EURASIP Journal on Applied Signal Processing 2002:2, 143–151
© 2002 Hindawi Publishing Corporation

Multimedia Data Hiding and Authentication via Halftoning and Coordinate Projection

Chai Wah Wu
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA
Email: chaiwah@watson.ibm.com

Received 4 May 2001 and in revised form 14 September 2001

We present image data hiding and authentication schemes based on halftoning and coordinate projection. The proposed data hiding scheme can embed images of the same size and similar bit depth into the cover image and robustness against compression is demonstrated. The image authentication scheme is based on the data hiding scheme and can detect, localize, and repair the tampered area of the image. Furthermore, the self-repairing feature of the authentication scheme has a hologram-like quality; any portion of the image can be used to reconstruct the entire image, with a greater quality of reconstruction as the portion size increases.

Keywords and phrases: data hiding, authentication, tamper detection, self-repairing images, error diffusion, digital halftoning, coordinate projection.

1. INTRODUCTION

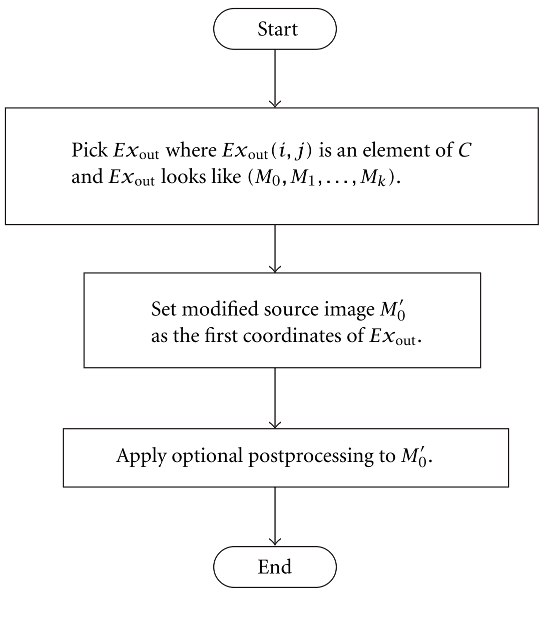
Recently, there has been much interest in multimedia data hiding, where information is imperceptibly embedded into multimedia content such as images and music [1]. The embedded data can contain ownership identification, tracking information, recipient information, time stamps, authentication data, and other information useful for various applications such as copyright protection, data integrity verification, verification of origin of data, recipient tracking, and so forth. In many applications, the main requirement is that the embedding changes the multimedia content imperceptibly. In this paper, we describe a system of embedding images into a source image by using digital halftoning algorithms and coordinate projections. In particular, we show how an entire image can be embedded into another source image and illustrate design requirements for robustness against distortions such as JPEG compression. The proposed embedding method can also be used in an image authentication scheme, where changes to an image can be detected, localized, and repaired. The repair operation reconstructs the original source image using information distributed over the entire image. In this sense it exhibits hologram-like properties; any portion of the image can be used to reconstruct the entire image. The larger the portion used for reconstruction, the higher the fidelity of the reconstructed image.

2. DATA EMBEDDING AND EXTRACTION ALGORITHMS

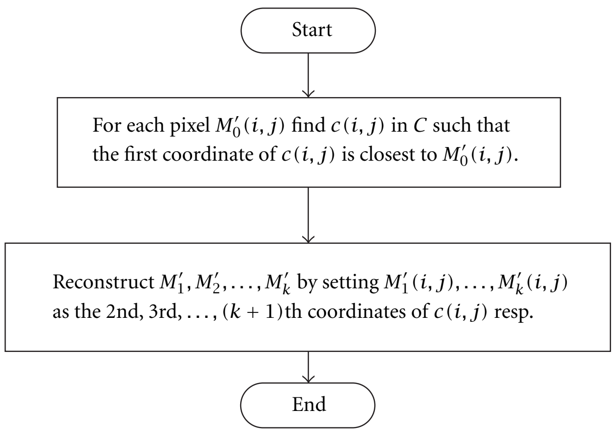
We consider images represented as matrices of vectors. For example, an image M is represented as an $n \times m$ matrix of d-dimensional vectors. The (i, j)th entry of M, denoted as $M(i, j)$, is called the (i, j)th pixel of M. Each pixel $M(i, j)$ is a d-dimensional vector, where d denotes the dimension of the color space, that is, $d = 1$ for a grayscale image, $d = 3$ for an image in RGB or LAB color space and $d = 4$ for an image in CMYK space. We assume that each pixel is in the d-dimensional set S^d. In many current image formats, $S^d = \{0, 1, \dots, 255\}^d$ is d bytes of data. In some applications, we also represent M as d matrices of scalars. For instance, a color image in RGB space can be considered as 3 grayscale images by separating the image into its separate color planes.

Given a source image M_0 and additional images $M_1, M_2, \dots, M_k$, the proposed data hiding method consists of two algorithms. The data embedding algorithm embeds $M_1, M_2, \dots, M_k$ imperceptibly into the source image (or cover image) M_0 resulting in a *modified source image* M_0'. The data extraction algorithm extracts the images $M_1, M_2, \dots, M_k$ from M_0'.

We first give a general description of these algorithms. A set C of extended colors is first chosen. Each member of this set C is a $(k + 1)$-tuple of d-dimensional vectors. For a member $D \in C$, each of the $k + 1$ d-dimensional vectors of D is called a *coordinate* of D. For example, for $D = (c_1, c_2, \dots, c_{k+1})$ where each c_i is a d-dimensional vector, the first, second, $\dots, (k + 1)$th coordinates of D are $c_1, c_2, \dots, c_{k+1}$, respectively. The set C is chosen such that it satisfies the following condition: for each j in the set S^d, there exists at most one member of C such that the first coordinate is j. This implies that the size of C is less than or equal to the size of S^d ($= 256^d$, assuming $S^d = \{0, 1, \dots, 255\}^d$) and projection of C into the first coordinate is a one-to-one map. In


Start
Pick Ex_{out} where $Ex_{\text{out}}(i,j)$ is an element of C and Ex_{out} looks like $(M_0, M_1, \dots, M_k)$.
Set modified source image M_0' as the first coordinates of Ex_{out}.
Apply optional postprocessing to M_0'.
End

(a)


Start
For each pixel $M_0'(i,j)$ find $c(i,j)$ in C such that the first coordinate of $c(i,j)$ is closest to $M_0'(i,j)$.
Reconstruct $M_1', M_2', \dots, M_k'$ by setting $M_1'(i,j), \dots, M_k'(i,j)$ as the 2nd, 3rd, $\dots$, $(k+1)$th coordinates of $c(i,j)$ resp.
End

(b)

FIGURE 1: (a) Flowchart of data embedding algorithm. (b) Flowchart of data extracting algorithm.

other words, C is of the form $(x, f(x))$, $x \in U \subset S^d$ where f is a map from S^d into $(S^d)^k$. In some applications we require additionally that C is of the same size as S^d. This means that for each j in the set S^d, there exists exactly one member of C such that the first coordinate is j and the projection of C into its first coordinates is a bijection p_C from C onto S^d. In this case, $U = S^d$.

We embed the images $M_1, \dots, M_k$ into M_0 as follows: an extended output image Ex_{out} is chosen as a matrix of elements of C. Then the modified source image M_0' is generated by taking the first coordinate of the entries of Ex_{out}, that is, $M_0'(i,j)$ is the first coordinate of $Ex_{\text{out}}(i,j)$.

If the size of C is strictly less than the size of S^d, in general M_0' is different from M_0 and the following optional postprocessing step can be implemented to change M_0' in order to reduce the difference between M_0 and M_0'. For each $M_0'(i,j)$, let $T(i,j)$ be the set of points in S^d which are strictly closer to $M_0'(i,j)$ than to any other first coordinates of elements of C. Then pick the new $M_0'(i,j)$ as the element of $T(i,j)$ which is closest to $M_0(i,j)$. Note that if the size of C is equal to the size of S^d, then $T(i,j)$ consists of a single element $M_0'(i,j)$ and the postprocessing step does not change M_0' at all.

To extract the embedded images from M_0', the following algorithm is used. For each of the pixels $M_0'(i,j)$ of M_0', find the element $c(i,j)$ in C whose first coordinate is closest to $M_0'(i,j)$.[1] Then generate the reconstructed embedded images $M_1', M_2', \dots, M_k'$ by setting $M_u'(i,j)$ equal to the $(u+1)$th coordinate of $c(i,j)$, $u = 1, \dots, k$. It is clear that the postprocessing step does not affect the extraction of $M_1', \dots, M_k'$. Flowcharts of these two algorithms are shown in Figure 1.

To illustrate these ideas, consider the following simple example of embedding a sequence M_1 into another sequence M_0 (i.e., $k = 1$). Let $C = \{(1,2), (4,4), (7,2), (10,8)\}$, $M_0 = (4, 11, 2, 7, 4)$, and $M_1 = (3, 8, 2, 2, 4)$. Then $(M_0, M_1) = ((4,3), (11,8), (2,2), (7,2), (4,4))$. Suppose we pick Ex_{out} as $((4,4), (10,8), (1,2), (7,2), (4,4))$. Note that each of the pairs in Ex_{out} is an element of C. Then M_0' is obtained as the first coordinates of the elements in Ex_{out} : $M_0' = (4, 10, 1, 7, 4)$. After the postprocessing step we obtain the embedded source sequence $M_0' = (4, 11, 2, 7, 4)$ which is equal to M_0. Applying the data extracting algorithm to M_0', we obtain $M_1' = (4, 8, 2, 2, 4)$ which is similar to M_1.

It is clear that $M_0', M_1', M_2', \dots, M_k'$ form the coordinates of Ex_{out}. Since Ex_{out} consists of elements of C and $(M_0, M_1, \dots, M_k)$ consist of elements of $(S^d)^{k+1}$, and in general C is a small subset of $(S^d)^{k+1}$, it follows that in general M_i' will not be the same as M_i. The goal is to pick Ex_{out} appropriately in order to ensure that the images $M_0', M_1', M_2', \dots, M_k'$ *look like* $M_0, M_1, M_2, \dots, M_k$, respectively. To accomplish this, the problem is recast as a halftoning problem and a suitable halftoning algorithm is used to pick the entries of Ex_{out}.

We now describe the proposed algorithms in more detail. Given a source image M_0 and k auxiliary images $M_1, M_2, \dots, M_k$, the goal is to create a modified image M_0' which looks like the source image M_0 such that images $M_1', \dots, M_k'$ can be extracted from M_0' and $M_1', \dots, M_k'$ look like $M_1, M_2, \dots, M_k$, respectively. We say that the images $M_1, M_2, \dots, M_k$ are *embedded* into the modified image M_0'. We denote $M_u(i,j)$ as the (i,j)th pixel of image M_u. Each pixel of M_u is a scalar or a vector, depending on whether the image is a grayscale image or a color image. Furthermore, the pixels of each image M_u can be represented in different color spaces, that is, M_1 can be in RGB space, M_2 can be in LAB space, and so forth.

In a halftoning problem, given an input image I, the goal is to generate a halftone image H, where each pixel of H is

[1]If the bijection p_C exists, then $c(i,j) = p_C^{-1}(M_0'(i,j))$. In this case the first coordinate of $c(i,j)$ is $M_0'(i,j)$.

from a restricted set of output colors, such that H "looks" like I when viewed at the proper distance. Given the algorithm to extract $M_1', \ldots, M_k'$ as described earlier, the problem of choosing Ex_{out} can be recast as a halftoning problem. Consider the image X where the (i, j)th pixel is the vector $(M_0(i, j), M_1(i, j), M_2(i, j), \ldots, M_k(i, j))$. The set of possible output colors is C. The corresponding halftoning problem is to generate a halftone image where every pixel is an element of C such that the halftone image looks like X. Then the solution obtained by a halftoning algorithm which solves this halftoning problem is used as the extended output image Ex_{out}. Some halftoning algorithms we use for this purpose will be vector error diffusion and local iterative methods.

How does one quantify the requirement that the halftone image "looks" like the original image? One commonly used way is to model the human visual system (HVS) as a linear low-pass filter. The image A "looks" like image B if $\|L(A-B)\|$ is small, where L denotes the linear low-pass operator of the human visual system. In our case, the halftoning problem becomes the following optimization problem: find the output image Ex_{out} consisting of pixels from the set C such that

$$\sum_{u=0}^{k} v_u \| L_u(Ex_{\text{out}}^u - M_u) \|^{p_u} \tag{1}$$

is minimized where Ex_{out}^u denotes the image extracted from Ex_{out} by taking the $(u+1)$th coordinates of each pixel of Ex_{out}. The exponents p_u are generally chosen to be 2. The weights v_u correspond to different weighting factors for the different images. For example, if the closeness between the modified source image M_0' and the original source image M_0 is important, then the weight v_0 should be larger than the other weights. L_u is the linear low-pass filter for the image M_u. When the images M_u are all in the same color space, $L_u = L$ is the same for all u. We will discuss a case later where the images have different modalities (some of the M_u's are not even images) and L_u is different for different u's. Examples of the linear filter L can be found in Näsänen [2] and Sullivan et al. [3].

Consider the use of vector error diffusion [4] as the halftoning method to minimize (1). We pick the (i, j)th pixel of the extended output image $Ex_{\text{out}}(i, j)$ as the member of C which is the closest to the $(k+1)$-tuple

$$(\tilde{M}_0(i, j), \tilde{M}_1(i, j), \ldots, \tilde{M}_k(i, j)). \tag{2}$$

The notion of closeness can be the Euclidean norm or a weighted Euclidean norm, with different weights for the different images $\tilde{M}_0, \ldots, \tilde{M}_k$, for example, the distance between a member of C, namely, $(c_0, c_1, \ldots, c_k)$, and $(\tilde{M}_0(i, j), \tilde{M}_1(i, j), \ldots, \tilde{M}_k(i, j))$ is $(\sum_u v_u \|\tilde{M}_u(k, l) - c_u\|^p)^{1/p}$.

The difference between $(\tilde{M}_0(i, j), \tilde{M}_1(i, j), \ldots, \tilde{M}_k(i, j))$ and $Ex_{\text{out}}(i, j)$ is an error vector $e(i, j)$ defined as

$$\begin{aligned} e(i, j) &= (e_0(i, j), \ldots, e_k(i, j)) \\ &= (\tilde{M}_0(i, j), \tilde{M}_1(i, j), \ldots, \tilde{M}_k(i, j)) - Ex_{\text{out}}(i, j). \end{aligned} \tag{3}$$

$\tilde{M}_u(i, j)$ is the modified input for M_u at location (i, j) and is defined as

$$\tilde{M}_u(i, j) = M_u(i, j) + \sum_{x,y} w_u(x, y) e_u(i - x, j - y), \tag{4}$$

where w_u are the error diffusion weights corresponding to M_u. See [5] for the design of error diffusion weights appropriate for the chosen human visual system model.

After $Ex_{\text{out}}(i, j)$ is constructed, the first coordinate of the $(k+1)$-tuple $Ex_{\text{out}}(i, j)$ will be used to form the (i, j)th pixel of the modified source image M_0'. Pseudocode of this algorithm is shown in Algorithm 1. We have omitted the optional postprocessing step in this description as it is the same as before and this step is not needed if the size of C is equal to the size of S^d.

To extract the auxiliary images $M_1', \ldots, M_k'$ from M_0', for each pixel $M_0'(i, j)$ find the $(k+1)$-tuple in C whose first coordinate is closest to $M_0'(i, j)$. Then $M_1'(i, j), \ldots, M_k'(i, j)$ are set to be the 2nd, $\ldots$, $(k+1)$th coordinates of this $(k+1)$-tuple, respectively.

In certain applications, some of the embedded data $M_1, M_2, \ldots, M_k$ are not images, but rather information bits containing the name of the owner, the date, compressed image data, and so forth. Without loss of generality, we assume that $M_u (u \geq q)$ are this type of embedded data. In this case these recovered data M_u' should match exactly the embedded data M_u for $u \geq q$. A necessary condition for this to happen is that each pixel of $(M_q, \ldots, M_k)$ is equal to the $q+1, \ldots, (k+1)$th coordinates of some element of C. To ensure that M_u' matches M_u, v_u is set to be very large and L_u is set to be the impulse delta function (all-pass filter) for $u \geq q$. This will make sure that the error term of (1) corresponding to data M_u is $v_u \|Ex_{\text{out}}^u - M_u\|^p$ for $u \geq q$ which is minimized by choosing $M_u' = Ex_{\text{out}}^u = M_u$ for $u \geq q$. In the vector error diffusion algorithm this is accomplished by setting the error diffusion weights $w_u = 0$ for $u \geq q$. This makes $\tilde{M}_u = M_u$ for $u \geq q$, and thus $Ex_{\text{out}}^u(i, j)$ is chosen to be the closest to $M_u(i, j)$ which is equal to setting $Ex_{\text{out}}^u(i, j) = M_u(i, j)$ for $u \geq q$ by the necessary condition above. Thus for the uth image $(u \geq q)$, error diffusion is not used and the image is simply quantized.

In particular, consider the special case $q = k = 1$, and $C = (x, f(x))$ where $x \in \cup S_i$ with each S_i a discrete set of quantization points and f is defined as $f(x) = i$ if $x \in S_i$. If M_1 consists of indices i and $L_0 = L_1$ is the impulse delta function, then the resulting scheme is equivalent to quantization index modulation (QIM) when the information bits are embedded into in the space of image pixels [6]. In this case, we get better results by setting L_0 equal to an HVS low-pass filter and apply halftoning (such as error diffusion) in addition to QIM. For example, in QIM applied to image pixels, each image pixel is quantized according to a quantizer indexed by the information bits. If error diffusion is used as the halftoning algorithm, the modified input pixel $\tilde{M}_0(i, j)$ rather than the image pixel $M_0(i, j)$ is quantized by the indexed quantizer.

Another case where error diffusion is not used for the data M_u is when the information in M_u does not have the spatial

```
For each $i$                                              /* For each row*/
  For each $j$                                            /* For each column*/
    For each $u$                                          /* For each image*/
      $\tilde{M}_u(i,j) = M_u(i,j) + \sum_{x,y} w_u(x,y)e_u(i-x,j-y)$
    Endfor
    $Ex_{\text{out}}(i,j) = \underset{c\in C}{\operatorname{argmin}}\big(\sum_u v_u \|\tilde{M}_u(k,l) - c_u\|^p\big)^{1/p}$     /* $c_u$ is the $(u+1)$th coordinate of c*/
    $(e_0(i,j),\dots,e_k(i,j)) = (\tilde{M}_0(i,j),\tilde{M}_1(i,j),\dots,\tilde{M}_k(i,j)) - Ex_{\text{out}}(i,j)$
  Endfor
Endfor
Set modified source image $M_0'$ as the first coordinates of $Ex_{\text{out}}$.
```

Algorithm 1: Pseudocode of data hiding algorithm using vector error diffusion.

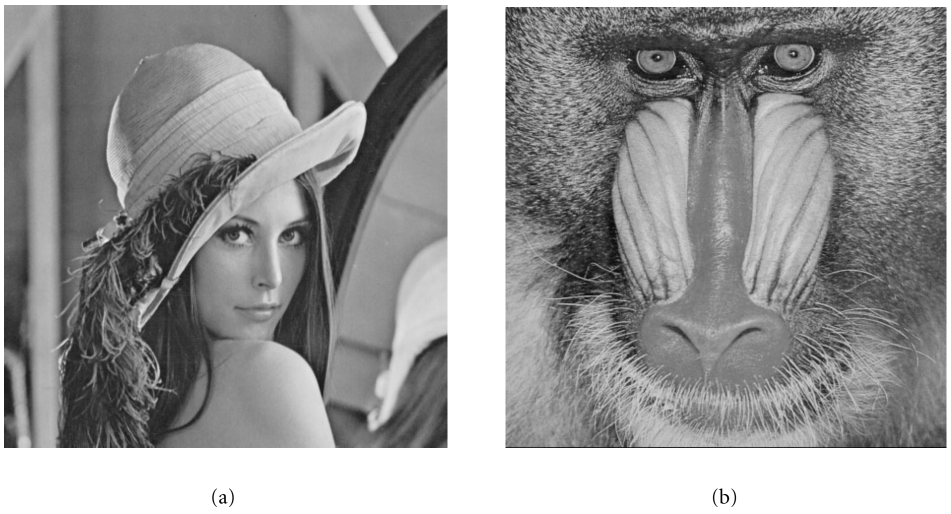


Figure 2: (a) Source image M_0. (b) Embedded image M_1.

relationship that error diffusion mandates, for example, when M_u are the DCT coefficients of an image.

3. EXAMPLES

We illustrate the data hiding scheme by considering the case of a grayscale image, that is, the dimension of the color space d as defined earlier is equal to 1. We assume that each pixel in the image is an integer between 0 and 255. Furthermore, assume that we want to embed a single image M_1 into the source image M_0 (i.e., $k = 1$).

Consider the source image M_0 shown in Figure 2a and an auxiliary image M_1 shown in Figure 2b. Figure 3a shows the modified source image M_0' after embedding M_1. Figure 3b shows the extracted image M_1'.

For vector error diffusion, the convex hull of the output color set C should cover as much as possible the space of possible $(k+1)$-tuples of d-dimensional vectors [7], or in this case pairs of integers in the range $[0, 255]$. Thus the convex hull of C should cover a large area of the square $[0, 255]^2$. As discussed before, because of the requirements on C, the set C must be a subset of a set of the form $(a, f(a))$ for some function f and a an integer in the range $[0, 255]$. In fact, in the examples, we will choose C to be equal to the set $(a, f(a))$ for some function f where $a = 0, 1, \dots, 255$. Figure 4a shows a plot of one such choice of C. We see that the convex hull of C covers almost all of $[0, 255]^2$ and the points of C are evenly distributed over $[0, 255]^2$. This set C is used to generate Figure 3a. This particular C will work well if the pixels of the two images are uniformly distributed with no cross correlation, a reasonable assumption to make when no further information is known about the images. If the joint probability distribution of the pixels in the two images are known, a more optimal C can be found by distributing the points of C according to this distribution by minimizing the expected distance to the nearest point in C while satisfying

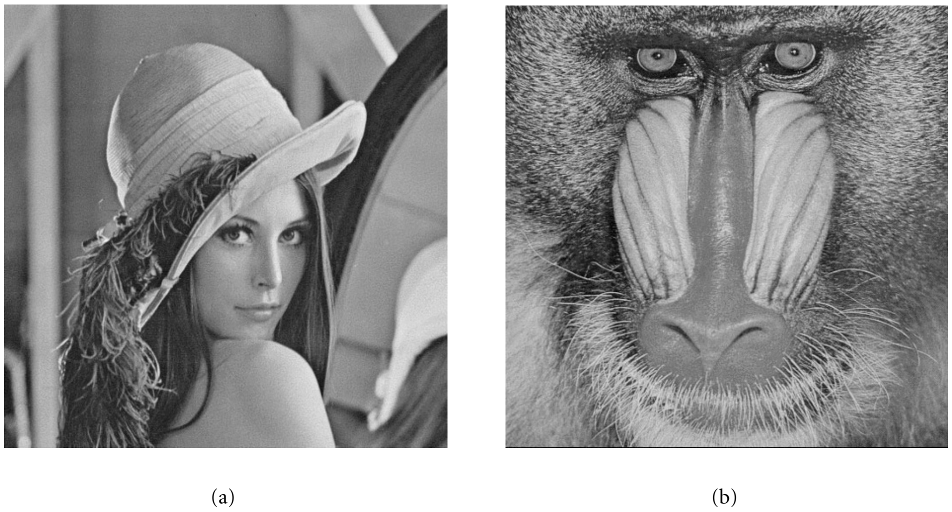


FIGURE 3: (a) Modified image M_0' with M_1 embedded. (b) Extracted image M_1'.

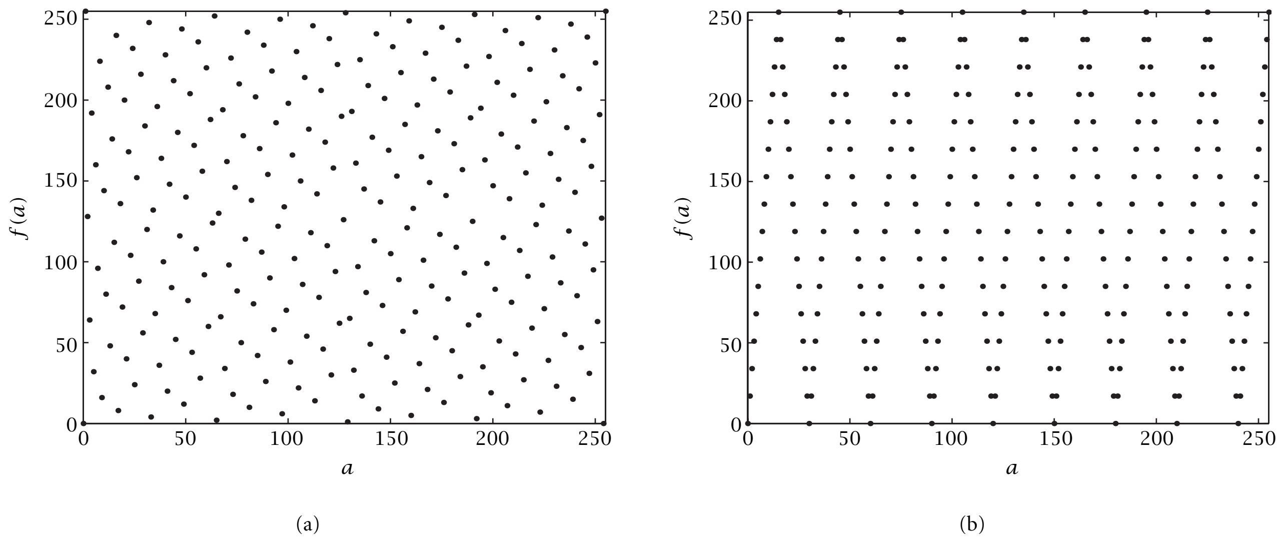


FIGURE 4: Candidates for the set C. (a) Distributed points covering $[0, 255]^2$. (b) A relative smooth function f.

the constraint that C is of the form $(a, f(a))$.

As indicated in Section 2 for the general case, the projection of C onto the first coordinate is the set $\{0, 1, \ldots, 255\}^d$, and C can be thought of as a set of the form $(x, f(x))$ where f is a function from $\{0, 1, \ldots, 255\}^d$ into $\{0, 1, \ldots, 255\}^{kd}$.

To cover a large area and be uniformly distributed across $[0, 255]^2$, the function f that defines C is generally highly discontinuous. This implies that small changes in the modified source image M_0' result in large changes in the extracted image M_1'. By using a smoother function for C, this effect can be reduced resulting in more robustness for the extracted image against changes in the modified source image. An example of such a smoother C is shown in Figure 4b. This usually comes at a cost of the convex hull of C covering a smaller area of $[0, 255]^2$ and the points of C being less distributed. This can result in larger errors in the error diffusion process and generates M_0', M_1' that are less faithful to the original images M_0, M_1, respectively. One way to avoid large errors is to project the input colors in $[0, 255]^2$ into the convex hull of C [7]. Note that the range of f in Figure 4b consists of 16 discrete values. Thus M_1 is halftoned into M_1' via 4-bit multilevel error diffusion. Actually, the halftoning is somewhat worse due to the constraint of also halftoning M_0.

We embed the image in Figure 2b into the image in Figure 2a using the set C in Figure 4b. Next the modified image M_0' (Figure 5a) is compressed using JPEG lossy compression with a high quality factor (90% in the program xv). After decompression, the embedded image M_1' is extracted as

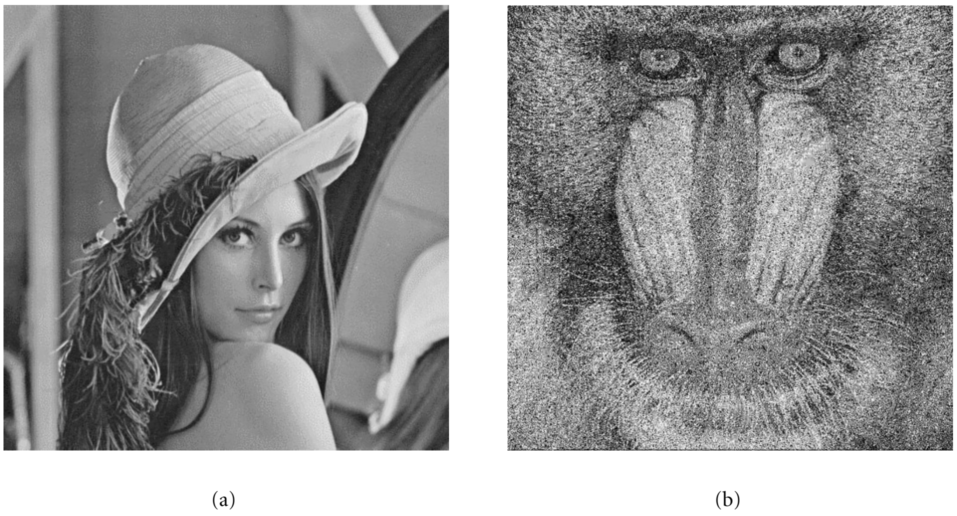


FIGURE 5: (a) Modified image M'_0 with M_1 embedded using the set C in Figure 4b. (b) Extracted image M'_1 after JPEG compression of M'_0.

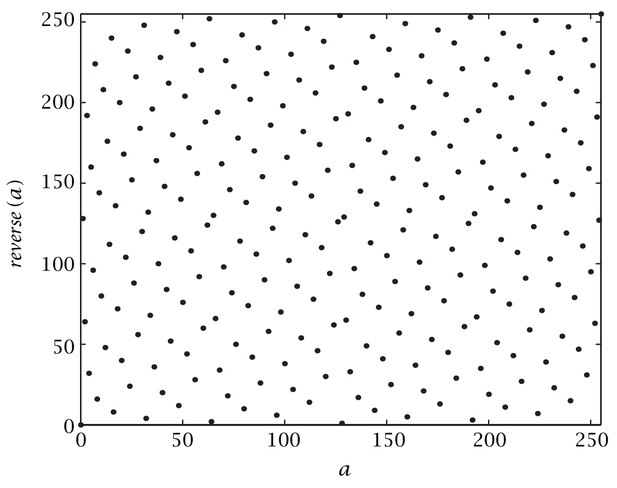


FIGURE 6: The set $C = (a, \mathit{reverse}(a))$ where $\mathit{reverse}(a)$ is a with the bits reversed.

shown in Figure 5b. We see that distortion has occurred due to the JPEG lossy compression, but the embedded image is still recognizable.

Since the extraction of the embedded data is done on a pixel-by-pixel basis, (part of) the embedded data can still be extracted if the image is cropped or transformed geometrically without modifying the pixel values.

In a paper by Knox [8], a system was proposed to embed an image into another image which share some similarities with the above algorithm. In fact, Knox's approach is similar to a special case of the proposed algorithm with a specific set C. The data embedding and extraction algorithms in Knox's approach are based on reversing the bits of the pixels. This corresponds to a set C of the form $(a, \mathit{reverse}(a))$ where $\mathit{reverse}(a)$ is a with the bits reversed. This set C is shown in Figure 6. Except for the lower-right and upper-left corners, this set covers the square $[0, 255]^2$ rather uniformly. In fact Figure 4a is created from Figure 6 by moving some of the more closely clustered dots to the lower-right and upper-left corners.

One difference between Knox's approach and the proposed algorithm is that in Knox's algorithm two separate 4-bit multilevel error diffusion algorithms are applied independently to the two images and therefore the choice of the output colors can be less ideal than the current algorithm where the error diffusion is applied jointly to the two images by considering pairs of pixel values. Furthermore, the current algorithm has the additional flexibility of having different weights v_u for the different images.

Another feature of the proposed scheme is the flexibility in which the set C can be chosen; C can be any set of the form $(a, f(a))$ and the function f can be chosen depending on the application and requirements. For instance, f can be smooth for more robustness, or f can be chosen so the convex hull of C cover a large portion of $(S^d)^{k+1}$ for small distortion between M_u and $\tilde{M}_u$. Furthermore, the function f can be one-to-one (as is the case for $\mathit{reverse}(a)$) or not (as is the case in quantization index modulation).

4. IMAGE AUTHENTICATION SCHEME

In general, the embedded images can contain ownership information, recipient information, and so forth, which are retrieved in the data extraction process. We now present an application where the embedded information can be used to detect, localize, and repair changes to the modified source image and can be used as part of an authentication system for images. In this application, the embedded image M_1 is a scrambled version of the source image M_0. In particular, M_1 is generated from M_0 by a permutation of the pixels: $M_1(P(i, j)) = M_0(i, j)$ where P is a random permutation of the pixel locations (i, j). Since such a permutation destroys

```
For each iteration                                /* For each iteration*/
  For each $i$                                    /* For each row*/
    For each $j$                                  /* For each column*/
      For each member $d$ of $C$                  /* Search through all possible members of $C$*/
        Set $Ex_{\text{out}}(i,j) = d$
        Compute $v(d) = v_0\|L(Ex^0_{\text{out}} - M_0)\|^2 + v_1\|L(PEx^1_{\text{out}} - M_0)\|^2$
      Endfor
      Set $Ex_{\text{out}}(i,j) = \underset{d}{\operatorname{argmin}}\, v(d)$
    Endfor
  Endfor
Endfor (iteration) or until $Ex_{\text{out}}$ has not changed between two consecutive iterations.
Set modified source image $M_0'$ as the first coordinates of $Ex_{\text{out}}$.
```

Algorithm 2: Pseudocode of data hiding algorithm using iterative halftoning.

the spatial relationship between the pixels, the inverse permutation is applied in the halftoning algorithm to restore the spatial relationship before calculating the error function, that is, the function to minimize is

$$v_0\|L(Ex^0_{\text{out}} - M_0)\|^2 + v_1\|L(PEx^1_{\text{out}} - M_0)\|^2, \qquad (5)$$

where PEx^1_{out} is the image defined by

$$PEx^1_{\text{out}}(i,j) = Ex^1_{\text{out}}(P(i,j)). \qquad (6)$$

Since the spatial relation between the pixels in Ex^0_{out} and Ex^1_{out} is now different, a halftoning algorithm such as error diffusion which relies on a certain order of pixel traversal is not suitable. We will use an isotropic iterative halftoning algorithm instead.

In particular, at each iteration the pixels are traversed in a particular order and for each location (i,j), the pixel $Ex_{\text{out}}(i,j)$ is chosen from the set C, such that (5) is minimized. This is run through several iterations until a local minimum is reached (i.e., no pixel changes between two iterations) or the maximum number of iterations is reached. The pseudocode of this algorithm is shown in Algorithm 2.

Because of the permutation, localized changes to the modified source image result in changes to the embedded images which are spread out throughout the image. This is similar to the interleaving techniques used in error correction codes to combat burst errors in, for example, compact disk recordings. The reconstructed embedded image (after inverse permutation) is compared with the modified source image to check whether significant changes have occurred. This operation can also localize where such changes have occurred. Furthermore, because the changes are spread out throughout the embedded image, the changes to the embedded image will not be as disruptive visually than if the changes occurred in blocks. This allows the corrupted source image to be repaired by using the embedded image. This is illustrated in Figures 7, 8, and 9.

In Figure 7a, the modified source image M_0' is corrupted by erasing a portion of the image and adding text and other changes to the image. Next we extract the embedded image M_1' from this corrupted M_0'. After applying the inverse permutation to M_1' we see that it still resembles M_0, as shown in Figure 7b. In fact, the tampered portions of the image have been recovered to a certain degree. The noise pixels which occur throughout the image correspond to the tampered pixels in the source image which have been spread out by the permutation P.

Define $PM_1'(i,j) = M_1'(P(i,j))$. By low-pass filtering $(M_0' - PM_1')$ and finding the pixels with relatively large norms, an estimate of where the modification occurs can be found. This is shown in Figure 8a, where the black pixels indicate where the modifications to the source image occurred. This estimate can be further refined by morphological operations to remove small clusters of pixels and fill in small holes.

This estimate can also be used to determine where the embedded image should be recovered. If a pixel is corrupted in the modified source image M_0', then this pixel (after applying the permutation) is not recovered in the embedded image PM_1', but interpolated from other pixels which are not corrupted. Using this algorithm with bilinear interpolation results in Figure 8b. We see that much of the noise in Figure 7b has been removed. The residual impulsive noise can be further removed by means of a 3 by 3 median filter (Figure 9).

An alternative algorithm for recovering the source image is as follows. Using the above estimate, for the pixels which were not corrupted in the modified source image, the “repaired” source image pixels are set to the modified source image pixels. For the corrupted pixels, the “repaired” source pixels are recovered from the embedded image using interpolation as before.

To enhance the security of the authentication scheme, the permutation P and the choice of the set C can be generated

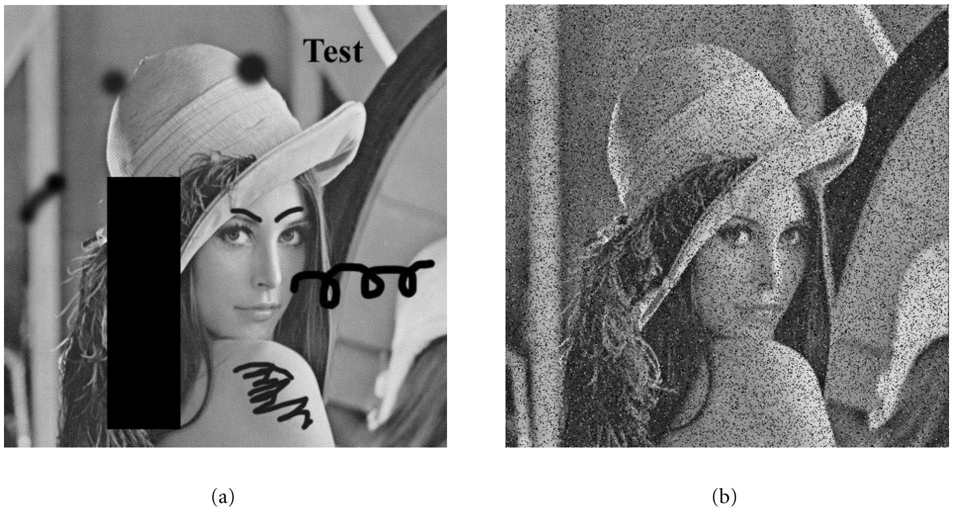


Figure 7: (a) Embedded image M'_0 after tampering. (b) Recovered image.

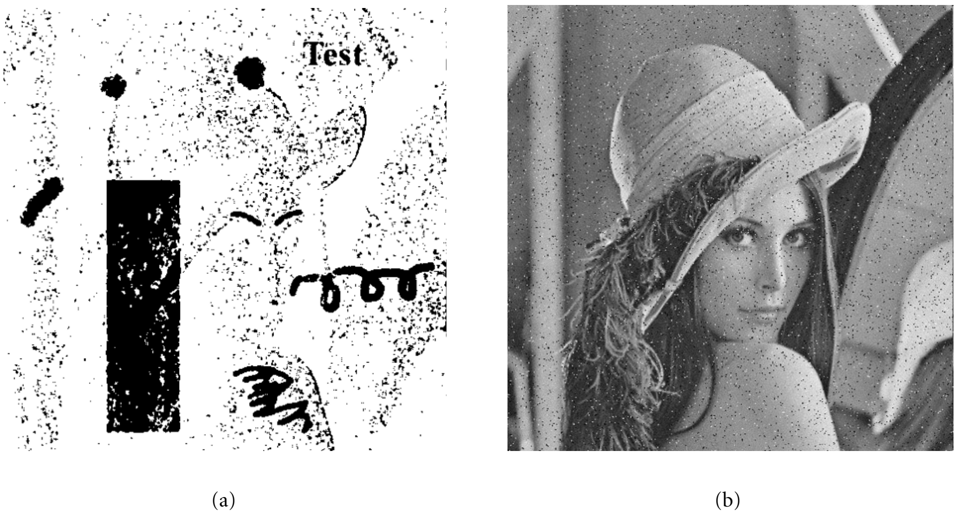


Figure 8: (a) Estimate of where tampering has occurred. (b) Recovered image by interpolating from untampered pixels.

from secret keys. For color images, the size of C can be as large as or larger than 256^3. This could be too large to implement the halftoning algorithm efficiently. In this case, the color image is split into the color planes, and each color plane is processed independently. In other words, an RGB image is considered as 3 grayscale images which are processed independently.

5. CONCLUSIONS

We have proposed a data hiding scheme which utilizes halftoning algorithms to maintain imperceptibility of embedding by minimizing the distortion of the modified source image and of the embedded images. One-to-one coordinate projection is used to ensure that the embedded images can be extracted. Tradeoff between robustness against minor modifications and embedding distortion can be obtained by a proper choice of the extended color set C. The data hiding scheme can be used in an image authentication scheme where tampering can be detected, localized and reversed.

We have considered the coordinates of elements of the set C as values of single pixels. An extension of the algorithm is to consider the case where the coordinates are values of blocks of pixels.

By considering video as a sequence of images, the proposed algorithms can be used to hide data in video streams as well. On the other hand, by considering video as a 3D data set with time as another dimension, spatiotemporal

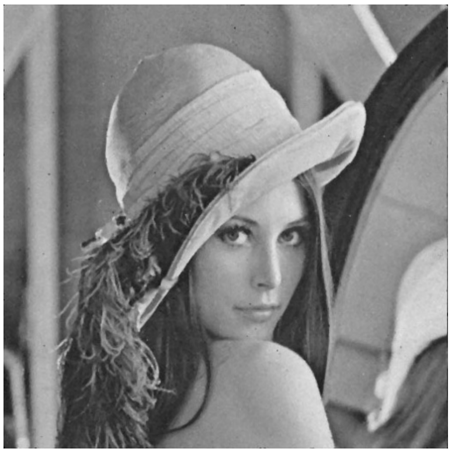


Figure 9: Reconstructed source image after median filtering of Figure 8b.

halftoning can be used to embed data into video streams as a whole.

ACKNOWLEDGEMENT

The author would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—a survey," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1062–1078, 1999.

[2] R. Näsänen, "Visibility of half-tone dot textures," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 14, no. 6, pp. 920–924, 1984.

[3] J. Sullivan, L. Ray, and R. Miller, "Design of minimum visual modulation halftone patterns," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 21, no. 1, pp. 33–38, 1991.

[4] H. Haneishi, T. Suzuki, N. Shimoyama, and Y. Miyaki, "Color digital halftoning taking colorimetric color reproduction into account," *Journal of Electronic Imaging*, vol. 5, pp. 97–106, January 1996.

[5] B. W. Kolpatzik and C. A. Bouman, "Optimized error diffusion for high quality image display," *Journal of Electronic Imaging*, vol. 1, no. 3, pp. 277–292, 1992.

[6] B. Chen and G. W. Wornell, "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1423–1443, 2001.

[7] R. Adler, B. Kitchens, M. Martens, A. Nogueira, C. Tresser, and C. W. Wu, "Error bounds for error diffusion and other mathematical problems arising in digital halftoning," in *IS&T/SPIE Conference on Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts V, Proceedings of SPIE*, vol. 3963, pp. 437–443, January 2000.

[8] K. Knox, "Reversible digital images," in *IS&T/SPIE Conference on Security and Watermarking of Multimedia Contents, Proceedings of SPIE*, vol. 3657, pp. 397–401, January 1999.

Chai Wah Wu received his B.S. and B.A. degrees from Lehigh University, and his M.S., M.A., and Ph.D. degrees from the University of California at Berkeley. He is currently a research staff member at IBM T.J. Watson Research Center in Yorktown Heights, New York. He has written over 40 journal publications and his research interests include image watermarking, multimedia security, digital halftoning, and synchronization of chaos. Dr. Wu is a Fellow of the IEEE and was an associate editor of IEEE Transactions on Circuits and Systems, part I.