An improvement of a single-dot method for an information-hiding method by applying an error-correcting code

Hiroyasu KITAZAWA, Kitahiro KANEDA,
Keiichi IWAMURA
Department of Electrical Engineering, Tokyo University of Science,
6-3-1 Ni-jiku Katsusika-ku Tokyo Japan,
kitazawa@sec.ee.kagu.tus.ac.jp

ABSTRACT

In this study, we focus on a technique that embeds an electronically readable watermark directly on printed material. This technique is used in the single-dot method proposed by Kaneda et al.
We improve upon this single-dot method proposed by Kaneda. In the existing method, the paper used was required to be card-sized, and the extraction rate was 97.68%. In this paper, we increase the paper size to A4 and raise the extraction rate to 98.72%.
We also experiment on paper-based documents with a foreground and succeed in embedding information and extracting it without error by using error-correcting codes with less redundancy.

KEYWORDS

Information hiding, Watermark, Background pattern, Error correcting code, LDPC code, Printed -documents

1 INTRODUCTION

Paper remains the most popular communication media even in today's technology-centric society. In recent years, printers have been able to produce high-quality characters at increasingly high resolutions (e.g., 1,200 dpi or 2,400 dpi). As a result, tampering with documents and unauthorized reproductions have become easier. In fact, most unauthorized duplications occurred with printed materials. In 2012, 715 instances of unauthorized duplications accounted for 74.9% of the total number in that year; in 2011, 1,066 instances accounted for 68.7% of the total from reference [1]. As a result, technology for embedding and extracting information in print has grown in importance as a deterrent. However, copy-protection technologies for printed materials have been studied less than equivalent techniques for electronic media. Recent print-based copy-protection techniques have been based on modulating the properties of individual text characters. Such methods include the color quantization of individual characters [2–6], which requires that the text be printed in color and be half-toned, the perturbation of character outlines [7–9], and the selection of a specific area for embedding to allow the copy of a document to be distinguished from the original [10].

2 SINGLE-DOT METHOD (EXISTING SYSTEM)

2.1 Single-dot method summary

In order to avoid a drop in document quality, and to allow the extraction of embedded information without using complicated Gabor filter calculations, a watermarking method that makes use of a dot pattern was proposed by Kaneda et al [11]. The proposed method superimposes the hatching pattern that represents the information on the background of the document, and the dot pattern is produced by rearranging dots in the background tile pattern.

2.2 Base pattern

In the single-dot method, a pattern with a one-pixel dot in the center of an 18×18-pixel tile is defined as “1” and a pattern with no dot in the center is defined as “0.”

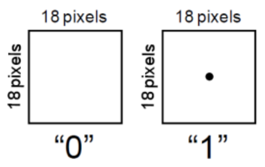


Fig. 1 Dot pattern showing instances of “1” and “0” when using the single-dot method

2.3 Detection method

A histogram of the luminosity value of the input image along the horizontal and vertical directions is respectively calculated in order to detect the presence of dots. If there is at least one dot along each direction, each histogram will have a peak. We define the *i*th vertical peak position and *j*th horizontal peak position of the histogram as tile coordinate_ij (pcxi, pcyj), and the corresponding area is defined as tile area_ij. When at least one pixel beyond a predetermined value exists in the tile area_ij, it is referred to as “1,” and when there is no pixel, it is referred to as “0.” We show this detection principle in Figure 2.

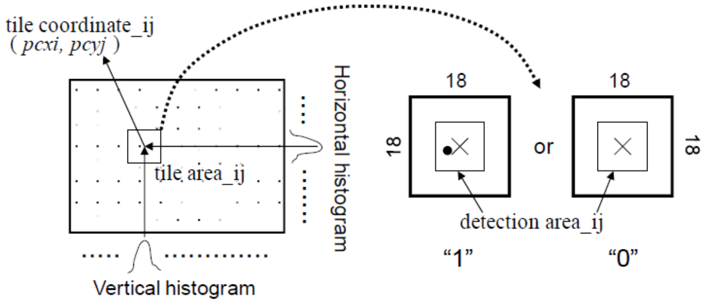


Fig. 2 Illustration of the principle of extraction

2.4 Problems with the single-dot method

When expanding the size of paper used from card size to A4, the following problems occur.
(1) Deviation of the paper
(2) Displacement of the dot pitch
(3) Omission of the dot
We show the effect of (1) the deviation of the overall paper in Figure 3, and (2) the displacement of the dot pitch in Figure 4. Regarding (3), when the output from a bmp file rather than a printer, we are able to detect the dot without missing. As such, we have determined that there is a problem due to the hardware of the printer. We suggest solutions to these problems in the next section.

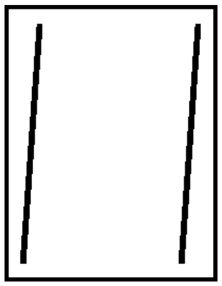


Fig. 3 Slope of the direction

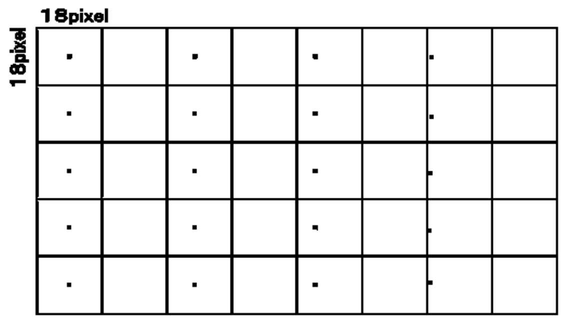


Fig. 4 Displacement of the dot pitch

3 SINGLE-DOT METHOD (PROPOSED SYSTEM)

3.1 Information-embedding method

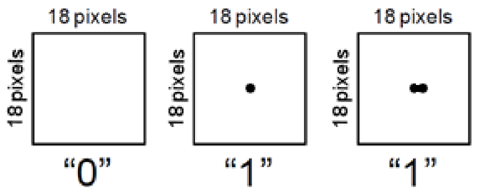


Fig. 5 Comparison with the two-point method

In order to solve (3), the problem of the omitted dot, we propose a two-points-out method. With the two-points-out method, a pattern with a two-pixel dot in the center is defined as “1”; otherwise, the pattern is defined as “0.” Figure 5 shows the patterns for “0” and “1” using the one-point-out method and the two-points-out method.

3.2 Detection method

Prior to detection, the “1” pattern must be tiled as a marker in the longitudinal and transverse directions of the sheet, as shown in Figure 6. Figure 7 depicts an enlarged diagram of the marker in the lower left part of Figure 6When detected, to calculate the density histogram horizontal and vertical direction. Read information by examining whether there are points in the intersection of the peak position in each direction and its surrounding. A simple marker is better, because it must be provided separate from the information that is to be embedded. However, because the grid is based on this marker, it must be possible to extract the marker reliably. For our method, we use four blocks, which is appropriate given the above reasons.

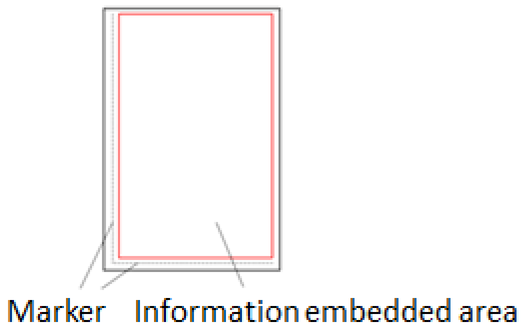


Fig. 6 Detection method

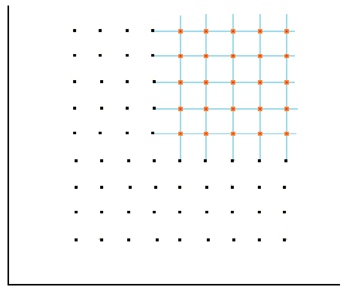


Fig. 7 Detection method (extension)

However, using only the grid as described above, the dots may not be extracted correctly, because an unexpected inclination could occur. The paper could be tilted substantially to the left or right, as shown in Figure 3, either because the shape of the sheet is not rectangular or is distorted slightly, or because the roller in the printer did not uniformly apply force to the paper when moving it.

In order to detect the inclination, we determine the coordinates of the dots in the upper and lower left corners to obtain the inclination of the left side. By applying the same inclination to the right side, the whole grid is deformed into a parallelogram.

3.3 Character cut

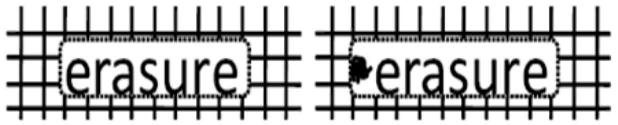
The results of the error correction are different, depending on whether the characters in the printed material are included or not. In this study, we used a “delimiter” to remove the characters before error correction.

We define “character cut” as the process of separating characters and document images using an optical character reader (OCR). The decision whether to cut a character is considered on a case-by-case basis. Characters can be separated individually or as a group. When a character cut is required, the dot pattern is laid out only around the characters during embedding, as shown in Figure 8(a). Similarly, the dot pattern is read without the characters during extraction.

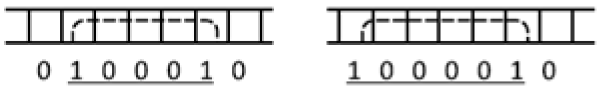
Deviations could occur when cutting the character out during embedding and during extraction, because of dirt and/or staining on the paper, as well as artefacts from printing or scanning, any of which could be erroneously identified as a character, as shown in Figure 8(b). If the encoded word is processed with error-correction codes, the shift in character-cutting could change the code length. If the shift is not corrected through calibration, error correction becomes inaccurate.

An effective way to calibrate embedded dot patterns has not yet been established; therefore, we propose a calibration mechanism using a dummy bit to store calibration information in significant bits. A string of 100…001 bits is used as the dummy bit. If the character-cutting range while embedding is different from that while extracting, the code can be aligned, as shown in Figure 9 A “1” tile is inserted before and after the dummy bit to more easily distinguish the information to be embedded from the dummy bit during extraction.

If the approximate location of the character can be determined during extraction, part of the information can be used instead of the dummy bit. However, in order to extract the error-correction code with the dummy bit, the code must be treated as information bits. The character cut deviates by only one tile at most, and the dummy bit would be lost only if it deviates by two bits. Therefore, the dummy bit can become a calibration guide, similar to the blank character, which occurs primarily in asymmetrical inversion errors and which is used to improve the correction efficiency. Thus, the channel model becomes an asymmetric inverted error-symmetric erasure error channel. If the dot pattern of the original is not dependent on whether the dummy bit is a “1” or a “0,” the loss error is symmetrical.



(a) When embedding (b) During extraction
Fig. 8 Character cutting



(a) When embedding (b) During extraction

Fig. 9 Dummy bit

4 EXPERIMENT

4.1 Experimental purposes

With the combination of the embedded method that was introduced and character-cut, we verify which combinations are to be extracted without error from the embedded information.

We use low density parity checking (LDPC) code as the error-correcting code. LDPC code is a linear code defined by a sparse matrix consisting of 1s and 0s, proposed by Gallager in the 1960s [12]. The LDPC code is longer, but it performs better. Because the single-dot method has a large capacity for embedded information, it can store a longer code and therefore perform better.

4.2 Dot Extraction

We verify which combinations are to be extracted without error from the embedded information.

For the proposed single-dot method, we selected one option from each of the following conditions (A)–(C) to generate eight scenarios.

(A) Pattern size (18×18 or 9×9)

(B) One-point-out method or two-points-out method

(C) With or without character cut

4.3 Error-correction simulation

The error-correction simulation randomly generates a “1” and “0"” as the first information sequence, encodes the information sequence using the generator matrix, adds noise, and decodes the information using the parity-check matrix. The goal of the simulation is to determine the encoding rate. We embedded 18×18-pixel tiles on one side of an A4-sized sheet of paper. If one code is stored on one sheet, the code length becomes 106,975 bits. The long code length can be used to determine the performance of the LDPC code. However, this simulation is not possible, because the capacity of the parity-check matrix becomes too large. Instead, we used 10 code words of the embedded code to perform the simulation, resulting in a code length of 10,560 bits.

We used the block error rate to calculate the error rate when comparing the actual code with the results of decoding. In addition, we counted the error bits within the information portion only, and not in the dummy bit part. We used a logarithmic domain sum-product decoding method with a maximum iteration of 100 times. Without character cuts, the error rates are as shown in the “yes-character” columns of Tables 1 and 2. With character cuts, where the characters comprise 30% of the paper, the error rates are as shown in the “no-character” columns of Tables 1 and 2. We assume that the error rate of removing characters is 0.01%, although the error rate varies depending on the density of the paragraphs on the paper. The programs in the simulation were based on [13].

4.4 Equipment used

(A) Standard equipment (printer, scanner)

・Canon MP970

The resolution of both scanner and printer was set to 600 dpi.

(B) Paper media

・Black-and-white copy/printer paper (Fuji Xerox Co. Ltd.)

(C) Printed material used for the verification print with characters

・JEITA standard [standard pattern J1]

(D) Computer and software

・Intel(R) Xeon(R)CPU E3-1320 V2@3.30GHz

・Microsoft Visual C++ 6.0, Microsoft Visual C++ 2010

4.5 Results

We show the error rate of the 18×18-pixel tiles, as well as the number of errors, in Table 1, for both the one-points-out and the two-points-out methods. The error rates are given for documents with a foreground, and for documents without a

foreground. Both the number of errors and the error rate are given for each case. An error is defined as being a “1” detected that was originally “0,” or a “0” that was originally a “1.”
The error rate used here is calculated by dividing the possible amount of embedding information by the amount of information that has been extracted by mistake.
Table 2 shows in the same information given in Table 1, but for the case of 9×9-pixel tiles.

Table 1. 18 × 18 test results

	18×18					
	no-character			yes-character		
	prob0to1	prob1to0	total error	prob0to1	prob1to0	total error
one dot	0.06%	11.78%	5.92%	7.42%	11.42%	9.42%
	31	5805	5836	3661	5634	9295
two dots	0.10%	2.46%	1.28%	3.86%	2.36%	3.11%
	53	1211	1264	1904	1164	3068

Table 2. 9 × 9 test results

	9×9					
	no-character			yes-character		
	prob0to1	prob1to0	total error	prob0to1	prob1to0	total error
one dot	0.12%	10.34%	5.23%	4.28%	7.62%	5.95%
	245	20598	20843	8511	15188	23699
two dots	0.32%	1.12%	0.72%	3.30%	0.58%	1.93%
	632	2249	2881	6554	1138	7692

The results of the error-correction simulation are given in Figures 10 and 11. The vertical axis is the block error rate, while the horizontal axis is the coding rate. The coding rate increases as it goes to the right on the horizontal axis, while the probability of a block error decreases as it continues downward on the vertical axis.

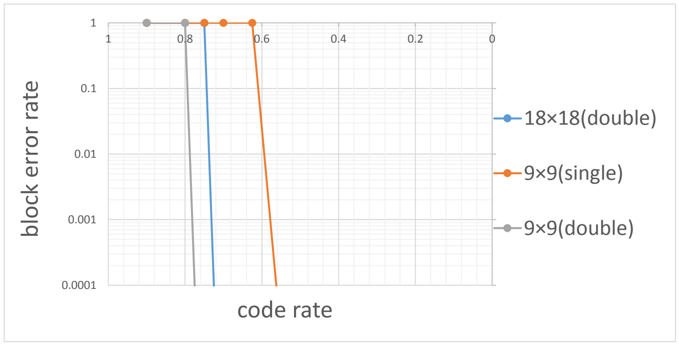


Fig. 10 Simulation results of LDPC (no character cut)

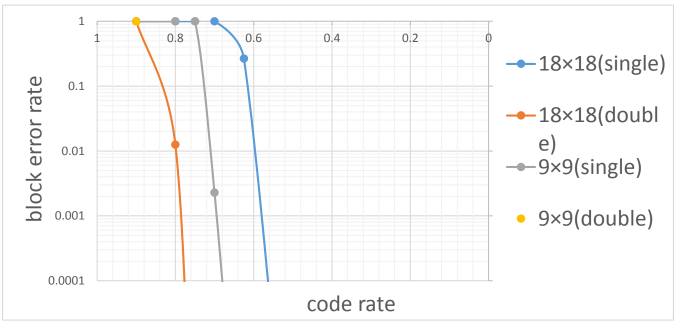


Fig. 11 Simulation results of LDPC (with character cut)

4.6 Consideration (before error correction)

With a block size of 18×18 pixels, we compared the two-points-out method and the one-point-out method with no characters. The number of errors when reading a “1” tile and then a “0” tile (1 → 0) was reduced from 5,805 with the one-point-out method down to 1,211 with the two-points-out method, because the two-pixel dot is easier to see than the one-pixel dot. The error rate when reading 1 → 0 in both methods is lower without characters than with characters, because the characters overlap exactly at a specific point on the extraction grid. On the other hand, the error rate when reading 0 → 1 increased in both methods, because the characters overlapped at the location of the original “0.”
We used a foreground pattern where characters comprise approximately 30% of the total area. If “1” tiles are embedded in advance on 30% of a grid, the probability of incorrectly reading 0 → 1 becomes 30%. However, in this experiment, the probability of error stayed at 3.86% for the two-points-out method and 7.42% for the one-point-out method. Characters are preferred to a set of dots. For example, it is more difficult to find a one–pixel dot 100 times than it is to find a single 100–pixel dot. If the dot has two pixels, the number of tiles that overlapped differed greatly. We assume that the same principle influenced the results in this experiment.
The same general trend was found with both 9×9-pixel tiles and 18×18–pixel tiles. The difference between the two was in the errors when reading 1 → 0. By reducing the pattern size, the density of the dots is improved.

4.7 Consideration (after error correction, no character cut)

We show the simulation results in Figures 10 and 11. For each combination of pattern size and method, the most suitable code and amount of information is shown in Table 3.

Table 3. Most suitable code (no character cut)

	code rate	Amount of imformation
18×18(single)		
9×9(single)	0.5	211,200[bit]
18×18(double)	0.7	73,929[bit]
9×9(double)	0.75	316,800[bit]

4.8 Consideration (after error correction, with character cut)

When characters are excluded, the percentage of cut characters was 30%. That is, with 18×18-pixel tiles at 100% extraction, we can extract 73,920 bits of information without error. In this simulation, the extraction rates of the dots were calculated using the results from Tables 1 and 2. For each combination of pattern size and method, the most suitable code and amount of information is shown in Table 4. When characters are included, the two-points-out method is also able to extract much more information without error than the one-point-out method. By excluding characters, we increase the coding rate and reduce the number of dots to be embedded. By including characters, much more information can be extracted without error.

Table 4. Most suitable code (with character cut)

	code rate	Amount of imformation
18×18(single)	0.5	36,960[bit]
9×9(single)	0.5	147,840[bit]
18×18(double)	0.75	55,400[bit]
9×9(double)	0.8	236,544[bit]

5 SUMMARY

In this paper, we demonstrated that, compared with the conventional method, single-dot printing can embed and extract more information directly on the printed matter without additional visual discomfort by rearranging the watermark dots and applying error correction using LDPC codes. Further, it can maintain a high extraction rate even with materials that have preprinted characters. The best combination, which uses a regular LDPC code of 0.75 with a pattern size of 9×9 pixels and no characters cut, can extract 316,800 bits of information without error. Because the single-dot method can embed a large amount of information, the code length could be increased and the performance of the LDPC code could be improved.

For future work, the challenge is to improve tamper resistance. In this study, we assumed that errors are caused by naturally occurring dirt; however, we must consider the possibility that the paper is maliciously tampered. Graffiti or mutilation of the paper will increase errors. For example, if the paper is cut, a short code embedded multiple times is preferable to a long code embedded once.

It should be noted that the proposed method is also applicable to augmented reality technologies. For example, by viewing the layout of a building with a phone camera, the embedded information could display a stereoscopic view of the building on the phone.

6 REFERENCES

[1] NPO, Japan Network Security Association, “2012, Survey report on information security incidents～ Leakage of personal information～”

[2] R Villán, et al. "Text data-hiding for digital and printed documents: theoretical and practical considerations." *Electronic Imaging 2006*. International Society for Optics and Photonics, 2006.

[3] R Villán, et al. "Tamper-proofing of electronic and printed text documents via robust hashing and data-hiding." *Electronic Imaging 2007*. International Society for Optics and Photonics, 2007.

[4] P Borges, J Mayer, and E Izquierdo, “Robust and Transparent Color Modulation for Text Data Hiding.” IEEE International Conference on Image Processing (ICIP), vol. 10. No. 8. 12791489. December 2008.

[5] AK Bhattacharjya, and H Ancin. "Data embedding in text for a copier system." *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*. Vol. 2. IEEE, 1999.

[6] IV Kurilin, et al. “Embedding positional-independent hidden data into hardcopy.” *Pattern Recognition and Image Analysis* 21.3 (2011): 511-514.

[7] S Das, S Rane, and A Vetro. “Hiding information inside structured shapes.” *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International*

Conference on. IEEE, 2010.

[8] QG Mei, EK Wong, and ND Memon. “Data hiding in binary text documents.” *Photonics West 2001-Electronic Imaging*. International Society for Optics and Photonics, 2001.

[9] ML Bensaad and MB Yagoubi. “High capacity diacritics-based method for information hiding in Arabic text.” *Innovations in Information Technology (IIT), 2011 International Conference on*. IEEE, 2011.

[10] D Elliman, P Blanchfield, and A Albakaa. “Integrity Check for Printed Binary Document Images.” *Networked Digital Technologies*. Springer Berlin Heidelberg, 2010. 523-532.

[11] K Kaneda, et al. “A study of information hiding performance using simple dot pattern with different tile sizes.” *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on*. IEEE, 2008.

[12] R Gallager. “Low-density parity-check codes.” *Information Theory, IRE Transactions on* 8.1 (1962): 21-28.

[13] T Wadayama, “Public program about LDPC code relationship” http://www-tkm.ics.nitech.ac.jp/~