serif	90	119%	107	0	0	0	108	0	0	0
Script	72	26%	32	23	22	23	6	3	3	2
Tahoma	82	121%	98	0	0	0	100	14	12	0
TNR	90	127%	115	0	0	0	114	0	0	0
Verdana	68	126%	84	0	0	0	87	0	0	0

Table 2: Bit Error Results During Detection - After Print and Scan (Old Cap refers to the capacity obtained in [5], TNR refers to Times New Roman.)

Font	Old Cap	Change	Bit Error Rate							
			150 dpi				300dpi			
			Cap	A	B	C	Cap	A	B	C
Arial	88	118%	102	0	0	0	105	0	0	0
Arial Narrow	99	120%	115	0	0	0	122	0	0	0
Comic Sans	69	119%	82	0	0	0	82	0	0	0
Courier New	52	144%	75	0	0	0	75	17	32	30
MS Sans Serif	90	119%	107	0	0	0	108	0	0	0
Script	72	26%	32	1	1	1	6	0	0	0
Tahoma	82	121%	98	0	0	0	100	0	0	0
TNR	90	127%	115	0	0	0	114	0	0	0
Verdana	68	126%	84	0	0	0	87	0	0	0

Table 3: Bit Error Results During Detection - Before Print and Scan (TNR refers to Times New Roman.)

achieved forty one zero bit errors and an increase in capacity. It should be noted that a part of this increase is as a result of further work carried out as part of [6] which identified the importance of matching the embedding and scanning resolutions. In [6] experiments were conducted on embedding and scanning at the same resolution using three fonts: Comic Sans, Times New Roman and Verdana. For these fonts in the 150 and 300dpi categories 9 zero errors out of a possible 18 were achieved. This is further improved to 18 out of 18 with the new method proposed in this paper. The comparison to the experiments conducted in [6] demonstrates that the new methods not only increase the capacity but also improve the robustness of the watermark beyond that achieved by embedding and scanning at the same resolution.

Arial Narrow, Comic Sans, MS Sans Serif, Times New Roman and Verdana all produced zero bit error results for all three watermarks at both 150dpi and 300dpi. Tahoma and Courier New both successfully detected all three watermarks with zero bit errors at 150dpi, but had bit errors at 300dpi. Arial detected the three watermarks without error at 300dpi and detected one out of the three, without error, at 150dpi. The results from the Script font performed poorly throughout. This was expected since it is a simulated handwriting font and as such does not contain any letter spaces.

As can be seen from Figure 8 the results are slightly better at 150dpi than 300dpi. At 150dpi twenty two out of twenty seven watermarks were detected without zero errors. Whilst at 300dpi nineteen out of 27 watermarks were detected with zero errors. We expected the results at 150dpi to be worse than those at 300dpi. Embedding at 150dpi is a

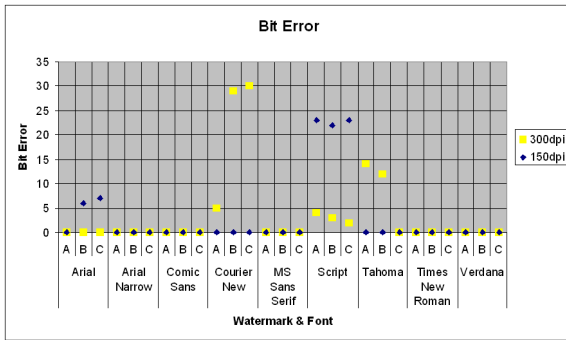


Figure 8: Graph of Bit Errors

more difficult task than embedding at 300dpi, since everything is smaller and there is fundamentally less space available. Also, we would have expected more threshold calculation errors at the lower resolution. The gaps between the peak frequency groups at 150dpi can be as little as 1 pixel, so even the smallest of errors can have detrimental effects. At 300dpi the peaks for letter and word spaces are further apart and as such less likely to be misinterpreted.

Table 3 shows the results when the watermark is detected prior to printing and scanning. If we exclude the Script font, which we expect not to work, 45 out of the 48 results are perfect with zero bit errors. The errors in Table 2 are introduced as a result of the printing and scanning and as we shall see in Section 4.1 and 4.2 are caused by isolated extreme cases. We believe such problems can be rectified by using a more adapt way of identifying weak points in the document and not embedding in them. This would reduce the capacity by only a few bits, but improve the robustness.

4.1 Error Propagation

An undesirable side effect of the new method is that if a single classification error occurs during the thresholding process, the error is propagated throughout the entire document. Previously in [5] errors could only be propagated through a line, but since we now treat the document as one long line, in order to increase capacity, we no longer have anything to stop the propagation. As a result, a single bit error will almost always result in all the remaining bits being incorrect. The large bit error count for the two results for the Tahoma font, and all three results for the Courier New font, at 300dpi were caused by error propagation. Upon inspection there is in fact only a single line with misclassifications of white space but the number of errors appears greater due to propagation.

4.2 Unusual Characters Problem

Unusual characters can be defined as groups of characters that you would not normally expect to see in an english

document, for example a string of eight ones (11111111). Different fonts render this very differently, Verdana groups the 1's together like a word, whilst Arial treats each '1' a single word. The Arial rendering causes problems, the whitespace between each '1' is greater than a letterspace, but less than a word space. This can result in the two peaks in the frequency distribution becoming joined and therefore no word spaces being detected. After printing and scanning the distortion can cause changes in the frequency distribution, so that two peaks are found and therefore new word spaces are found as well, causing bit errors.

5 Conclusion

In this paper we have made further improvements to our multi-set embedding method. In doing so we have proposed a new method of thresholding and threshold buffering. Our experiments have shown that the method successfully increases the capacity by an average of 20 percent. We have also seen an improvement in the robustness to the print and scan process. As a result we have taken another step towards our end goal of providing enough capacity to hold a simple authenticating watermark. We have identified areas of the method that require further improvement and our aim is to focus our efforts on these next.

References

- [1] Qian G. Mei, Edward K. Wong, and Nasir D. Memon. Data hiding in binary text documents. volume 4314, pages 369–375. SPIE, 2001.
- [2] M. Wu, E. Tang, and B. Liu. Data hiding in digital binary images. In *International Conference on Multimedia and Expositions*, volume 1, pages 393–396, 2000.
- [3] A. T. S. Ho, N. B. Puhon, A. Makur, P. Marziliano, and Y. L. Guan. Imperceptible data embedding in sharply-contrasted binary images. In *ICARCV*, volume 2, 2004.
- [4] D. Zou and Y. Q. Shi. Formatted text document data hiding robust to printing, copying and scanning. In *IEEE International Symposium on Circuits and Systems (ISCAS05)*, 2005.
- [5] C. Culnane, H. Treharne, and A. T. S. Ho. A new multi-set modulation technique for increasing hiding capacity of binary watermark for print and scan processes. In *International Workshop on Digital Watermarking (IWDW06)*, pages 96–110. LNCS 4285, 2006.
- [6] C. Culnane. Digital watermarking of binary text documents, robust to print and scan. Master's thesis, University of Surrey, 2006.