DATA HIDING IN HARD-COPY TEXT DOCUMENTS ROBUST TO PRINT, SCAN AND PHOTOCOPY OPERATIONS

Avinash L. Varna
University of Maryland,
College Park, USA.

Shantanu Rane, and Anthony Vetro
Mitsubishi Electric Research Laboratories,
Cambridge, USA.

ABSTRACT

This paper describes a method for hiding data inside printed text documents that is resilient to print/scan and photocopying operations. Using the principle of channel coding with side information, the embedder inserts a message into a text document while treating the content of the document as known interference. The data is embedded by making small changes to text characters before the document is printed. Using a simple correlation-based detector in conjunction with an error correction code, the hidden data can be extracted from a photocopy of the printed document. By enhancing the detector with an optical character recognition algorithm, the embedded data can be extracted even after multiple rounds of photocopying. Results from subjective tests show that the changes made by the embedding algorithm, while perceptible, are not obtrusive to a lay reader.

Index Terms— Watermarking, data hiding, printed text documents, channel coding with encoder side information.

1. INTRODUCTION

A very common mode of leakage of sensitive information is via unauthorized distribution of hard-copy documents. To trace the source of such leakage, it is often necessary to embed tracking information inside a printed document. During an investigation, forensic experts can scan the document and retrieve the embedded information which can convey relevant details such as the identification number of the printer, the date and time at which it was printed, or the IP address of the machine which issued the print command. Further, this information can be a cryptographic hash calculated from the text, which can later be used to authenticate the document and to detect and localize tampering.

Embedding data in printed text documents poses significant challenges. Unlike natural images, text data is extremely structured and hence there is little room to make changes in the text without the changes becoming visible to a human observer. Besides, minute changes made to embed data may be lost when the document is photocopied. Printed documents also undergo wear and tear during normal use that affects the accuracy of extracting the hidden information. The printed document must be converted into a suitable digital format before detection can be performed. The digitization process, e.g., scanning, invariably introduces additional distortion, and the data embedding algorithm must be designed to be robust to such degradations.

There exist a number of schemes which *visibly* embed information into the background of a printed document, for example [1, 2]. These embedding schemes work independently of the printed content; indeed, the printed content obscures the data embedded into the background, making it harder to detect. Further, these techniques can be obtrusive and interfere with the readability of the printed content.

Methods to *unobtrusively* embed data by modulating the distance between successive lines of text [3] or words [4] have been proposed, but these techniques have low embedding rates. Data can also be embedded into individual characters by modulating their grayscale values or halftone patterns [5]. These subtle changes are invisible to the human eye but can be detected by scanning a printed document. However, such schemes are not robust to photocopying operations. An alternative technique to embed data by inserting dots in pseudorandom locations in the background was proposed in [6]. This scheme is robust to one round of photocopying. However, the small dots do not survive multiple rounds of photocopying.

In this paper, we propose a technique to embed data into printed text documents by slightly altering the shapes of certain characters. It is shown that the hidden data can be successfully extracted even after multiple rounds of photocopying. We present results from subjective tests to demonstrate that the embedding algorithm does not cause significant perceptual degradation in the documents.

2. EMBEDDING DATA IN PRINT DOCUMENTS

We adopt the framework of channel coding with side information for embedding information [7] in which the document to be watermarked is treated as known interference at the embedder. Operations such as printing, scanning and photocopying, as well as any intentional modifications made by an attacker are modelled by a noisy channel. The aim of our scheme is to make the watermark resilient to this noisy channel, i.e., the watermark extraction module should be able to extract a reliable estimate of the original watermark even after these modifications. The embedding and extraction modules are described in the subsequent sections.

2.1. Symbol embedding and extraction

Given the text document, we first locate all characters with vertical strokes of length at least l pixels and width at least w pix-

This work was performed while A. Varna was an intern at Mitsubishi Electric Research Laboratories.

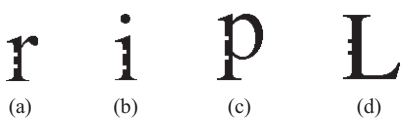


Fig. 1. Characters with different embedded symbols at 300% magnification. The distance between the notches or bumps encodes a symbol.

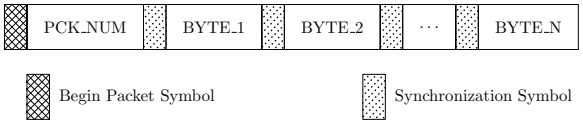


Fig. 2. The embedded data is assembled into packets in order to facilitate synchronization and error correction at the detector.

els. A symbol is embedded into the left edge of the stroke by adding or removing two groups of pixels which are equidistant from the center. The distance between the groups of pixels encodes the different symbols. The size and shape of the group of pixels added or removed can be chosen to trade off robustness for imperceptibility. Modifying more pixels increases the robustness of the symbols to copying and other attacks, but makes them more perceptible. Fig. 1 shows examples of some characters with different embedded symbols.

To extract the embedded data from a scanned grayscale document image Y, we determine the locations of characters with vertical strokes of length at least l' and width at least w' pixels. The values of l' and w' are chosen based on the values of l, w, the printing resolution and the scanning resolution. Once the locations of the vertical strokes have been determined, the embedded symbol is identified by correlating the corresponding stroke from the grayscale image Y with each of the candidate symbols and choosing the symbol with the highest correlation, if the correlation is larger than a threshold.

From our experiments, we observed that some symbols are significantly degraded during photocopying. Such symbols may not be detected at all, which results in a loss of synchronization at the detector. To minimize the data loss due to such errors in the detection process, a packet-based synchronization scheme is used as described in the next subsection.

2.2. Message packetization and ECC

Prior to embedding, the data is grouped into different packets consisting of a header, the data, and synchronization symbols as shown in Fig. 2. The header consists of a *Begin Packet* symbol, followed by the packet number in binary (PCK_NUM). N bytes of the message are interleaved with synchronization symbols and appended to the header to form the packet. A large value of N implies a smaller number of packets per page and hence a lower packetization overhead. However, if a single packet is lost due to desynchronization, then N bytes are lost at once. Thus, the value of N determines the robustness versus overhead tradeoff of the embedding scheme.

To extract the embedded data, the detector looks for the *Begin Packet* symbol and extracts the packet number. If the packet number cannot be correctly extracted, or if the number of synchronization symbols present in the packet is not N, the

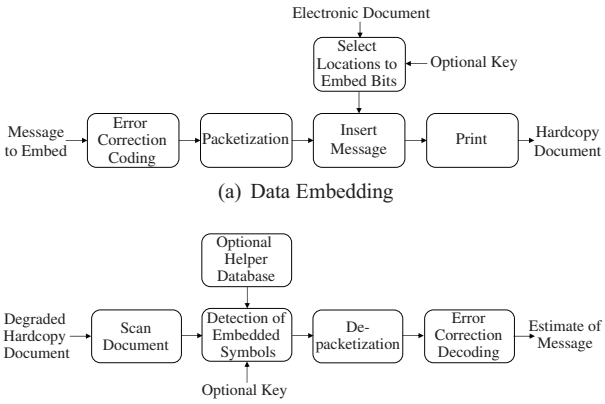


Fig. 3. Insertion and extraction of hidden data using the principle of channel coding with encoder side information.

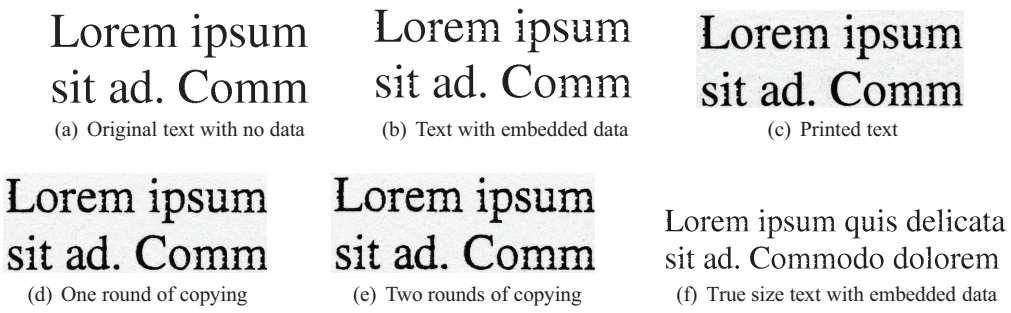
entire packet is treated as an erasure. If a particular byte of a packet cannot be extracted correctly, it is also treated as an erasure. Additionally, some of the data symbols may be detected in error. An error correcting code (ECC), with an appropriately chosen coding rate is used to correct these errors and erasures.

The overall data embedding and extraction procedure is shown in Fig. 3. An error correction code is applied to the message (e.g. a watermark) and the result is assembled into packets. The packets are embedded into the text as described in Section 2.1. The locations at which the data is embedded are selected by a secret key common to the embedder and detector. For data extraction, the document is first scanned and the embedded symbols are identified. The synchronization symbols and packet structure are used to extract the payload. ECC decoding is then performed to correct any errors that may have occurred in the detection process.

3. EXPERIMENTAL EVALUATION

We test the robustness of our data embedding scheme using a 20-page document. The same document is formatted using different font sizes and typefaces for testing. We used a HP Laserjet 5200 for printing the documents at the default resolution of 600 dots per inch (dpi). A Ricoh Aficio 2510 copier was used for photocopying and an Epson scanner was used to scan documents. The scanning resolution was chosen to be equal to the printing resolution of 600 dpi.

To embed a symbol, we either add or remove two groups of pixels in the shape of a 4×4 square along the edge of vertical strokes of length at least $l = 29$ pixels and width at least $w = 6$ pixels. These values were chosen experimentally to provide a good tradeoff between robustness and imperceptibility. Fig. 1(a) and (b) show the symbols used to embed bits 0 and 1 respectively. Fig 1(c) shows the synchronization symbol and Fig. 1(d) shows the *Begin Packet* symbol used in our experiments. 20 bytes of plaintext data are embedded in approximately one half of a page full of text. A Reed-Solomon code over $GF(256)$ with rate 1/4 is then applied, so that there are now 80 bytes to be embedded. These are assembled into 20 packets, each containing $N = 4$ bytes each. The width of



(a) Original text with no data (b) Text with embedded data (c) Printed text

(d) One round of copying (e) Two rounds of copying (f) True size text with embedded data

Fig. 4. (a) 12 point TNR font with no embedded data at 150% magnification. (b)-(e)Text with embedded data at 150% magnification, after degradation by various print, scan and photocopy operations. (f) Text with embedded data at true size.

the packet number field (PCK_NUM) is set to 5 bits, so that a maximum of 32 packets can be indexed. Additionally, there are 5 synchronization symbols, so that the number of symbols embedded in half a page is $20 \times (4 \times 8 + 5 + 5) = 840$.

Fig. 4(a)-(e) show a magnified portion of the text to highlight the details preserved after various operations. Fig. 4(a) shows the original unmarked text, Fig. 4(b) shows the text with embedded data, and Fig. 4(c)-(e) show the scanned document after printing, one round of photocopying and two rounds of photocopying, respectively. Fig. 4(f) shows a portion of a text document with embedded data at normal resolution. We observe that at normal reading distances, the modifications made are nearly imperceptible. Further, as the number of copying operations increases, the visual quality of the document becomes worse, and the embedded symbols undergo significant distortion.

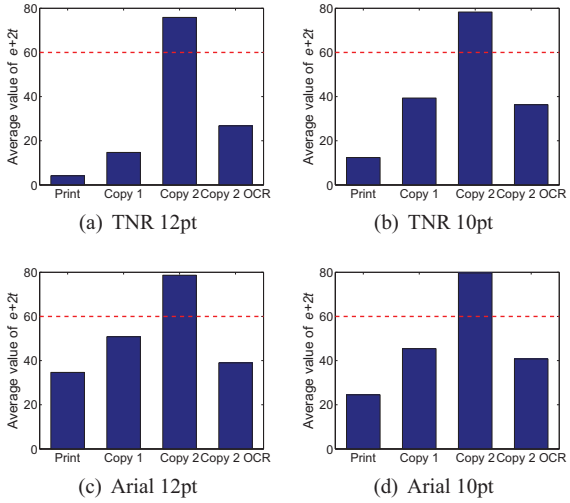
3.1. Detection results

A simple correlation-based detector is used to identify the embedded symbols as described in Sec. 2.1. After the symbols are extracted, the synchronization symbols are used to extract the individual bytes. Note that if one synchronization symbol or *Begin Packet* symbol is lost, then the entire packet is discarded, resulting in 4 erasures. When all the extracted symbols have been processed, a Reed-Solomon decoder is applied to correct the errors and erasures and extract the embedded message.

Fig 5 shows the average number of erasures (e) plus twice the number of errors (t) in extracting the embedded data after printing and photocopying operations. For the given choice of ECC parameters, the embedded data can be recovered as long as $e+2t \leq 60$. Thus, we can successfully extract the embedded data from the printed document and the first copy in most cases. However, the number of errors and erasures in the second copy is too large to be corrected by the error-correcting decoder and the data cannot be extracted. To address this problem, we use an optical character recognition (OCR) aided detector that can be used to extract data even from the second copy.

3.2. Optical Character Recognition (OCR) aided detector

To improve the accuracy of the detector, an OCR engine is used to first identify the characters, and then use this information to extract the hidden data accurately. It is assumed that the font typeface and size can be identified using font recognition systems such as [8]. We use the Tesseract OCR engine [9] to segment and identify the characters. Any errors in character recognition are manually corrected.

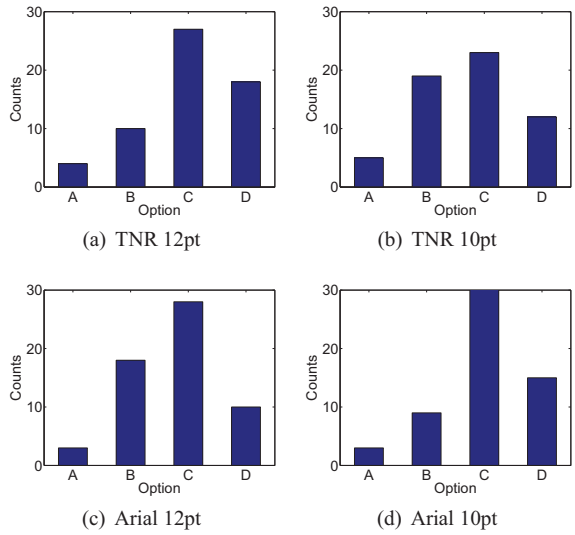


(a) TNR 12pt (b) TNR 10pt

(c) Arial 12pt (d) Arial 10pt

Fig. 5. Detection results for different fonts. The vertical bars represent the average value of $e + 2t$ for different operations. The dashed horizontal line represents the error correction capability of the Reed-Solomon code, i.e., $e + 2t \leq 60$.

The detector contains a database of commonly occurring font typefaces and sizes. Given the document with the embedded data, and the output of the OCR engine, the detector compares each character with the corresponding unmodified character from its database to determine the likely location of the hidden data. It then crops a small neighborhood of this location from the scanned image and correlates it with all the symbols to identify the most likely one. With this side information, the average number of errors and erasures in the second copy reduce by about $50 - 60\%$ to the values shown in Fig 5. Thus, OCR-aided detection enables extraction of the message from even the second photocopy.



(a) TNR 12pt (b) TNR 10pt

(c) Arial 12pt (d) Arial 10pt

Fig. 6. Histograms of qualitative scores.

3.3. Subjective evaluation of document quality

We also conducted subjective tests to determine the quality of the documents with embedded data. Four pages containing the same text formatted using two fonts (Arial and Times New Roman (TNR)), in two different sizes (10 pt and 12 pt) were used for embedding random data. To evaluate the worst case quality of the watermarked documents, data was embedded into every possible vertical stroke of length at least 29 pixels and width 6 pixels, representing the highest possible embedding rate and worst possible degradation due to embedding. For each page containing embedded data, another page was created with the same font and same size, but without embedded data. Thus there were 8 pages in all, 4 of which contained embedded data, while the other 4 did not. These 8 pages were then printed on a HP Laserjet 5200 printer at a resolution of 600 dpi.

Participants were asked to compare two pages in the same font and the same size with and without embedded data and choose one of the following options for each pair:

A. I don't see any difference between the two.
B. Barely perceptible differences.
C. Perceptible but not obtrusive / irritating.
D. Very obtrusive and irritating.

Fig. 6 shows histograms of the rating received for each pair of watermarked and unwatermarked documents. We observe that a larger number of people found the modifications obtrusive in TNR 12pt font and Arial 10pt font when compared to the other two. Overall, more than 75% of the users rated A, B, or C when asked to compare the same text with and without embedded data.

We again reiterate that the scores obtained above are indicative of a worst case scenario where *every possible* vertical stroke was used for embedding. In practice, only a fraction of these strokes would be chosen for embedding based on a key, as shown in Fig. 3(a). To further improve the perceptual quality, some of the symbols may be embedded on the right edge of these strokes and at different vertical positions, so that the distortion introduced is more random.

4. SUMMARY AND OUTLOOK

A method to embed data into hard-copy documents by making small changes to certain text characters was described. Using a simple correlation-based detector in conjunction with an error correcting code, the embedded data can be extracted from a photocopy of the original printed document. Using an OCR-aided detector, the detection accuracy can be further improved, and the hidden data can be extracted even from a photocopy of the photocopy. Subjective tests show that at the worst perceptual quality corresponding to the highest embedding rate, a majority of the participants found the changes made to the documents perceptible, but not obtrusive. In the future, we plan to make the embedding scheme robust to arbitrary cropping (tearing) attacks and to extend it to stroke-based fonts.

5. REFERENCES

[1] T. Sugai, U. Shimmyo, H. Ito, and M. Suzuki, "Development of a watermarking method for printed documents," in *70th Convention of the Information Processing Society of Japan*, Tsukuba, Japan, Mar. 2008, (in Japanese).

[2] T. Anan, K. Kuraki, and S. Nakagata, "Watermarking technologies for security-enhanced printed documents," *Fujitsu Scientific and Technical Journal*, vol. 4, no. 2, pp. 197–203, Apr. 2007.

[3] J. T. Brassil, S. Low, and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," *Proc. of the IEEE*, , no. 7, pp. 1181–1196, Jul. 1999.

[4] D. Huang and H. Yan, "Interword distance changes represented by sine waves for watermarking text images," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1237–1245, Dec. 2001.

[5] R. Villán, S. Voloshynovskiy, O. Koval, J.E. Vila-Forcén, E. Topak, F. Deguillaume, Y. Rytsar, and T. Pun, "Text data-hiding for digital and printed documents: Theoretical and practical considerations," in *Proc. of SPIE-IS&T Electronic Imaging 2006, Security, Steganography, and Watermarking of Multimedia Contents VIII*, San Jose, USA, January 15–19 2006.

[6] H. Y. Kim and J. Mayer, "Data hiding for binary documents robust to print-scan, photocopy and geometric distortions," in *Proc. of the XX Brazilian Symposium on Computer Graphics and Image Processing*, Washington, DC, USA, 2007, pp. 105–112.

[7] I. Cox, M. Miller, and A. McKellips, "Watermarking as communications with side information," *Proc. of the IEEE*, vol. 87, no. 7, pp. 1127–1141, Jul. 1999.

[8] A. Zramdini and R. Ingold, "Optical Font Recognition using Typographical Features," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 877–882, Aug. 1998.

[9] "Tesseract OCR engine," Available online at http://code.google.com/p/tesseract-ocr/.