Data Hiding in Binary Image for Authentication and Annotation

Min Wu, *Member, IEEE,* and Bede Liu, *Fellow, IEEE*

***Abstract*—This paper proposes a new method to embed data in binary images, including scanned text, figures, and signatures. The method manipulates "flippable" pixels to enforce specific block-based relationship in order to embed a significant amount of data without causing noticeable artifacts. Shuffling is applied before embedding to equalize the uneven embedding capacity from region to region. The hidden data can be extracted without using the original image, and can also be accurately extracted after high quality printing and scanning with the help of a few registration marks. The proposed data embedding method can be used to detect unauthorized use of a digitized signature, and annotate or authenticate binary documents. The paper also presents analysis and discussions on robustness and security issues.**

***Index Terms*—Annotation, authentication, binary image, data hiding, digital watermarking.**

I. Introduction

DIGITAL watermarking and data hiding techniques have been proposed for a variety of digital media applications, including ownership protection, copy control, annotation, and authentication. Most prior works on image data hiding are for color and grayscale images in which the pixels take on a wide range of values [2]–[9]. For most pixels, changing the pixel values by a small amount may not be noticeable under normal viewing conditions. This property of human visual system plays a key role in watermarking of perceptual data [5], [6]. For images in which the pixels take value from only a few possibilities, hiding data without causing visible artifacts becomes more difficult. In particular, flipping white or black pixels that are not on the boundary is likely to introduce visible artifacts in binary images.

Hiding data in binary image, though difficult, is getting higher demands from our everyday life. An increasingly large number of digital binary images have been used in business. Handwritten signatures captured by electronic signing pads are digitally stored and used as records for credit card payment by many department stores and for parcel delivery by major courier services such as the United Parcel Service (UPS). Word processing software like Microsoft Word allows a user to store his/her signature in a binary image file for inclusion at specified locations of an electronic document. The documents signed in such a way can be sent directly to a fax machine or be distributed across a network. The unauthorized use of a signature, such as copying it onto an unauthorized payment, is becoming a serious concern. In addition, a variety of important documents, such as social security records, insurance information, and financial documents, have also been digitized and stored. Because of the ease to copy and edit digital images, annotation and authentication of binary images as well as detection of tampering are very important.

This paper discusses data hiding techniques for these authentication and annotation purposes, possibly as an alternative to or in conjunction with the cryptographic authentication approach. Such targeted applications calls for fragile or semifragile embedding of many bits. It should be stressed that while it is desirable for the embedded data to have some robustness against minor distortion and preferably to withstand printing and scanning, the robustness of embedded data against intentional removal or other obliteration is not a primary concern. This is because an adversary in authentication applications would have much more incentive to counterfeit valid embedded data than to remove them, and there is no obvious threat of removing embedded data in many annotation applications.

Several methods for hiding data in specific types of binary images have been proposed in literature. Matsui *et al.* [10] embedded information in dithered images by manipulating the dithering patterns and in fax images by manipulating the run-lengths. Maxemchuk *et al.* [11] changed line spacing and character spacing to embed information in textual images for bulk electronic publications. These approaches cannot be easily extended to other binary images and the amount of data that can be hidden is limited. In [12], Koch and Zhao proposed a data hiding algorithm which enforces the ratio of black versus white pixels in a block to be larger or smaller than 1. Although the algorithm aims at robustly hiding information in binary image, it is vulnerable to many distortions/attacks, and it is not secure enough to be directly applied for authentication or other fragile use. The number of bits that can be embedded is limited because the particular enforcing approach has difficulty in dealing with blocks that have low or high percentage of black pixels. However, the idea of enforcing properties of a group of pixels via the local manipulation of a small number of pixels can be extended as a general framework of data embedding. Another approach of marking a binary document is proposed in [13] by treating a binary image as a grayscale one and manipulating the luminance of dark pixels slightly so that

Manuscript received April 1, 2001; revised March 28, 2002. This work was supported in part by a New Jersey State R&D Excellence Award, NSF Grants MIP-9408462 and CCR-0133704, and an Intel Technology for Education 2000 Grant. Part of this work was presented at the IEEE International Conference on Multimedia & Expo (ICME'00) [1]. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sankar Basu.

M. Wu was with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA. She is now with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: minwu@eng.umd.edu).

B. Liu is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: liu@princeton.edu).

Digital Object Identifier 10.1109/TMM.2004.830814

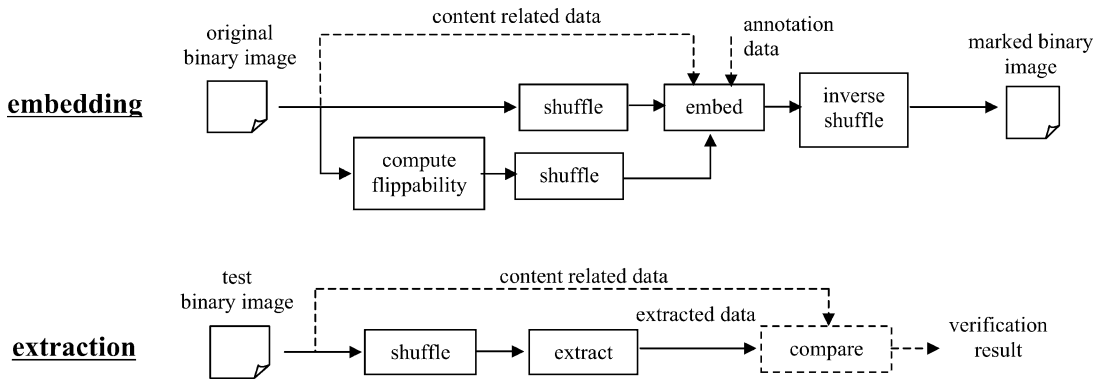


Fig. 1. Block diagram of the embedding and extraction process in binary images for authentication and/or annotation.

the change is imperceptible to human eyes yet detectable by scanners. This approach, targeted at intelligent copier systems, is not applicable to bilevel images hence is beyond the scope of this paper. The bilevel constraint also limits the extension of many approaches proposed for grayscale or color images to binary images. For example, applying the spread spectrum embedding, a transform-domain additive approach proposed by Cox *et al.* [5], to binary image would not only cause annoying noise on the black–white boundaries, but also have reduced robustness hence limited embedding capacity due to the post-embedding binarization that ensures the marked image is still a bilevel one [14]. For these additive embeddings, hiding a large amount of data and detecting without the original binary image is particularly difficult. In summary, these previously proposed approaches either cannot be easily extended to other binary images, or can only embed a small amount of data.

We propose a new approach that can hide a moderate amount of data in general binary images, including scanned text, figures, and signatures. The hidden data can be extracted without using the original unmarked image, and can also be extracted after high quality printing and scanning with the help of a few registration marks. The approach can be used to verify whether a binary document has been tampered with or not, and to hide annotation labels or other side information.

The paper is organized as follows. The proposed approach is presented in Section II and illustrated using three applications in Section III. Further discussions on robustness and security are given in Section IV, including such issues as recovering hidden data from high quality printing-and-scanning. Section V concludes the paper and suggests a few directions of future work.

II. Proposed Method

There are two basic ways to manipulate binary images for hiding data. The first class of approaches changes low-level features such as flipping a black pixel to white or vice versa. The second class of approaches changes high-level features such as modifying the thickness of strokes, curvature, spacing, and relative positions. Since the number of parameters that can be changed by the second class of approaches is limited, especially under the requirements of invisibility and blind detection (i.e., without using the original image in detection), the amount of data that can be hidden is usually limited except for special types of images [11].

We focus in this paper on the first class of approaches. An image is partitioned into blocks and a fixed number of bits are embedded in each block by changing some pixels in that block. For simplicity, we shall show how to embed one bit in each block. Three issues will be discussed: 1) how to select pixels for modification so as to introduce as little visual artifacts as possible, 2) how to embed data in each block using these flippable pixels, and 3) why to embed the same number of bits in each block and how to enhance the efficiency. The entire process of embedding and extraction is illustrated in Fig. 1.

A. Flippable Pixels

There is little discussion in the literature on a human visual model for binary images. A simple criterion, proposed in [12], is to only flip boundary pixels for high contrast image such as text image, and to only introduce new pixel patterns that are rather isolated for dithered image. We take the human perceptual factor into account by studying each pixel and its immediate neighbors to establish a score of how unnoticeable a change on that pixel will cause. The score is between 0 and 1, with 0 indicating no flipping. Flipping pixels with higher scores generally introduces less artifacts than flipping a lower one.

To assign flippability score manually according only to neighborhood patterns has the shortcomings that the storage of every pattern can be huge (except for small neighborhood) and that such a fixed assignment does not offer flexibility for binary images with different characteristics. Our approach in this paper is to determine the scores dynamically by observing the smoothness and connectivity. The smoothness is measured by the horizontal, vertical, and diagonal transitions in a local window (e.g., 3×3), and the connectivity is measured by the number of the black and white clusters. For example, the flipping of the center pixel in Fig. 2(b) is more noticeable than that in Fig. 2(a). We order all 3×3 patterns in terms of how unnoticeable the change of the center pixel will be. We then examine a larger neighborhood, say 5×5, to refine the score. Special cases are also handled in larger neighborhood so as to avoid introducing noise on special patterns such as sharp corners. By changing the parameters in our procedure, we can easily adjust the intrusiveness of different kind of artifacts and tailor to different types of binary images. Details of our approach are given in the Appendix.

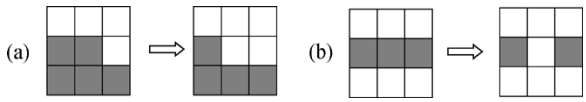


Fig. 2. Two examples of 3 × 3 neighborhood, for which flipping the center pixel to white in (a) is less noticeable than that in (b).

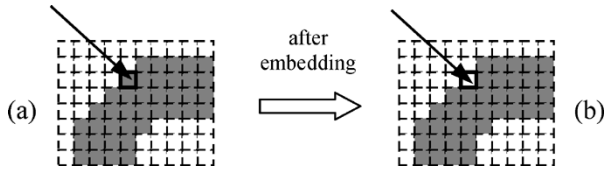


Fig. 3. Example illustrating a boundary pixel could become "nonflippable" after flipping.

B. Embedding Mechanism

Directly encoding the hidden information in flippable pixels (e.g., set to black if to embed a "0" and to white if to embed a "1") may not allow the extraction of embedded data without the original image. The reason is that the embedding process may change a flippable pixel in the original image to a pixel that may no longer be considered flippable. As a simple example, suppose only black pixels that are immediately adjacent to white pixels are considered as "flippable". One such flippable pixel, marked by thick boundary in Fig. 3(a), is changed to white to carry a "1", as shown in Fig. 3(b). It can be seen that after embedding, this pixel is no longer considered flippable if applying the same rule. This simple example shows the difficulty for the detector to correctly identify which pixel carries hidden information without using the original image.

Instead of encoding the hidden information directly in flippable pixels, we embed data by manipulating pixels with high flippability scores to enforce a certain relationship on low-level features of a group of pixels. One possibility is to use the total number of black pixels in a block as a feature. To embed a "0" in a block, some pixels are changed so that the total number of black pixels in that block is an even number. Similarly, to embed a "1", the number of black pixels is enforced to an odd number. We may also choose a "quantization" step size Q and force the total number of black pixels in a block to be 2kQ for some integer k in order to embed a "0", and to be $(2\mathrm{k}+1)\mathrm{Q}$ to embed a "1". A larger Q gives higher tolerance to noise at the expense of decreased image quality. This "odd–even" embedding can be viewed as a special case of lookup table embedding similar to that in [8], [9]. These two approaches are illustrated in Fig. 4, where each possible quantized number of black pixels per block is mapped to 0 or 1. While other relationship enforcing techniques [15] are possible, we shall use in this paper the enforcing of odd or even number of black pixels in a block.

The above approaches can be characterized more generally by

$$v_i' = \arg \min_{x:T(x)=b_i, x=kQ} |x - v_i| \tag{1}$$

where v_i is the i^{th} original feature, v_i' is the feature value after embedding, b_i is the bit to be embedded in i^{th} feature, and

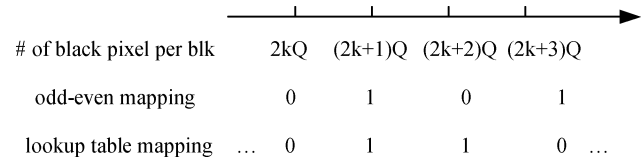


Fig. 4. Illustration of odd–even mapping and table lookup mapping.

$T(\cdot)$ is a prescribed mapping from feature values to hidden data values {0,1}. Detection is done by checking the enforced relationship:

$$\hat{b}_i = T(v_i''), \tag{2}$$

where v_i'' is the feature extracted from the i^{th} block of a test image, and $\hat{b}_i$ is the estimated value of the embedded bit in the i^{th} block.

If the same bit is repeatedly embedded in more than one block, majority voting is performed to determine which bit has been hidden. More sophisticated coding than simple repetition can also be used to achieve a certain level of tradeoff between robustness and capacity.

C. Uneven Embedding Capacity and Shuffling

As outlined earlier, we embed multiple bits by dividing an image into blocks and hiding one bit in each block via the enforcement of odd–even relationship. However, the distribution of flippable pixels may vary dramatically from block to block. For example, no data can be embedded in the uniformly white or black regions, while regions with text and drawing may have quite a few flippable pixels, especially on the nonsmooth boundary. This uneven embedding capacity can be seen from Fig. 5 where the pixels with high flippability scores, indicated by black dots in Fig. 5(b), are on the rugged boundaries.

General approaches to handling uneven embedding capacity have been discussed in [15], [16]. Regarding the uneven embedding capacity in a binary image, using variable embedding rate from block to block is not feasible for the following reasons. First, a detector has to know exactly how many bits are embedded in each block. Any mistake in estimating the number of embedded bits is likely to cause errors in decoding the hidden data for the current block, and the error can propagate to the following blocks. Second, the overhead for conveying this side information via embedding is significant and could be even larger than the actual number of bits that can be hidden. We therefore adopt constant embedding rate to embed the same number of bits in each region and use shuffling to equalize the uneven embedding capacity from region to region [15], [17].

Fig. 6 shows the flippable pixels before and after a random permutation of all pixels, and Fig. 7 shows the histogram of the number of flippable pixels per 16 × 16-pixel block. It is seen that the distribution before shuffling extends from zero to 40 flippables per block and that about 20% of the blocks do not have any flippable pixels. The distribution after shuffling, shown in the dotted line, concentrates in the range of ten to 20, and all shuffled blocks have flippable pixels. Let m_r be the number of blocks each having exactly r flippable pixels $(r = 0, \ldots, q)$, S the total number of pixels, q the block size, N the number of

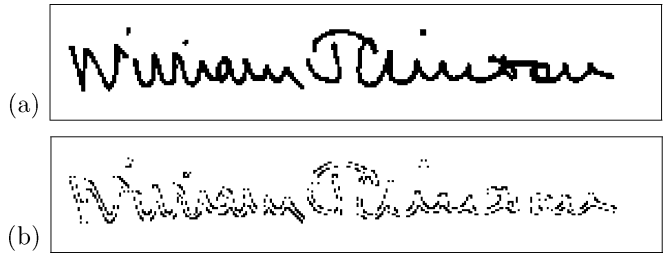


Fig. 5. (a) A binary signature image and (b) its pixels with high flippability scores shown in black.

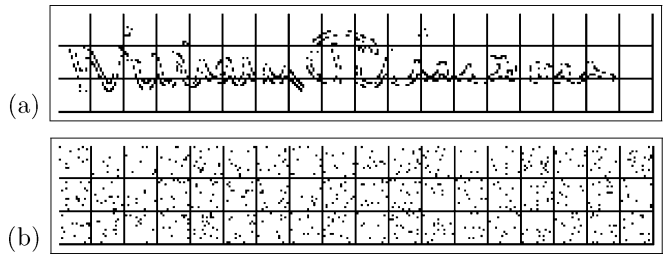


Fig. 6. Distributions of flippable pixels per 16 × 16-pixel block of the binary image in Fig. 5: (a) before shuffling and (b) after shuffling.

blocks, and p the percentage of flippable pixels.[1] It can then be shown [15], [16] that the mean and variance of each normalized bin m_r/N of the histogram as

$$E\left[\frac{m_r}{N}\right] = \frac{\binom{q}{r}\binom{S-q}{n-r}}{\binom{S}{n}}, \tag{3}$$

$$Var\left[\frac{m_r}{N}\right] = \frac{1}{N}\cdot\frac{\binom{q}{r}\binom{S-q}{n-r}}{\binom{S}{n}} + \left(1-\frac{1}{N}\right)\frac{\binom{q}{r}\binom{q}{r}\binom{S-2q}{n-2r}}{\binom{S}{n}} - \left[\frac{\binom{q}{r}\binom{S-q}{n-r}}{\binom{S}{n}}\right]^2. \tag{4}$$

For the signature image of ig. 5, we have

$$\begin{cases} \text{block size} & q = 16\times 16, \\ \text{image size} & S = 288\times 48, \\ \text{block number} & N = \frac{S}{q} = 18\times 3, \\ \text{flippable percentage} & p = 5.45\%. \end{cases}$$

The analytic results are shown in Fig. 8, along with the simulation results from 1000 random shuffles. The statistics of blocks with no or few flippables are also shown in Table I. The analysis and simulation are seen to agree well, and the percentage of blocks with no or few flippables is extremely low. Error correction coding can be used to handle a very small number of blocks that have no flippable pixels. As illustrated by the block diagram in Fig. 1, the embedding of one bit per block described in Section II-B is performed in the shuffled domain, and inverse shuffling is performed to get a marked image.

We would like to point out that shuffling does not produce more flippable pixels. It dynamically assigns the flippable pixels in active regions and along rugged boundaries to carry more data than less active regions. This is done without the need of specifying much side information that is image dependent. Shuffling also enhances security since the receiver side needs the shuffling table or a key for generating the table to correctly extract the hidden data.

[1]In this analysis, we set a threshold of 0.1 on the flippability score and consider the pixels with score higher than this as flippable.

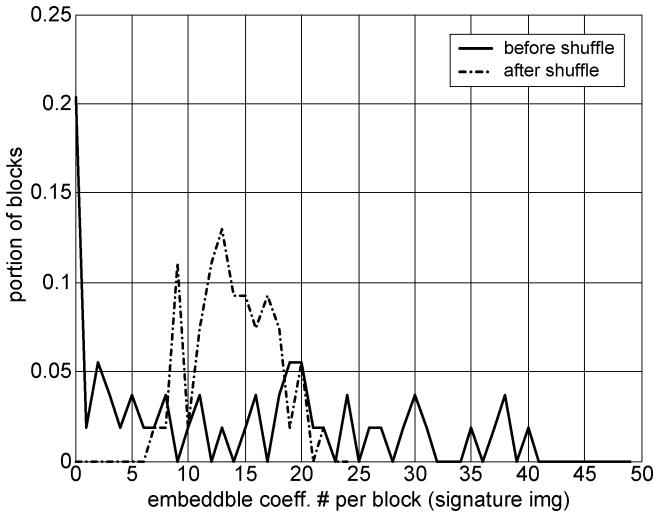


Fig. 7. Histogram of flippable pixels per 16 × 16-pixel block of the binary image in Fig. 5: (solid line) before shuffling and (dotted-dash line) after shuffling.

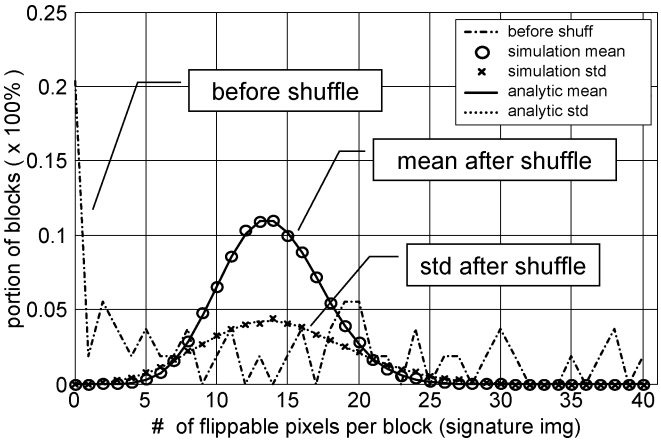


Fig. 8. Analysis and simulation of the statistical behavior of shuffling for the binary image in Fig. 5(a).

III. Applications and Experimental Results

The proposed data hiding for binary image is targeted mainly at authentication and annotation. In this section, we present three specific applications along with experimental results.

A. "Signature in Signature"

Using archived signatures for unintended purposes is a potential concern toward the increasingly popular deployment of digitized signature in our everyday life. A "Signature in Signature"[2] system annotates the signer's signature with the data that is related to the signed documents, so that the unauthorized use of a signature can be detected [18].

[2]Here the second "signature" refers to the actual digital version of a person's signature, while the first "signature" refers to a checksum related to the document content or other annotation information.

TABLE I
ANALYSIS AND SIMULATION OF THE BLOCKS WITH NO OR FEW FLIPPABLE PIXELS BEFORE AND AFTER SHUFFLING FOR THE BINARY IMAGE IN FIG. 5(a)

	before shuffle	mean after shuffle		std after shuffle	
		analysis	simulation	analysis	simulation
m_0/N (0^{th} bin)	20.37%	5.16×10^{-5}%	0 %	9.78×10^{-5}	0
m_1/N (1^{st} bin)	1.85%	7.77×10^{-4}%	0 %	3.79×10^{-4}	0
m_2/N (2^{nd} bin)	5.56%	5.81×10^{-3}%	5.56×10^{-3}%	0.0010	0.0010

Fig. 9. "Signature in Signature": (a) original image; (b) marked copy with seven letters (49 bits) embedded in; (c) difference between the original and the marked (shown in black).

The data hiding method proposed in this paper can be applied to annotating a signature in such applications as faxing signed documents and storing digitized signatures as transaction records. Compared with the traditional cryptographic authentication approach [19] that has been used in secure communication, the proposed data embedding based approach has the advantage of being user-friendly, easy to visualize, and integrating the authentication data with the signature image in a seamless way.

An example is demonstrated in Fig. 9, in which seven characters (49 bits) are embedded in a 287×61 signature. The embedding rate is one bit per block of 320 pixels. Fig. 9(a) is the original signature; Fig. 9(b) is the signature after embedding, which is indistinguishable from the original; and Fig. 9(c) shows where the altered pixels are.[3]

B. Invisible Annotation for Line Drawings

Artists may wish to annotate their drawings with information, such as the creation date and location, in such a way that the annotation data interfere minimally with perceptual appreciation. Our proposed data hiding approach can be used to invisibly annotate line drawings such as the 120×150 picture of Fig. 10. In this example, a character string of the date "01/01/2000" is embedded in Fig. 10(b).

C. Tamper Detection for Binary Document

A large number of important documents have been digitized and stored for records. The authentication of these digital documents as well as the detection of possible tampering are important concerns. The data hiding techniques proposed in this paper can be applied for these purposes, as an alternative to or in conjunction with the cryptographic authentication approach. More specifically, data are be embedded in an image in such a fragile way that it will be obliterated if the image is altered and/or that it no longer matches some properties of the image, indicating the tampering of content [16]. The hidden data can be an easily recognized pattern, or some features or their digest version related to the content of the host image.

Shown in Fig. 11(a) is a part of a U.S. patent document, consisting of 1000×1000 pixels. This binary image contains texts, drawings, lines, and bar codes. Fig. 11(b) is a visually identical figure, but with 976 bits embedded in it using the proposed techniques. In this particular example, 800 bits of the embedded data come from a "PUEE" pattern shown in Fig. 11(g). If the date "1998" on the top is changed to "1999", the extracted data will be the random pattern shown in Fig. 11(g), indicating that alteration was made.

IV. ROBUSTNESS AND SECURITY CONSIDERATIONS

In this section, we discuss a few robustness and security issues of the proposed scheme. Other considerations associated with shuffling, such as the methods for handling bad shuffles and for adaptively choosing the block size, can be found in [15], [16].

A. Analysis and Enhancement of Robustness

The robustness against noise of the embedding presented in Section II-B is quite limited, and generally depends on whether and how much quantization we applied [15], [16]. Let us consider the simple odd–even case with no quantization, i.e., the total number of black pixels is enforced to an even number to embed a "0", and to an odd number to embed a "1". When a single pixel is changed due to noise, the bit embedded in the block to which the pixel belongs will be decoded in error. When several pixels in an embedding block are subject to be changed, whether or not the bit can be decoded correctly depends on how many pixels are flipped; if the change is independent from pixel to pixel and is with probability p for each of n pixels where $n \geq 1$, the probability of getting a wrongly decoded bit is

$$P_{e_1} = \sum_{k=1, k \text{ odd}}^{n} \binom{n}{k} p^k (1-p)^{n-k} = \frac{1-(1-2p)^n}{2}. \quad (5)$$

The error probability P_{e_1} is small for small p and small n. In this case, error correction encoding can be applied to correct errors if accurate decoding of hidden data is preferred. When p is close to 0.5, so is P_{e_1}, implying the difficulty in embedding and extracting data reliably. Notice that because of shuffling, the assumption of independent change is likely to hold even if the noise involves nearby pixels. This is because adjacent pixels in the original image will be distributed to several blocks. If the

[3]The gray areas in Fig. 9(c), Fig. 10(c) and Fig. 11(d) visualize the strokes and the background, respectively. We show them in gray to assist viewers associating the difference between the original and the marked image with their precise location in the images. The pixelwise differences are indicated by black pixels.

(a) (b) (c)

Fig. 10. Invisible annotation for line drawings: (a) original image; (b) marked copy with ten-letter date information (70 bits) embedded in; (c) difference between the original and the marked (shown in black).

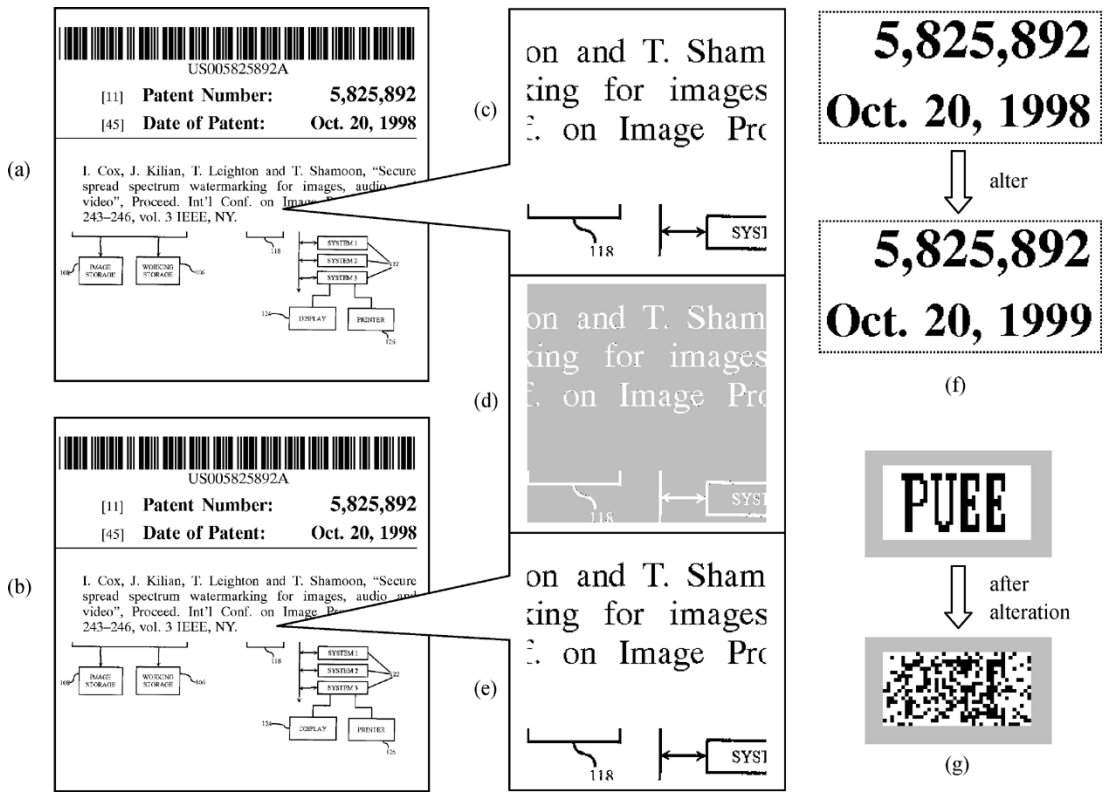


Fig. 11. Data hiding in binary document image: (a) original copy; (b) marked copy with 976-bit embedded in; (c) magnified original image; (d) difference between original and marked (shown in black); (e) magnified marked image; (f) portion of the image where alteration is done on the marked image by changing "1998" to "1999"; (g) among the 976-bit hidden data, 800 bits form a "PUEE" pattern; the 800-bit data pattern extracted after alteration is visually random and significantly different from the originally embedded "PUEE".

total number of changed pixels in the whole image is small (no matter whether they are close to each other in the original image or far away), it is likely that most of those pixels are involved in different embedding blocks hence the extracted bits from those blocks will be wrong. On the other hand, if many pixels have been changed, each embedding block may include several of these pixels and the decoded bit from each block is wrong with approximately 0.5 probability. This implies that the decoded data are rather random, as what we have seen in Fig. 11(g). The case of incorporating quantization can be analyzed similarly.

Besides the noise involving flipping of individual pixels, misalignment is another cause of decoding errors. For this matter, using shuffling has the disadvantage of increasing the sensitivity against geometric distortion such as translation. This is due to the shift-variant property of the shuffling operation, i.e., the shuffling result of a shifted image is very different from that of the nonshifted one. To alleviate the sensitivity with respect of translation, we can put the hidden data in a cropped part of the image, as shown in Fig. 12. Without loss of generality, we consider the case of black foreground and white background. The upper-left point of the data hiding region is determined by the uppermost and leftmost black pixel, and the lower-right point is by the lowermost and rightmost black pixel. The data hiding region therefore covers all black pixels. This approach can reduce the shifting sensitivity as long as both embedding and detection systems agree on the protocol and no cropping or addition of the outermost black pixels occurs.

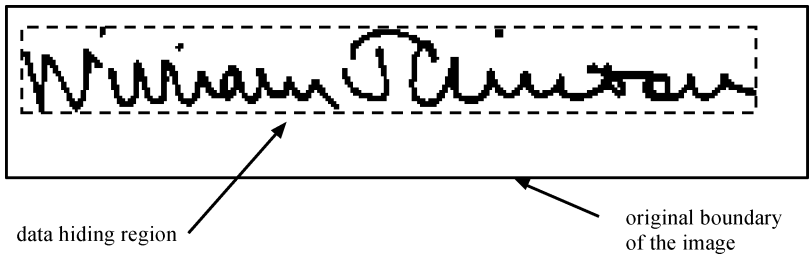


Fig. 12. Illustration of achieving robustness against small translation. Here we use the outermost black pixel to determine a data hiding region (indicated by a dash box) covering all black pixels.

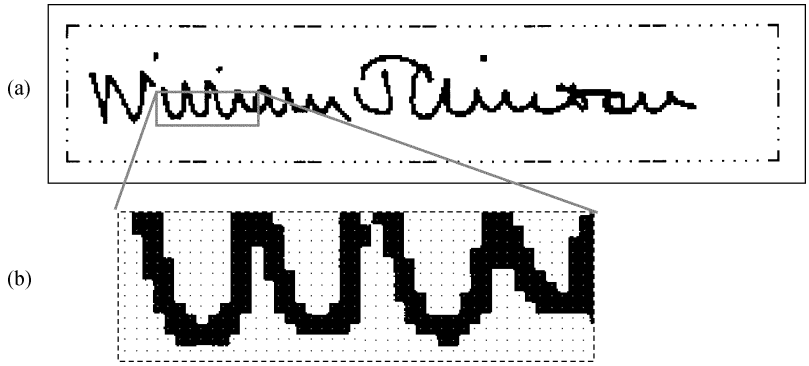


Fig. 13. Recovering binary image from high quality printing and scanning: (a) specially designed dotted signature box helps the recovery; (b) estimated centers of each original pixel are shown in light color in this zoomed-in view.

In addition to the above approach, adding registration marks such as a signature line helps survive high-resolution printing and scanning. In general, recovering image from printing and scanning with precision as high as one pixel is a nontrivial task, because this D/A-A/D process may result in small rotation, up-scaling of an unknown factor, and noisy boundary. If resolution-wise one pixel in the original image corresponds to only one pixel in the scanned version, it will be very difficult to combat the distortion introduced by the D/A-A/D process. On the other hand, if significant oversampling is performed so that one original pixel corresponds to a number of pixels in the scanned version, it would be possible to sample at the center of each "original" pixel, and avoid the noise introduced on the boundary and/or by the rounding errors in deskewing. Registration marks can help us identify the boundary and the size of the original image as well as correct skewing. We noted that while the size of one original pixel in terms of pixels in the scanned image may be estimated from a well-designed registration mark (e.g., we may estimate that one original pixel corresponds to 8×8 pixels in a scanned image), minor errors in such estimation could be accumulated when we determine the width and height of the original image up to single pixel precision. To overcome this problem, we impose constraints on the width and height of original signature images, for example, to be multiples of 50.

Fig. 13(a) shows one possible way to handle the recovery by adding a dotted signature box that resembles what has been commonly used in the current practice of signing. The four corner marks of the signature box and the dash line segments on the four sides at an interval of 50 pixels horizontally and of 25 pixels vertically serve as a ruler to facilitate the recovery. In this experiment, we printed out the signature image via

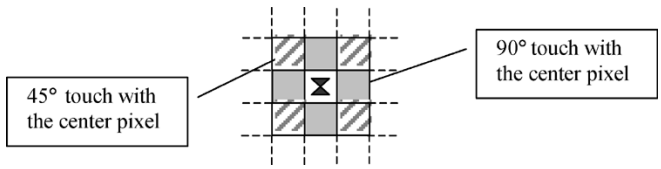


Fig. 14. Pixels that touch each others by 90° (i.e., $(i, j \pm 1)$ or $(i \pm 1, j)$) or by 45° (i.e., $(i + 1, j \pm 1)$ or $(i - 1, j \pm 1)$) and that have the same pixel value are often considered as *connected*.

the Microsoft Word 2000 program (with image importing resolution 72 dpi) using an HP LJ4100DTN laser printer, and scan back with 600 dpi resolution and 256 gray levels using a Microtek 3600 scanner. The image is binarized with a threshold that equals to the average of the maximum and minimum of scanned luminance values. We use the registration marks to determine the image boundary, to perform deskewing, and to compute the proper scaling factor. The estimated center of each original pixel on the scanned version is shown in Fig. 13(b). Sampling at those pixels can recover the original digital image perfectly from the scanned one hence allow the embedded data to be extracted correctly. More detailed discussion on recovering of binary image from printing and scanning can be found in [16].

The above mentioned approaches fall in the category of visible registration. The key purpose is to use the marks as reference to determine the boundary of binary image and the scaling factor after the printing-and-scanning process, and in turn, recover the image accurately. It should be noted that the accurate recovery of image is needed regardless of how the authentication data is stored (invisibly or visibly). Although we can attach authentication data separately, for example, to put a message digest or a cryptographic digital signature in the form of a text string or

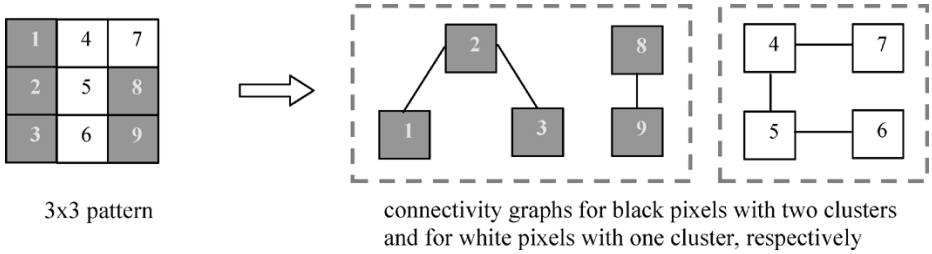


Fig. 15. Graph representation of connectivity for black and white pixels. Showing here is an example of 3×3 pattern with five black pixels forming two clusters and with four white pixels forming one cluster. Four-way connectivity (90° touching) is considered in this example.

a bar code next to the image to be authenticated, it does not solve the authentication problem under printing-and-scanning. This is because even though the authentication data can be easily and accurately obtained, one still has to recover the image to be authenticated, from which a message digest or digital signature is computed and compared with the attached authentication data.

Recovery from printing-and-scanning using fewer or no visible registration marks is desirable and is a direction of future work. A general approach is to embed an additional "pilot" sequence known to detectors. If a detector cannot successfully extract this sequence from a test image, it will perform an extensive search to find a set of distortion parameters so as to produce an undistorted image from which the pilot sequence is correctly extracted.

B. Security Considerations

Drawing analogy between data hiding and communication, the embedding methods serve as a physical communication layer, on top of which other functionalities and features can be built [15]. For instance, security issues may be handled by top layers in authentication applications. In these applications, the major objective of an adversary is to forge authentication data so that an altered document can still pass authentication tests.

The security issues can be addressed by using traditional cryptography-based authentication to produce a cryptographical digital signature and by embedding it in the associated binary image. This traditional approach relies on a cryptographically strong hash function to produce a digest of the document to be signed as well as on public-key encryption to enable verification without giving up the encryption keys, hence only authorized person can produce a correct signature [19]. By using embedding, we can associate the authentication data with the image in a seamless way.

Although a cryptographical signature can be adopted as (part of) the embedded data, the embedding approach proposed in this paper has the potential of allowing plaintext to be embedded since secret information such as keys/seeds have already been incorporated via shuffling and/or lookup table. Envisioning potential malicious attacks such as those studied in [20], it is important to study the following two problems for authentication applications, assuming that the attacker has no knowledge about any secret keys: 1) the probability of making content alterations while preserving the m-bit embedded authentication data, and 2) the possibility for an adversary to hide specific data in an image.

For the first problem, we have discussed in Section IV-A that an n-pixel alteration on a marked image would change the decoded data. If n is small compared to the total number of blocks m, there are approximately n bits in the decoded data that will be different from the originally embedded one; if n is large, the probability of getting the decoded data to be exactly the same as the originally embedded one is approximately 2^{-m}, which is very small as long as m is reasonably large. Therefore, the threat of making content alterations while preserving the m-bit embedded authentication data is very low.

For the second problem, it depends on whether or not multiple watermarked versions of the same image with different data embedded are available to an adversary. When multiple copies are not available, it is extremely hard for an adversary to embed specific data in an image, even if he/she knows the algorithm. This is due to the secrecy in shuffling table. However, in such applications as "signature in signature", an adversary may be able to obtain multiple copies, for example, signatures embedded with different signing date or different payment amount. This is similar to the plaintext attack in cryptography [19]. We would like to know whether an adversary can derive information regarding which pixels carry which bit by studying the difference between those copies hence create new images embedded with data at his/her will (e.g., specific date or payment amount). If the embedding imposes minimal necessary changes to enforce a desirable relationship (for example, in the odd–even case, at most one pixel will be flipped in each embedding block), the pixels that differ among the multiple copies are those used to embed hidden information. Assuming an adversary collects sufficiently many copies and knows what data is embedded in each copy, he/she will be able to identify which pixels carry which bit and to hide his/her desired data by manipulating the corresponding pixels.

To prevent the above mentioned attack, we have to introduce more uncertainty. One approach is to use a different shuffling table, for example, choose one table from K candidates, similar to the approach used for handling bad shuffles in [15], [16]. Another approach is that instead of making minimal changes for hiding one bit in each embedding block, we also flip, with probability of 0.5 in each block, an additional pair of flippable pixels. Consider an example of embedding a "0": if the number of black pixels in a shuffled block is already an even number, with a total probability of 0.5 we flip an additional pair of pixels selected equiprobably from three highly flippable pixels; if the number of black pixels is an odd number, with probability 0.5 we flip all three highly flippable pixels, and with another total probability of 0.5 we flip only one pixel selected equiprobably from

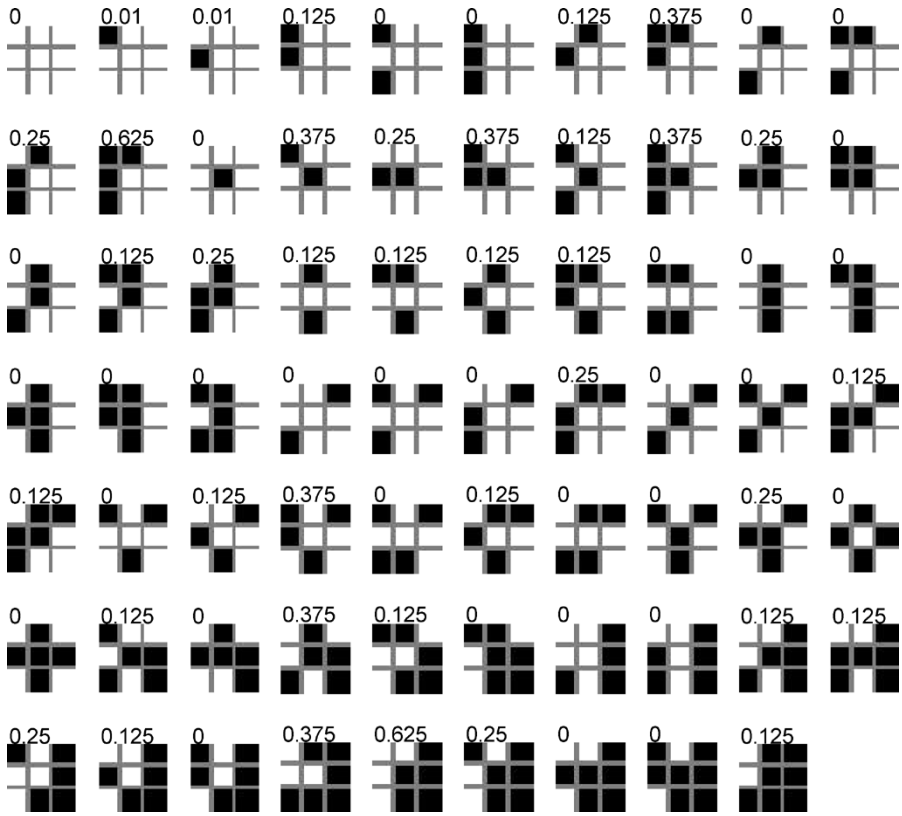


Fig. 16. One possible flippability lookup table for 3×3 patterns, excluding the patterns that differ only by rotation, mirroring, or complement. Larger value indicates that the change of center pixel is less noticeable hence the change is more likely to be made for hiding information.

those three pixels. When more than three highly flippable pixels are available, we may make the above selection from a larger pool. Now if we look at two image copies whose hidden data differ in just one bit, the difference between the two images via minimal-change embedding is just at one pixel, while the difference via the above-mentioned randomization involves many other pixels in a random fashion. In the latter case, if a total of N bits are embedded, we can show that on average $(4N+1)/3$ pixels will be different. When N is sufficiently large, it is difficult for an adversary to identify which pixels are associated with which bits. As a tradeoff, the randomization requires three flippable pixels to be available for each shuffled block and changes more pixels at the embedding step. Note that this countermeasure assumes that for any given hidden data, only one copy of a marked image is available to an attacker. Otherwise, he/she may be able to average out the randomization and compromise our solution.

V. Conclusions

This paper addresses the problem of data hiding for binary image. We propose a new fragile to semifragile data hiding method for authentication and annotation of binary images. The method manipulates "flippable" pixels to enforce a specific block-based relationship to embed a significant amount of data without causing noticeable artifacts. Shuffling is applied before embedding to equalize the uneven embedding capacity. The hidden data can be extracted without using the original image. With the help of a few registration marks, they can also be accurately extracted after high quality printing and scanning. The algorithm can be applied to detect unauthorized use of signatures in binary image format, to detect alterations on documents, and to annotate signatures and drawings.

Some directions for future investigation include the further refinement of flippability model for different types of binary images such as texts, drawings, and dithered images, as well as the recovery of binary image from high quality printing and scanning using fewer or no visible registration marks.

Appendix
Details on Flippability Scores

In this Appendix, we describe a procedure for computing flippability scores of pixels in nondithered binary image. The scores are used to determine which pixels to flip with high priority during the embedding process. We use a 3×3 window centered on the pixel. The procedures can be further refined by studying a larger neighborhood and by using more extensive analysis, especially for Step-2. The parameters and rules should be adjusted for dithered image.

Step-1) Compute Smoothness and Connectivity Measures of 3 × 3 Patterns

The *smoothness* of the neighborhood around pixel (i,j) is measured by the total number of transitions along four directions in the 3×3 window:

$$\text{horizontal}$$
$$N_h(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{0} \mathcal{I}\left(\{p_{i+k,j+l} \neq p_{i+k,j+l+1}\}\right),$$
$$\text{vertical}$$
$$N_v(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{0} \mathcal{I}\left(\{p_{i+l,j+k} \neq p_{i+l+1,j+k}\}\right),$$
$$\text{diagonal}$$
$$N_{d_1}(i,j) = \sum_{k,l\in\{-1,0\}} \mathcal{I}\left(\{p_{i+k,j+l} \neq p_{i+k+1,j+l+1}\}\right),$$
$$\text{anti}-\text{diagonal}$$
$$N_{d_2}(i,j) = \sum_{k\in\{0,1\},l\in\{-1,0\}} \times \mathcal{I}\left(\{p_{i+k,j+l} \neq p_{i+k-1,j+l+1}\}\right)$$

where $\mathcal{I}(\cdot)$ is an indicator function taking value from $\{0,1\}$, and $p_{i,j}$ denotes the pixel value of the i^{th} row and j^{th} column of the image.

The *connectivity* is measured by the number of the black and white clusters. A commonly used criterion, illustrated in Fig. 14, considers the pixels that touch each others by 90° or 45° and have the same pixel value as *connected*. The 90° touching alone is often known as *four-way connectivity*, and 90° plus 45° touching is known as *eight-way connectivity* [21]. We choose between four-way and eight-way connectivity criterion depending on the specific constraints of visual artifacts. A graph for black (or white) pixels can be constructed, in which each vertex represents a black (or white) pixel, and there is an edge between two vertices if and only if the two corresponding pixels are connected. An example is shown in Fig. 15 with five black pixels forming two clusters and four white pixels forming one cluster. The number of clusters can be automatically identified by traversing the graph via *depth-first search* [22].

Step-2) Compute Flippability Score

The smoothness and connectivity measures are passed into a decision module to produce a flippability score. Main considerations when designing this module are 1) whether the original pattern is smooth, 2) whether flipping will increase nonsmoothness by a large amount, and 3) whether flipping will cause any change in connectivity. The changes on these patterns are generally more noticeable. Listed here are the rules used by our decision module.

1) The lowest score (i.e., not flippable) is assigned to uniformly white or black regions as well as to the isolated single white or black pixels. These trivial cases are handled first.
2) If the number of transitions along horizontal or vertical direction is zero (i.e., the pattern is smooth and regularly structured), assign zero as a final score for the current pixel. Otherwise, assign to the pixel a base score S_B and proceed to the next rule.
3) If the number of transitions along diagonal or anti-diagonal direction is zero, reduce the score. Otherwise, if the minimum number of transition points along any one of the four directions is below a given threshold T_1, which means the pattern is somewhat smooth, reduce the score by a smaller amount.
4) If flipping the center pixel does not change the number of transition points, increase the score. Otherwise, if flipping results in the increase of transition points (i.e., reduces smoothness and makes the pattern more noisy), decrease the score.
5) If flipping changes the number of black clusters or white clusters, reduce the score.

Applying these rules produces a lookup table of all 3×3 patterns ordered in terms of how unnoticeable the change of the center pixel will cause. For small neighborhood such as 3×3, this table has a small number of entries ($2^{3\times 3} = 512$) hence can be off-line computed. The flippability score of every pattern in an image can then be determined by looking up the stored table. When larger neighborhood is involved, the table size increases exponentially and may exceed the available memory size for particular applications. We solve this problem by a hierarchical approach, namely, to obtain preliminary flippability measure based on a small neighborhood (e.g., 3×3) by table lookup, and then if necessary, to refine the measure by on-line computing based on a larger neighborhood (see also Step-3).

Shown in Fig. 16 is one possible lookup table for 3×3 patterns. For patterns that differ only by rotation, mirroring, or complement, we only show one representative to save space. Here we set the threshold $T_1 = 3$, the base flippability score $S_B = 0.5$, and the flippability adjustments in substeps (3)–(5) are multiples of 0.125.

Step-3) Handle Special Cases

Some special cases involving larger neighborhood can be handles by detecting specific patterns such as sharp corners to avoid introducing annoying artifacts on them.

Step-4) Impose Minimum Distance Constraint Between Two Flippable Pixels

Up to now, the flippability evaluation is done independently for the pattern revealed in a moving window centered at each pixel, assuming that any pixels other than the center one will not be flipped. Pixels that are close to each others may be considered flippable by this independent study, but simultaneously flipping them could cause artifacts. We handle this problem by imposing constraints on the minimum distance between two pixels that can be flipped and pruning the pixels with relatively low flippability in its neighborhood.

Step-5) Assign a Predetermined Score to the Remaining Boundary Points (Optional)

Edge pixels that have not yet been assigned a nonzero score will be given a small flippability score. These pixels serve as a base line for hiding a particular bit when there is no pixel

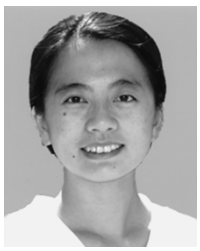
with higher score available to carry the data to be embedded. Adding this step helps to achieve a high embedding rate without significantly affecting the visual quality.

ACKNOWLEDGMENT

The authors would like to thank Prof. A. Finkelstein of Princeton University for the enlightening discussion on data hiding in binary image and for suggesting its application of "signature in signature", and E. Tang of the Princeton Summer Institute '99 for his contribution to the connectivity criterion for generating flippability scores. The signature image in Fig. 9 was edited from http://www.whitehouse.gov/ WH/EOP/OP/html/ OP_Home.html as of the 2000, the artistic line drawing in Fig. 10 was from the *Clip Art* collections of *Microsoft Office* software, and the patent image in Fig. 11 was edited from http://www.patents.ibm.com/details ?&pn=US05 825 892_ _.

REFERENCES

[1] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary image," in *IEEE Int. Conf. Multimedia & Expo (ICME'00)*, New York, 2000.
[2] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—a survey," *Proc. IEEE*, vol. 87, pp. 1062–1078, July 1999.
[3] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proc. IEEE*, vol. 87, pp. 1079–1107, July 1999.
[4] I. J. Cox, M. L. Miller, and J. A. Bloom, *Digital Watermarking*. San Mateo, CA: Morgan Kaufmann, 2001.
[5] I. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Processing*, vol. 6, pp. 1673–1687, Dec. 1997.
[6] C. Podilchuk and W. Zeng, "Image adaptive watermarking using visual models," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 525–538, May 1998.
[7] M. Wu, H. Yu, and B. Liu, "Data hiding in images and videos: Part II—Designs and applications," *IEEE Trans. Image Processing*, vol. 12, pp. 696–705, June 2003.
[8] M. Wu and B. Liu, "Watermarking for image authentication," in *Proc. IEEE Int. Conf. Image Processing (ICIP'98)*, vol. 2, Chicago, IL, 1998, pp. 437–441.
[9] M. M. Yeung and F. Mintzer, "An invisible watermarking technique for image verification," in *Proc. IEEE ICIP'97*, vol. 2, Santa Barbara, CA, 1997, pp. 680–683.
[10] K. Matsui and K. Tanaka, "Video-steganography: how to secretly embed a signature in a picture," *Proc. IMA Intellectual Property Project*, vol. 1, no. 1, 1994.
[11] N. F. Maxemchuk and S. Low, "Marking text documents," in *Proc. IEEE ICIP'97*, 1997.
[12] E. Koch and J. Zhao, "Embedding robust labels into images for copyright protection," in *Proc. Int. Congr. Intellectual Property Rights for Specialized Information, Knowledge & New Technologies*, 1995.
[13] A. K. Bhattacharjya and H. Ancin, "Data embedding in text for a copier system," in *Proc. IEEE ICIP'99*, vol. 2, Kobe, Japan, 1999 , pp. 245–249.
[14] Y. Liu, J. Mant, E. Wong, and S. H. Low, "Marking and detection of text documents using transform-domain techniques," in *Proc. SPIE, Electronic Imaging (EI'99) Conference on Security and Watermarking of Multimedia Contents*, vol. 3657, San Jose, CA, 1999, pp. 317–327.
[15] M. Wu and B. Liu, "Data hiding in image and video: Part-I—Fundamental issues and solutions," *IEEE Trans. Image Processing*, vol. 12, pp. 685–695, June 2003.
[16] M. Wu. (2001) Multimedia Data Hiding, Ph.D. dissertation. Princeton Univ., Princeton, NJ. [Online] Available http://www.ece.umd.edu/ ~minwu/research/phd_thesis.html
[17] M. Wu and B. Liu, "Digital watermarking using shuffling," in *Proc. IEEE ICIP'99*, vol. 1, Kobe, Japan, 1999, pp. 291–295.
[18] A. Finkelstein, "personal communication," unpublished, 1998.
[19] B. Schneier, *Applied Cryptography: Protocol, Algorithms, and Source Code in C*, 2nd ed. New York: Wiley, 1996.
[20] M. Holliman and N. Memon, "Counterfeiting attacks on oblivious blockwise independent invisible watermarking schemes," *IEEE Trans. Image Processing*, vol. 9, pp. 432–441, Mar. 2000.
[21] K. R. Castleman, *Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[22] R. Sedgewick, *Algorithms in C*. Reading, MA: Addison-Wesley, 1990.



Min Wu (S'95–M'01) received the B.E. degree in electrical engineering and the B.A. degree in economics from Tsinghua University, Beijing, China, in 1996 (both with the highest honors), and the M.A. degree and Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 1998 and 2001, respectively.

She was with the NEC Research Institute and Signafy, Inc., in 1998, and with Panasonic Information and Networking Laboratories in 1999. Since 2001, she has been an Assistant Professor of the Department of Electrical and Computer Engineering, Institute of Advanced Computer Studies and the Institute of Systems Research, University of Maryland, College Park. Her research interests include information security, multimedia signal processing, and multimedia communications. She holds four U.S. patents on multimedia data hiding. She co-authored the book *Multimedia Data Hiding* (New York: Springer-Verlag, 2002), and is a Guest Editor of a special issue on Multimedia Security and Rights Management of the *EURASIP Journal on Applied Signal Processing*.

Dr. Wu received a CAREER award from the U.S. National Science Foundation in 2002. She is a member of the IEEE Technical Committee on Multimedia Signal Processing, and Publicity Chair of 2003 IEEE International Conference on Multimedia and Expo (ICME'03, Baltimore, MD).

Bede Liu (S'55–M'62–F'72) was born in China. He received the B.S.E.E. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1954 and the D.E.E. degree Polytechnic Institute of Brooklyn, Brooklyn, NY, in 1960.

He is Professor of electrical engineering at Princeton University, Princeton, NJ. Prior to joining Princeton University in 1962, he had been with Bell Laboratories, Allen B. DuMont Laboratory, and Western Electric Company. He has also been a visiting faculty member at several universities in U.S. and abroad. His current research interest lies mostly with multimedia technology, with particular emphasis on digital watermarking and video processing.

Dr. Liu and his students have received the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY Best Paper Award for 1994, 1996, and 2003. His other IEEE awards include Centennial Medal (1984) and Millennium Medal (2000), Signal Processing Society's Technical Achievement Award (1985) and Society Award (2000), Circuit and Systems Society's Education Award (1988) and Mac Van Valkenburg Award (1997). He is a member of the National Academy of Engineering. He was the President of the IEEE Circuit and Systems Society (1982), and the IEEE Division I Director (1984, 1985). He also served as the 1978 ISCAS Technical Program Chair and the General Chair of the 1995 ICIP.