Computer Standards & Interfaces 36 (2014) 855–862

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A data hiding scheme based upon DCT coefficient modification

Yih-Kai Lin *

Department of Computer Science, National Pingtung University of Education, No. 4-18 Minsheng Rd., Pingtung City 90003, Taiwan

ARTICLE INFO

Article history:
Received 2 March 2012
Received in revised form 30 August 2013
Accepted 23 December 2013
Available online 18 January 2014

Keywords:
Information hiding
DCT
Secret communication
LSB substitution

ABSTRACT

In this paper, we propose a steganographic scheme based on the varieties of coefficients of discrete cosine transformation of an image. The major problem of hiding data in the high-frequency coefficients of discrete cosine transformation is that rounding errors will be added into the spatial-domain image, and thus cannot be transformed back to the correct modified coefficients of the discrete cosine transformation. To solve this problem, we use integer mapping to implement our discrete cosine transformation. Thus, the image recovered from the modified coefficients can be transformed again to the correct data hidden coefficients.

1. Introduction

Steganography [1,28,10,17] is the process of hiding data within other readable data to avoid that unreadable data attracting the attention of eavesdroppers. Many data hiding approaches have been proposed. These approaches can be classified into two categories: the spatial-domain and the frequency-domain categories. In the spatial-domain, data hiding technologies directly manipulate the pixel value for hiding data. For example, the LSB substitution replaces the length-fixed LSB of a pixel with the fixed length secret bits. Simple steganography involves hiding the secret message in the least-significant-bit (LSB) of the pixels of an image [4,11,15]. It is easy to use the simple LSB substitution method to create a noticeable distortion for the human eye because a little variant of smooth area can be noted by humans. The LSB substitution method is also easily detected by programs. For example, Fridrich et al. [7] use the *RS-diagram* to detect LSB steganography. Thus, Wu and Tsai [26] proposed a steganographic method using the difference value between two adjacent pixels to avoid detection by the RS-diagram.

Therefore, several approaches have been used to adjust the position and amount of embedding secret data in order to decrease the noticeable distortion. Recently, Yang et al. [27] proposed a pixel-value differencing (PVD) steganographic method that modifies the difference value between two adjacent pixels so that more secret data is embedded into two pixels located in the edge area than in the smooth area. Yang et al. [27] extended the technique for embedding data in an image using pixel-value differencing. Their approach provides high capacity and avoids detection by the RS-diagram.

Traditionally, frequency-domain processing has been a counterpart of the spatial-domain technique in signal filtering, pattern recognition, and image compression. An intricate problem in the spatial-domain approach sometimes becomes easy to deal with in the frequency-domain approach. For example, in the spatial domain, it is hard to find the redundant part of an image for lossy image compression, but this redundant part can be collected at high-frequency content when the image is transformed to frequency domain. Intuitively, edges and sharp transitions in an image contribute significantly to the high-frequency content of its discrete cosine transform. Thus, to look for the proper pixels for hiding data, transform-based schemes are reasonable approach. In these schemes, cover images are transformed by some frequency-oriented mechanism, e.g., *discrete cosine transform* (*DCT*) and *discrete wavelet transform* (*DWT*). Then, the secret data are embedded into the frequency-form images by modifying the coefficients.

In this paper, we propose a steganographic scheme based on the ability of human vision which is insensitive to the varieties of high-frequency components of an image. To improve the quality of the stego-image, we utilize the concept of decomposition of images. In the proposed method, images are decomposed into several different frequencies, and the high-frequency parts contain hidden data. A similar idea has been used to compress images in JPEG. In JPEG compression, the information loss due to fixed-precision data storing and computing in the domain transformation stage and also in the quantization stage creates a huge compression rate.

The major problem of hiding data in DCT coefficients is that when we transform the modified DCT coefficients back to the spatial-domain, because the pixels of the image are stored in integer form, rounding errors will be added into the spatial-domain image, and thus cannot be transformed back to the correct modified DCT coefficients when extracting the embedded data. For an illustration of this problem, see Fig. 1.

* Tel.: +886 8 722 6141x33559; fax: +886 8 721 5034.
E-mail address: yklin@mail.npue.edu.tw.

0920-5489/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.csi.2013.12.013

To solve this problem, we use one of the forms of integer mapping [9,18] to implement our DCT transformation which maps integer to integer. Thus, if the modified DCT coefficients remain as integers, the image recovered from the modified coefficient can be transformed again to the correct modified coefficient, and the embedded data can be extracted.

2. Related work

Yang et al. [27] proposed a pixel-value differencing steganographic method that modifies the difference value between two adjacent pixels so that more secret data is embedded into two pixels located in the edge area than in the smooth area. Their method partitions the host image into four-pixel blocks. The four pixels are called p_1, p_2, p_3, and p_4, and the corresponding gray values are called g_1, g_2, g_3 and g_4. The four-pixel block is partitioned into two two-pixel groups (g_1,g_4) and (g_2,g_3). The two group differences are computed as $d_1 = (g_4 - g_1)$ and $d_2 = (g_3 - g_2)$. The possible absolute values of d are classified into a number of contiguous ranges R_k where $k = 1, 2, 3, \ldots, r$. The width of R_k is $u_k - l_k + 1$ where u_k is the upper bound of R_k and l_k is the lower bound of R_k. The secret message is embedded by replacing the difference values of the two groups in a block. The new differences can be computed by $d_i^{'} = l_{k_i} + b_i$ where $i = 1, 2$, b_1 and b_2 are the decimal values of S_1 and S_2. This approach embeds secret data in some edge areas.

Lin et al. [16] present a data hiding scheme based on a transformation concept. Their data hiding strategy embeds secret information into quantized high-frequency components. The key idea behind [16] is to hide secret data in two neighboring coefficients. They adopt Tian's pixel expansion method [22] to hide data. A non-zero DCT coefficient and a zero DCT coefficient $(x,0)$ are selected as the embedding pair. If an embeddable pair is determined, Tian's method [22] is used to embed a secret bit in the embedding pair. Since each coefficient pair is not always a $(x,0)$ pair, the disadvantage of Lin et al.'s [16] method is that it can embed less capacity when compared with the spatial-domain method, for example, Ref. [27].

In 2002, Chang et al. [5] presented a steganographic method based on the JPEG format. They modify the quantization table to get quantized DCT coefficients and to hide the secret message in the least-two-significant bits of the middle-frequency of the quantized DCT coefficients. To improve the embedding capacity, Tseng and Chang [23] proposed a high capacity hiding method based on the adaptive least-significant bit substitution method. Later, Chang et al. [6] presented a reversible hiding scheme for hiding secret data in quantized DCT coefficients. In 2010, Chia-Chen Lin and Pei-Feng Shiu [16] extended Chang et al.'s idea [6,5] and presented a high capacity data hiding strategy that also embeds data into the coefficients in the middle frequency of the quantized DCT coefficients. That is, Chang et al., Tseng et al. and Lin et al. [5,6,23,16] utilize the redundancy in the quantized DCT coefficients. In other words, their method modifies the quantization stage of JPEG compression and stores the stego-image in JPEG format.

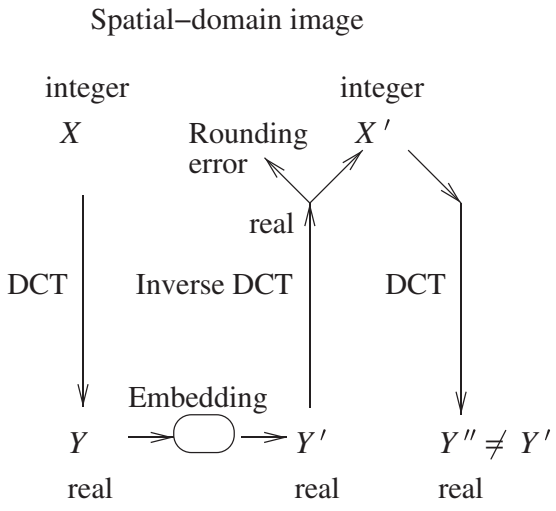

Fig. 1. Problem caused by rounding error in directly embedding secret data in DCT coefficients: modified DCT coefficients $\boldsymbol{Y}'$ cannot be restored from stego-image $\boldsymbol{X}'$.

Also, transform-based steganographic methods F5 [25], nsF5 [8], YASS [21], and recently proposed adaptive steganography [14] all embed data into the quantized DCT coefficients but our method directly embeds secret message into the DCT coefficients of the cover image. Thus, our stego-image stays in its raw image form instead of some special form of quantized coefficients e.g., JPEG-formatted. Since the quantized DCT coefficients are encoded by entropy coding in JPEG format, it is hard to hide data without increasing the size of stego-image. But our method directly embeds message into the coefficients and keeps stego-image in raw image and in the same file size. Furthermore, in order to deal with the errors caused in the image due to JPEG compression, a repeat-accumulate (RA) code (error-correcting codes) is used in YASS. On the other hand, our method avoids quantized error and error-correcting code.

In 2010, Qi et al. [20] proposed an integer inverse discrete cosine transformation (IDCT). They use the common factor extraction algorithm and the two-stage scale algorithm to achieve a high-accuracy (but not error-free) integer IDCT. Wahid et al. [24] use an algebraic integer encoding technique which eliminates the requirements to approximate the transformation matrix elements. An algebraic integer is a complex number that is roots of monic polynomials with integer coefficients. They use the algebraic integer to exactly represent the elements of the transformation matrix without any errors during transformation. However, in the last conversion step, some rounding errors can be generated by converting the algebraic integer numbers to fixed-precision number.

3. Preliminaries

An $M \times N$ digital image is defined by function $f: (\mathbb{N},\mathbb{N}) \mapsto \mathbb{N}$ where $\mathbb{N}$ denotes the set of nature numbers. For convenience, a traditional matrix is used to denote a digital image: $\boldsymbol{X} = [x_{i,j}]$ where $x_{i,j} = f(i,j)$ for $1 \le i \le M$ and $1 \le j \le N$. A real number b can be rounded to an integer (denoted by angled brackets $\langle\cdot\rangle$) $\langle b\rangle$ by rounding-down, rounding-up, or some other methods. Given an $N \times N$ matrix $\boldsymbol{X}$, the kth order *leading principal* of $\boldsymbol{A}$ is a sub-matrix formed by deleting the last N–k columns, say column $k + 1, k + 2, \ldots, N$ and the last N–k rows, row $k + 1, k + 2, \ldots, N$. The kth order *leading principal minor* of $\boldsymbol{A}$ is the determinant of a $k \times k$ principal sub-matrix. The data needing to be hidden are called the *secret data* and are a sequence of bits denoted by $\mathfrak{M}$. The *cover image* is the image for carrying the hidden secret data. The *stego-image* is the image containing the hidden secret data. The *image quality* refers to the similarity of the cover image and the stego-image. The *capacity* refers to the total bits of hidden secret data.

4. The proposed method

In this section, we present the proposed steganographic scheme based on discrete cosine transformation. We first transform a given cover image $\boldsymbol{X}$ into a sequence of DCT coefficients $\boldsymbol{Y}$, then present a hiding algorithm to embed the secret data into $\boldsymbol{Y}$, benefiting from frequency representation of images.

In our approach, the secret data $\mathfrak{M}$ is directly embedded in the high-frequency coefficient part of $\boldsymbol{Y}$ for producing the modified frequency-domain image $\boldsymbol{Y}'$. Then the modified frequency-domain image $\boldsymbol{Y}'$ is transformed back to the spatial-domain to get the stego-image $\boldsymbol{X}'$. Because the general DCT coefficients are in the real domain, one major problem is to ensure that the secret data can be extracted when the stego-images are stored in a small range of integers, e.g., 0 to 255. That is, with the unavoidable error in floating-point number calculation

with fixed-precision computing architecture, the image $\boldsymbol{X}'$ in the spatial-domain needs to be transformed exactly back to $\boldsymbol{Y}'$. Therefore, the secret data $\mathfrak{M}$ can be extracted from $\boldsymbol{Y}'$. Our proposed scheme is shown in Fig. 2.

4.1. Reversible integer mapping

As mentioned above, cover image $\boldsymbol{X}$ is transformed into coefficient matrix $\boldsymbol{Y} = \boldsymbol{AX}$ by transformation matrix $\boldsymbol{A}$. Based on the theorems from Ref. [9], matrix $\boldsymbol{A}$ has a factorization $\boldsymbol{PS}_N\boldsymbol{S}_{N-1}\cdots\boldsymbol{S}_0$ if and only if the minors of the leading principal sub-matrices of $\boldsymbol{A}$ are all 1s, where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{S}_m = \boldsymbol{I} + \boldsymbol{e}_m\boldsymbol{s}_m^T$ for $m = 1, 2, 3, \ldots, N$, $\boldsymbol{S}_0 = \boldsymbol{I} + \boldsymbol{e}_N\boldsymbol{s}_0^T$, e_m is the mth column vector of the identity matrix, $\boldsymbol{s}_m$ is a vector whose mth element is 0, and $\boldsymbol{I}$ is the identity matrix. Consider an integer column vector $\boldsymbol{x}$ of $\boldsymbol{X}$; the linear transformation $\boldsymbol{Ax} = \boldsymbol{y}$ can be computed by $\boldsymbol{PS}_N\boldsymbol{S}_{N-1}\cdots\boldsymbol{S}_1\boldsymbol{S}_0\boldsymbol{x} = \boldsymbol{y}$ where y is a column vector of $\boldsymbol{Y}$. Consider the linear transformation $<\boldsymbol{S}_m\boldsymbol{x}> = \boldsymbol{y}$ for $m = 0, 1, 2, 3, \ldots, N$,

$$<\boldsymbol{S}_m\boldsymbol{x}> = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ s_{m,1} & \cdots & 1 & \cdots & s_{m,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \\ \vdots \\ x_N \end{bmatrix}$$
$$= \begin{bmatrix} x_1 \\ \vdots \\ \left\langle \sum_{i=1}^{m-1} s_{m,i}x_i \right\rangle + x_m + \left\langle \sum_{i=m+1}^{N} s_{m,i}x_i \right\rangle \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \\ \vdots \\ y_N \end{bmatrix} = \boldsymbol{y}$$

where $<>$ denotes integer conversion. The reverse transformation is

$$<\boldsymbol{S}_m^{-1}\boldsymbol{S}_m\boldsymbol{x}> = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ -s_{m,1} & \cdots & 1 & \cdots & -s_{m,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \\ \vdots \\ y_N \end{bmatrix}$$
$$= \begin{bmatrix} y_1 \\ \vdots \\ -\left\langle \sum_{i=1}^{m-1} s_{m,i}x_i \right\rangle + y_m + -\left\langle \sum_{i=m+1}^{N} s_{m,i}x_i \right\rangle \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \\ \vdots \\ x_N \end{bmatrix} = \boldsymbol{x}. \quad (1)$$

Because $y_i = x_i$ for $i \neq m$, the fact

$$<\sum_{i=1}^{m-1} s_{m,i}y_i> + <\sum_{i=m+1}^{N} s_{m,i}y_i> = <\sum_{i=1}^{m-1} s_{m,i}x_i> + <\sum_{i=m+1}^{N} s_{m,i}x_i>$$

deduces the Eq. (1) — a reversible integer linear transformation. Notice that y_i and x_i must keep precision in store and computation. With a fixed-precision computer, keeping y_i and x_i in integer form is a way to preserve reversible linear transformation. In summary, the input signals are rearranged into vector $\boldsymbol{x}$ and yield $\boldsymbol{y}_0 = \boldsymbol{S}_0\boldsymbol{x}$. Then $\boldsymbol{y}_0$ is rounded to an integer and used to get $\boldsymbol{y}_1$ as $\boldsymbol{y}_1 = \boldsymbol{S}_1<\boldsymbol{y}_0>$ and so on. Finally, $\boldsymbol{y} = <\boldsymbol{P}<\boldsymbol{S}_N<\cdots<\boldsymbol{S}_1<\boldsymbol{S}_0<\boldsymbol{x}>>>\cdots>>$ and $\boldsymbol{x} = <\boldsymbol{S}_0^{-1}<\cdots<\boldsymbol{S}_{N-1}^{-1}<\boldsymbol{S}_N^{-1}<\boldsymbol{P}^{-1}<\boldsymbol{y}>>>>\cdots>$.

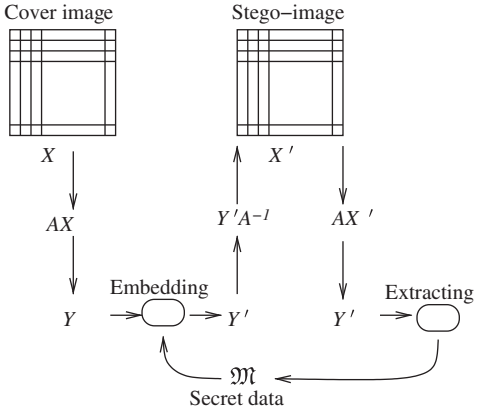

Fig. 2. Scheme for embedded secret data.

4.2. Decomposing the matrix of one dimension discrete cosine transformation

The discrete cosine transformation is a function $g : \mathbb{R}^N \mapsto \mathbb{R}^N$ where $\mathbb{R}$ denotes the set of real numbers. The N real numbers $x_1, x_2, \ldots x_N$ are maps to the N real numbers $y_1, y_2, \ldots y_N$ according to the formula

$$y_k = \sum_{n=1}^{N} x_n \frac{1}{2}\cos\left(\frac{\pi}{N}\left(n-1+\frac{1}{2}\right)(k-1)\right) \qquad \text{for } k = 1, \ldots, N.$$

The discrete cosine transformation is rearranged to a matrix form $\boldsymbol{C}$ such that $\boldsymbol{y} = \boldsymbol{Cx}$. For example, the 8×8 discrete cosine transformation matrix is

$$\mathbf{C} = \begin{bmatrix} \frac{1}{2}\cos\left(\frac{\pi}{8}\left(0+\frac{1}{2}\right)0\right) & \cdots & \frac{1}{2}\cos\left(\frac{\pi}{8}\left(N-1+\frac{1}{2}\right)0\right) \\ \vdots & \ddots & \vdots \\ \frac{1}{2}\cos\left(\frac{\pi}{8}\left(0+\frac{1}{2}\right)(7)\right) & \cdots & \frac{1}{2}\cos\left(\frac{\pi}{8}\left(7+\frac{1}{2}\right)(7)\right) \end{bmatrix}.$$

Because $\det \boldsymbol{C} = \sqrt{2}$, each element of the first row of $\boldsymbol{C}$ is divided by the constant $\sqrt{2}$ to satisfy $\det \boldsymbol{C} = \pm 1$. The modified matrix, denoted $\boldsymbol{A}$, is used as the discrete cosine transformation matrix in the remaining part of this paper. Matrix $\boldsymbol{A}$ is decomposed into $\boldsymbol{PS}_N\boldsymbol{S}_{N-1}\cdots\boldsymbol{S}_1\boldsymbol{S}_0$ such that

$$\begin{aligned}
\boldsymbol{s}_0 &= [1.0392 \quad -1.7683 \quad -3.6338 \quad -6.9117 \quad 0.5430 \quad 2.7891 \quad 6.5711 \quad 0.0000], \\
\boldsymbol{s}_1 &= [0.0000 \quad -0.4514 \quad -1.5042 \quad -3.2919 \quad 0.1687 \quad 1.0900 \quad 2.8067 \quad -0.4904], \\
\boldsymbol{s}_2 &= [-0.0181 \quad 0.0000 \quad 1.4600 \quad 2.6713 \quad -0.7097 \quad -1.4600 \quad -2.7933 \quad 0.4531], \\
\boldsymbol{s}_3 &= [0.5533 \quad -0.7259 \quad 0.0000 \quad 2.2995 \quad -0.8757 \quad -1.0000 \quad -1.3017 \quad 0.3289], \\
\boldsymbol{s}_4 &= [0.8777 \quad -0.5000 \quad -0.0689 \quad 0.0000 \quad -0.0192 \quad -0.0000 \quad -0.8966 \quad 0.2265], \\
\boldsymbol{s}_5 &= [1.0938 \quad -0.5098 \quad 0.7775 \quad -1.5860 \quad 0.0000 \quad 0.0000 \quad -0.9142 \quad 0.2310], \\
\boldsymbol{s}_6 &= [-0.9590 \quad 1.4863 \quad -0.6172 \quad 1.3194 \quad 1.0214 \quad 0.0000 \quad 1.7071 \quad -0.3266], \\
\boldsymbol{s}_7 &= [-2.3370 \quad 2.1020 \quad -0.3173 \quad 2.6973 \quad 1.2493 \quad -0.4142 \quad 0.0000 \quad -0.4619], \\
\boldsymbol{s}_8 &= [-5.9591 \quad 4.9646 \quad 0.3747 \quad 6.0503 \quad 3.3059 \quad -0.1313 \quad -2.9302 \quad -2.0000]
\end{aligned}$$

and

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

where $\boldsymbol{S}_m = \boldsymbol{I} + \boldsymbol{e}_m\boldsymbol{s}_m^T$ for $m = 1, 2, 3, \ldots, N$, $\boldsymbol{S}_0 = \boldsymbol{I} + \boldsymbol{e}_N\boldsymbol{s}_0^T$, $\boldsymbol{e}_m$ is the mth column vector of the identity matrix, $\mathbf{s}_m$ is a vector whose mth element is 0, and $\boldsymbol{I}$ is the identity matrix.

4.3. Embedding secret data into an image

The input cover image is divided into an 8×8 sub-image, denoted $\boldsymbol{X}$. Each column $\boldsymbol{x}$ of $\boldsymbol{X}$ is transformed into the frequency-domain coefficients $\boldsymbol{y} = \boldsymbol{Ax}$. Eight columns of $\boldsymbol{X}$ are transformed together by the matrix form

$$\boldsymbol{Y} = \begin{bmatrix} | & \cdots & | \\ \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_8 \\ | & \cdots & | \end{bmatrix} = \boldsymbol{A}\begin{bmatrix} | & \cdots & | \\ \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_8 \\ | & \cdots & | \end{bmatrix} = \boldsymbol{AX}.$$

The frequency-domain $\boldsymbol{Y}$ is computed step by step by $\boldsymbol{Y} = <\boldsymbol{P}<\boldsymbol{S}_N<\cdots<\boldsymbol{S}_1<\boldsymbol{S}_0<\boldsymbol{X}>>>\cdots>$ and reverse transformation is computed by $\boldsymbol{X} = <\boldsymbol{S}_0^{-1}<\cdots<\boldsymbol{S}_{N-1}^{-1}<\boldsymbol{S}_N^{-1}<\boldsymbol{P}^{-1}<\boldsymbol{Y}>>>\cdots>$. For example, take an 8×8 sub-image $\boldsymbol{X}_L$ of image 'Lena' shown as follows

$$\boldsymbol{X}_L = \begin{bmatrix} 111 & 110 & 114 & 112 & 116 & 109 & 113 & 113 \\ 112 & 106 & 114 & 114 & 121 & 108 & 111 & 112 \\ 111 & 114 & 115 & 110 & 108 & 109 & 106 & 105 \\ 112 & 113 & 112 & 106 & 114 & 112 & 105 & 112 \\ 110 & 109 & 111 & 107 & 110 & 111 & 114 & 113 \\ 108 & 115 & 110 & 108 & 104 & 110 & 108 & 117 \\ 109 & 108 & 108 & 108 & 113 & 112 & 115 & 110 \\ 111 & 113 & 108 & 109 & 111 & 110 & 110 & 110 \end{bmatrix},$$

the correspondence frequency-domain sub-image is

$$\boldsymbol{Y}_L = \begin{bmatrix} 312 & 314 & 315 & 308 & 318 & 311 & 312 & 315 \\ 2 & -2 & 7 & 5 & 7 & -2 & -2 & -1 \\ 0 & -3 & -1 & 4 & 6 & -2 & 4 & -1 \\ -2 & -1 & -1 & 1 & -2 & 1 & 5 & 7 \\ 2 & 1 & -1 & -2 & 2 & 1 & 1 & 0 \\ 0 & 2 & 0 & -2 & -1 & 2 & -1 & -2 \\ -2 & 7 & 2 & -3 & -9 & -3 & -5 & 1 \\ -1 & -2 & 1 & -1 & -2 & -1 & 5 & -5 \end{bmatrix} = <\boldsymbol{P}<\boldsymbol{S}_N<\cdots<\boldsymbol{S}_1<\boldsymbol{S}_0<\boldsymbol{X}_L>>>\cdots>.$$

The lower elements of $\boldsymbol{Y}$ are the lower frequencies component of the sub-image $\boldsymbol{X}$. Thus, the last p rows are used to embed the secret data. Given an element $y_{i,j}$ of $\boldsymbol{Y}$, the q least significant bits of $y_{i,j}$ are replaced by the secret data. The formal embedding algorithm is shown in algorithm DCT-HIDING and the formal extracting algorithm is shown in algorithm DCT-EXTRACTION.

DCT-HIDING($\boldsymbol{P}\boldsymbol{S}_8\cdots\boldsymbol{S}_0$, $\boldsymbol{I}$, p, q, $\mathfrak{M}$)

Input:

$\boldsymbol{P}\boldsymbol{S}_8\cdots\boldsymbol{S}_0$: a factorization of $\boldsymbol{A}$

$\boldsymbol{I}$: input cover image

p: the parameter for adjusting the rows of embedded data

q: the parameter for adjusting numbers of least significant bits of embedded data

$\mathfrak{M}[\,]$: the secret data stream

Output:

the stego-image corresponding to the input image

begin

```
for each 8 × 8 sub-image X of I /* divide input cover image in 8 × 8 blocks */
    T ← S_0 X      /* save result in temporary storage */
    T ← ⟨T⟩        /* round temporary result to integer */
    for i = 1 to 8
        T ← S_i T
        T ← ⟨T⟩
    Y ← PT
    for each element y_{i,j} of Y
        if (i > (8 − p))      /* if it's in last p row, embed data in it */
            take q bits from M[ ] and embed it in least q bits of y_{i,j}
    T ← P^{-1} Y      /* start to transform coefficient back to image */
    for i = 8 downto 0
        T ← S_i^{-1} T
        T ← ⟨T⟩
    X' ← ⟨T⟩
    X ← X'   /* replace original block X with modified one X' */
return I      /* I is now stego-image */
```

end

DCT-EXTRACTION($\boldsymbol{P}\boldsymbol{S}_8\cdots\boldsymbol{S}_0$, $\boldsymbol{I}$, p, q)

Input:

$\boldsymbol{P}\boldsymbol{S}_8\cdots\boldsymbol{S}_0$: a factorization of $\boldsymbol{A}$

$\boldsymbol{I}$: input stego-image

p: the parameter for adjusting the rows of embedded data

q: the parameter for adjusting numbers of least significant bits of embedded data

Output:

$\mathfrak{M}[\,]$: the secret data stream

begin

```
for each 8 × 8 sub-image X of I  /* divide input stego-image in 8 × 8 blocks */
    T ← S_0 X      /* save result in temporary storage */
    T ← ⟨T⟩        /* round temporary result to integer */
    for i = 1 to 8
        T ← S_i T
        T ← ⟨T⟩
    Y ← PT
    for each element y_{i,j} of Y
        if (i > (8 − p))      /* if it's in last p row, extract data from it */
            extract q bits from the least q bits of y_{i,j} and insert it into M[ ]
return M
```

end

When the partial frequency coefficients of a cover image are modified, it is possible to get a value overflow or underflow of the normal range for representing a pixel in the corresponding spatial domain. To avoid this problem, we skip the blocks, denoted *skipped blocks*, without embedding information if their corresponding spatial-domain blocks contain a pixel i where $0 \le i \le 255$. To fully recover the original image, we record the indices of skipped blocks and embed these indices into the image as a part of the secret bit.

5. Experiment and comparison

The proposed steganographic scheme is used to embed secret data into the set A, B, and C images — selected images from the USC-SIPI Image Database and BOSSbase v0.92 images [19] for evaluating the practical performance of the proposed steganographic scheme. For comparison with the images mentioned in the literature, set A includes six test images of size 512×512, namely 'Lena', 'Baboon', 'Peppers', 'Boat', 'Goldhill', and 'Tiffany'. The test set B contains 214 images which are adopted from the USC-SIPI Image Database and are divided into four volumes named 'Textures', 'Aerials', 'Misc', and 'Sequences'. The images in each volume are of various sizes in the original USC-SIPI Image Database but are resized to 512×512 for our evaluation experiments. The sizes of the four volumes are shown in column 2 of Table 2. The original images of USC-SIPI, divided into sets A and B, are stored in lossless TIFF format scanned from printed paper. As a large test set, the original 9074 images of BOSSbase v0.92 database [19], called set C, are never-compressed raw images taken from cameras. All experiments are performed on a personal computer with an Intel 3.2 GHz Pentium 4 CPU with cache size 2048 KB.

We use the library Performance Application Programming Interface (PAPI) [2], which can be downloaded from http://icl.cs.utk.edu/papi, to count running time of our integer DCT in clock cycles. The PAPI function PAPI_get_real_cyc is used to count the execution cycles of factorization stage and transformation stage of our DCT transformation. The transformation matrix $\boldsymbol{A}$ is factorized into $\boldsymbol{P}, \boldsymbol{S}_N, \cdots, \boldsymbol{S}_0$ in factorization stage which is performed only one time during embedding secret data into a cover image. And in the transformation stage, computation $<\boldsymbol{P}<\boldsymbol{S}_N<\cdots<\boldsymbol{S}_1<\boldsymbol{S}_0<\boldsymbol{X}>>>\cdots>$ is performed for each $N \times N$ blocks. The 10,000 different random images are used to get the average running time described below.

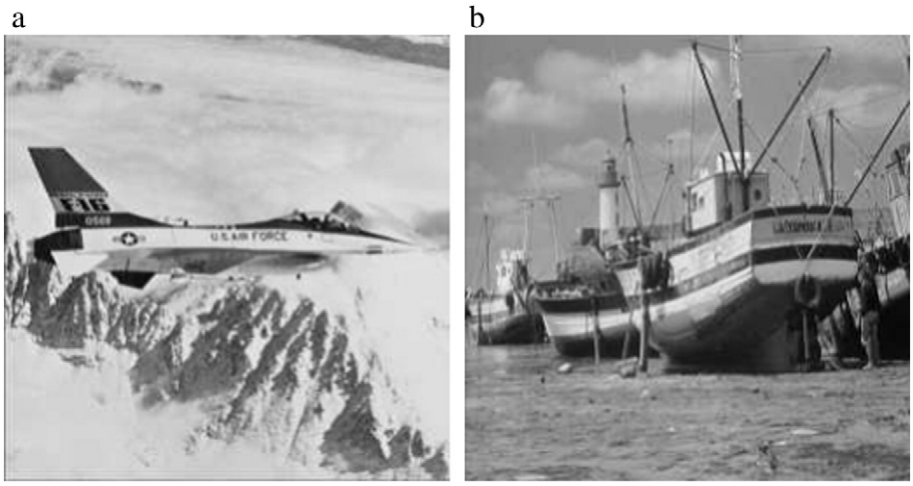

Fig. 3. Two secret images: (a) Airplane. (b) Boat256.

The factorization stage costs 113,695 clock cycles and the transformation stage costs 9478 clock cycles. For comparison with our method, the running time of floating-point version dct implemented in an open source computer vision library OpenCV [3] is 1502 clock cycles. Compared with the floating-point version, the additional cost of our method comes from rounding processes and matrix multiplications.

The proposed scheme is implemented to embed the random bit streams, image 'Boat256' and image 'Airplane' into the cover images 'Lena', 'Baboon', 'Peppers', 'Boat', 'Goldhill', and 'Tiffany', respectively. The images 'Boat256' and 'Airplane' are size 256 × 256 gray-scale images, as shown in Fig. 3. The six cover images 'Lena', 'Baboon', 'Peppers', 'Boat', 'Goldhill', and 'Tiffany' are size 512 × 512 gray-scale images, as shown in Fig. 4. The random bit streams, image 'Boat256' and image 'Airplane' are also embedded into each image in the four volumes of set B. The images in set B are all normalized to 512 × 512 gray-scale. The four instances chosen from each group of set B are shown in Fig. 5.

Table 1 compares the capacities, numbers of skipped blocks (denoted #SBs), and *peak signal-to-noise ratios* (PSNRs) of embedding the secret data into test set A with different values of p and q. Column 1 shows the name of the test files, column 2 shows the size of the original test files, and column 3 shows the items of comparison. Column 4, entitled "$p = 2, q = 7$", refers to the proposed method where $p = 2$ and $q = 7$. The first sub-column of column 4, entitled "Random bits", shows the capacities and PSNRs of embedding random bit streams into the test images. The 100 different random bit streams are used to get the average PSNRs shown here. The PSNRs shown here measure the differences between the cover images and the stego-images. For the two $M \times N$ gray-scale images U and V, the PSNR is computed as $10 \times \log_{10}\frac{255^2}{\mathrm{MSE}}$ where $\mathrm{MSE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(U_{i,j} - V_{i,j}\right)^2$ is the *mean squared error*. The second sub-column, denoted 'Boat256', of column 4, shows the capacities and PSNRs of embedding 'Boat256' into the test images. The third sub-column, denoted 'Airplane', of column 4, shows the

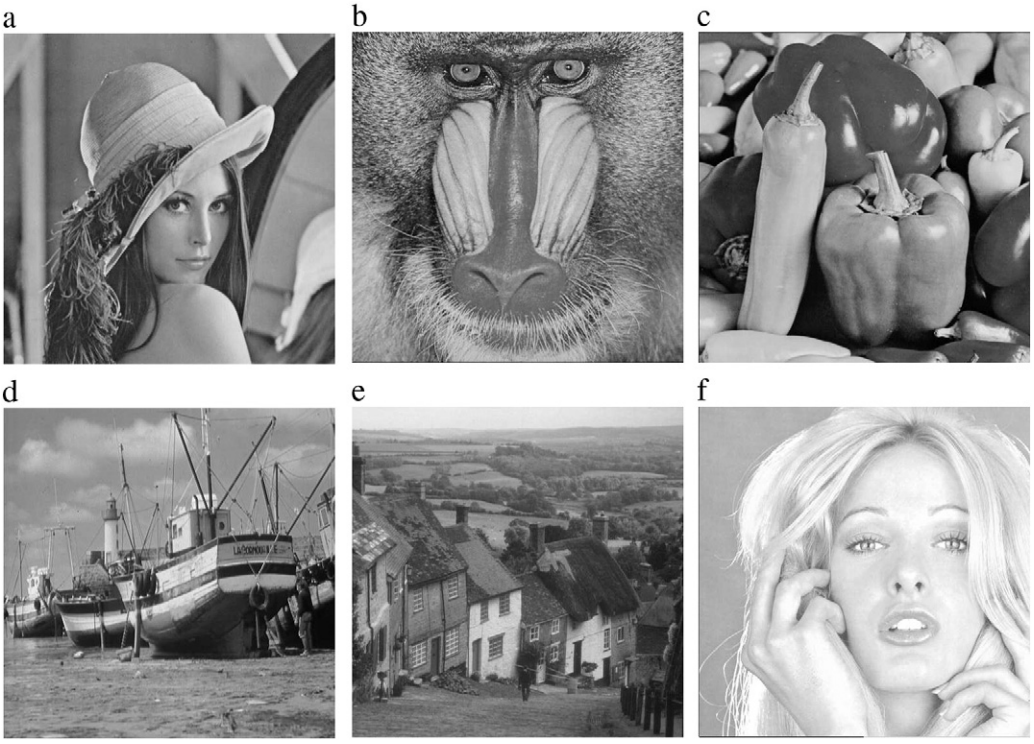

Fig. 4. Six test images of test set A: (a) Lena. (b) Baboon. (c) Peppers. (d) Boat. (e) Goldhill. (f) Tiffany.

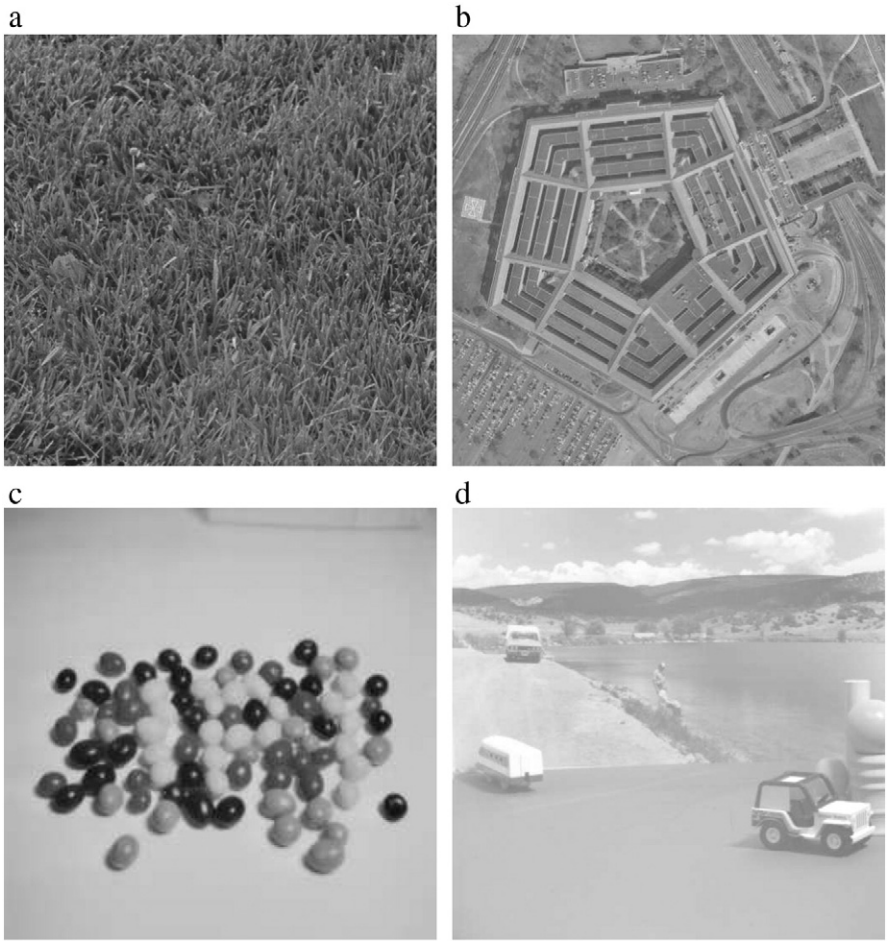

Fig. 5. Four instances from four volumes of test set B: (a) Textures. (b) Aerials. (c) Misc. (d) Sequences.

capacities and PSNRs of embedding 'Airplane' into the test images. Column 5, entitled "$p = 3\ q = 6$", refers to the proposed method where $p = 3$ and $q = 6$. The first sub-column of column 5, entitled "Random bits", shows the capacities and PSNRs of embedding random bit streams into the test images. The 100 different random bit streams are used to get the average PSNR shown here. The second sub-column, denoted 'Boat256', of column 5, shows the capacities and PSNRs of embedding 'Boat256' into the test images. The numbers of blocks skipped, denoted as *#SBs*, without embedding secret bits is shown for each image.

Table 1 shows that parameter p has a major effect on the quality of the stego-image and the capacity. Intuitively, $p = 2, q = 7$ is embedding 2 bits in each coefficient of the last 7 diagonals for every 8×8 blocks. As

Table 1
The capacities and PSNRs for embedding random bit streams using our approach with different p and q values (The 100 different random bit streams are used to get the average PSNRs shown here. The number of skipped blocks is denoted by #SB).

File name	Size		$p = 2, q = 7$			$p = 3, q = 6$		
			Random bits	Boat256	Airplane	Random bits	Boat256	Airplane
Lena	512 × 512	Capacity	458,752	458,752	458,752	589,824	589,824	589,824
		PSNR	43.979	44.302	43.868	38.856	39.079	39.028
		#SB	0	0	0	0	0	0
Baboon	512 × 512	Capacity	457,736	458,080	457,744	586,897	586,800	587,520
		PSNR	43.989	44.300	43.865	38.964	39.189	39.147
		#SB	9.07	6	9	20.32	21	16
Peppers	512 × 512	Capacity	452,085	452,480	451,360	553,438	554,112	554,832
		PSNR	44.030	44.317	43.907	39.170	39.395	39.336
		#SB	59.52	56	66	252.68	248	243
Boat	512 × 512	Capacity	458,423	458,528	458,416	588,674	588,960	588,816
		PSNR	43.974	44.270	43.866	38.917	39.141	39.101
		#SB	2.93	2	3	7.98	6	7
Goldhill	512 × 512	Capacity	458,752	458,752	458,752	589,824	589,824	589,824
		PSNR	43.936	44.269	43.806	38.907	39.119	39.064
		#SB	0	0	0	0	0	0
Tiffany	512 × 512	Capacity	446,089	447,104	446,432	570,108	570,960	569,952
		PSNR	44.0874	44.386	43.950	38.964	39.193	39.124
		#SB	113.06	104	110	136.91	131	138

Table 2
The capacities and PSNRs for embedding random bit streams using our approach with different p and q values (The 100 different random bit streams are used to get the average PSNRs shown here).

Volume	#		$p=2, q=7$			$p=3, q=6$		
name	images		Random bits	Boat256	Airplane	Random bits	Boat256	Airplane
Textures	64	Capacity	425,954	426,914	425,895	528,574	529,755	529,893
		PSNR	44.443	44.757	44.311	39.569	39.779	39.721
Aerials	38	Capacity	458,735	458,737	458,746	589,431	589,490	589,346
		PSNR	43.949	44.259	43.821	38.771	39.001	38.936
Misc	43	Capacity	436,237	436,565	435,914	556,111	543,658	556,205
		PSNR	44.509	44.806	44.243	38.379	39.812	37.944
Sequences	69	Capacity	455,514	455,546	455,453	584,474	584,617	584,531
		PSNR	43.898	44.234	43.565	38.619	38.863	38.750
Boss	9074	Capacity	422,499	423,284	422,176	530,892	531,796	531,112
		PSNR	44.229	44.563	44.093	39.584	39.411	39.279

we can see in Table 1, when the number of skipped blocks increases, the capacity decreases and PSNRs increase. If an image contains a highlighted area, our embedding algorithm has a high probability of getting overflow on the coefficient corresponding to this area. For example image 'Tiffany' has a large lighter area. Thus the embedding algorithm skips the corresponding blocks and gets a higher PSNR.

In Table 2, we test our approach on a bigger set of images. We find that the PSNR and capacity have a similar trend when comparing Tables 2 and 1. That can strengthen our confidence in comparing our approach with other methods using the six images.

We compare the capacities and PSNRs of our scheme where $p = 2$ and $q = 7$ with the methods proposed in Ref. [26] and Ref. [27]. The results of embedding random bit streams into the test set are presented in Table 3. Column 1 shows the names of the test files, and column 2 shows the size of the original test files. Column 3, entitled "Wu and Tasi's method", refers to the method proposed in Ref. [26], column 4, entitled "Yang et al.'s method", refers to the method proposed in Ref. [27], and column 5, titled "Our approach", refers to the proposed scheme where $p = 2$ and $q = 7$. The first sub-column of column 3, denoted 'Capacity', shows the capacities of the algorithm proposed in Ref. [26], while the second sub-column, denoted 'PSNR', of column 3, shows the PSNRs of the test images, and the first sub-column of column 4 shows the capacities of the algorithm proposed in Ref. [27], while the second sub-column of column 4 shows the PSNRs of the test images. The observation values of the proposed scheme are shown in column 5. Column 5 is also divided into two sub-columns. The first shows the capacities of the scheme, where the unit is bits, while the second shows the PSNRs of the test images. Since our method selects a fixed number of coefficients for hiding secret information in a block, the capacity of our approach is the same if the image size, p, and q are fixed when the number of skipped blocks is zero. For example, in Table 1, 'Lena' and 'Goldhill' have the same capacity when p and q are fixed, since their numbers of skipped blocks are both zero. However, the PSNRs of our method are various. On the other hand, spatial-domain methods in Ref. [26] and Ref. [27] modify partial pixels to hide information. Thus, their methods have various capacities when embedding secret information in the same image size.

Table 4 compares the results of embedding random bit streams into the test set with our method and with the two DCT-based methods proposed in Ref. [5] and Ref. [16]. Column 1 shows the names of the test files, and column 2 shows the size of the original test files. Column 3, entitled "Chang et al.'s method", refers to the method proposed in Ref. [5], column 4, entitled "Lin and Shiu's method", refers to the method proposed in Ref. [16], and column 5, titled "Our approach", refers to the proposed scheme where $p = 2$ and $q = 7$. The meaning of the sub-columns in Table 4 is similar to those in Table 3. Obviously, our approach outperforms in both PSNR and capacity for almost all test images in Tables 3 and 4. When test image is 'Baboon', Yang et al.'s method [27] has better capacity but worse PSNR than our approach. However, if parameters are set to $p = 3$ and $q = 7$, our approach outperforms in both PSNR and capacity for image 'Baboon'.

Table 3
Comparisons of the results for embedding random bit streams into cover images using the method in Ref. [26], the method in Ref. [27], and our approach with $p = 2$ and $q = 7$ (The 100 different random bit streams are used to get the average PSNRs shown here).

File name	Image size	Wu and Tsai's method [26]		Yang et al.'s method [27]		Our approach	
		Capacity	PSNR	Capacity	PSNR	Capacity	PSNR
Lena	512 × 512	406,632	41.71	410,854	40.54	458,752	43.97
Baboon	512 × 512	437,806	38.90	482,515	34.67	457,736	43.98
Peppers	512 × 512	401,980	41.07	408,281	40.47	452,085	44.03
Boat	512 × 512	415,554	40.67	430,888	38.11	458,423	43.97
Goldhill	512 × 512	405,634	42.20	418,575	40.25	458,752	43.93
Tiffany	512 × 512	403,764	41.47	408,486	39.24	446,089	44.08

Table 4
Comparisons of the results for embedding random bit streams into cover images using the method in Ref. [5], the method in Ref. [16], and our approach with $p = 2$ and $q = 7$ (The 100 different random bit streams are used to get the average PSNRs shown here.).

File name	Image size	Chang et al.'s method [5]		Lin and Shiu's method [16]		Our approach	
		Capacity	PSNR	Capacity	PSNR	Capacity	PSNR
Lena	512 × 512	212,992	34.84	90,112	35.28	458,752	43.97
Baboon	512 × 512	212,992	27.63	90,112	28.22	457,736	43.98
Boat	512 × 512	212,992	33.29	90,112	33.05	458,423	43.97

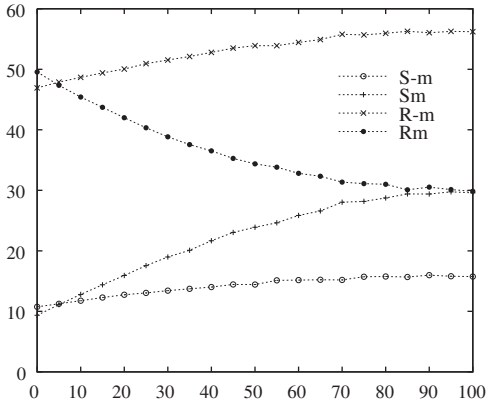

Fig. 6. RS-diagram for experiment of a stego-image 'Goldhill' using 1-bit LSB substitution. The horizontal axis is the percentage of pixels with flipped LSBs. The vertical axis is the percentage of regular and singular groups.

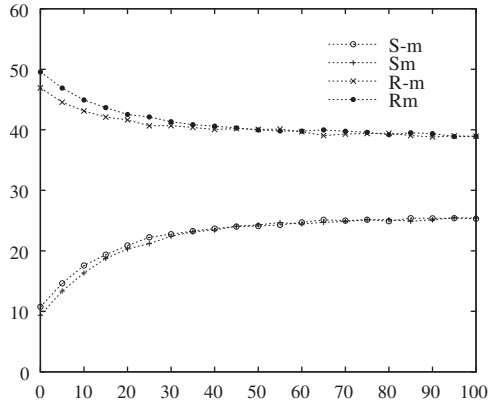

Fig. 7. RS-diagram for experiment of a stego-image 'Goldhill' using our method. The horizontal axis is the percentage of modified coefficients. The vertical axis is the percentage of regular and singular groups.

Based on Tables 3 and 4, frequency-domain methods do not always outperform spatial-domain method. In Table 4, the frequency-domain methods [5,16] have less capacity and worse PSNRs than the spatial-domain methods [26,27]. But our approach, as a frequency-domain method, surpasses, in both PSNR and capacity, the spatial-domain methods [26,27] as shown in Table 3. The major reason is that our DCT-based method is excellent in gathering the energy of the image into the low frequency and utilizes the room remaining in the high frequency to hide the secret information.

We use the test tool proposed by Fridrich et al. [7] for detection of the embedding of the gray-scale images. The detection results of testing 'Goldhill' with 1-bit LSB substitution and our method are shown in Figs. 6 and 7. In Fig. 6, because the difference between the percentages of the two singular pixel groups becomes larger when the percentage of pixels embedded by a simple 1-bit LSB substitution increases to 100%, the embedding is detected. In Fig. 7, because the RS-diagram of our method with $q = 1$ indicates that R_m is close to R_{-m} and that S_m gets closer to S_{-m} when the percentage of coefficients embedded by our method increases to 100%, the stego-image seems to have no embedded data.

The results of the work show that the quality can be greatly improved by the proposed scheme while also improving the capacity for all test images. Moreover, our method passes the RS-diagram test.

In the last experiment, we attack proposed steganographic algorithm using the 548-dimensional CC-PEV features [12] and ensemble classifier [13] which is downloaded from http://dde.binghamton.edu/download/ensemble/. The set of stego images was created using our approach with $p = 2$ and $q = 7$ for each payload $\alpha \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06\}$ bpp (bits per pixel). After embedding secret data with our approach, all 9074 BOSSbase images were JPEG compressed using the quality factor 100 using Matlab's command `imwrite`. Half of the 9074 images are used for training, and the other half for testing. Table 5 shows the detection accuracy. We can conclude that our approach can be easily detected with high payload.

6. Conclusions

To look for the proper pixels for hiding data, a transform-based scheme is proposed in this paper. To improve the quality of stego-images, cover images are decomposed into several different frequencies, and the high-frequency parts contain hidden data. Our experimental results show that the proposed scheme provides high image quality and large message capacity. We conclude that steganography based on DCT coefficients has high potential in competition with the pixel-value differencing based method. Our future work will focus on further increasing the embedding capacity, while maintaining the quality of the stego-image. A possible method is to find the way to hide different numbers of secret bits at different coefficient components to achieve this goal.

Table 5
Steganalysis of our approach.

Payload (bpp)	0.005	0.010	0.015	0.020	0.025	0.030	0.100
Testing error	0.4232	0.3325	0.2918	0.2663	0.2418	0.2222	0.1199

References

[1] W. Bender, D. Gruhl, N. Morimoto, A. Lu, Techniques for data hiding, IBM Syst. J. 35 (1996) 313–316.
[2] S. Browne, J. Dongarra, N. Garner, G. Ho, P. Mucci, A portable programming interface for performance evaluation on modern processors, Int. J. High Perform. Comput. Appl. 14 (3) (2000) 189–204.
[3] G. Bradski, The OpenCV library, Dr. Dobb's J. Softw. Tools (2000).
[4] C.K. Chan, L.M. Chen, Hiding data in images by simple LSB substitution, Pattern Recogn. 37 (2004) 313–316.
[5] C.C. Chang, T.S. Chen, L.Z. Chung, A steganographic method based upon JPEG and quantization table modification, Inf. Sci. 141 (2002) 123–138.
[6] C.C. Chang, C.C. Lin, C.S. Tseng, W.L. Tai, Reversible hiding in DCT-based compressed images, Inf. Sci. 177 (2007) 2768–2786.
[7] J. Fridrich, M. Goljan, R. Du, Reliable detection of LSB steganography in gray-scale and color images, Proceedings of the ACM Workshop on Multimedia and Security, 2001, pp. 27–30.
[8] J. Fridrich, T. Pevný, J. Kodovský, Statistically undetectable JPEG steganography: dead ends, challenges, and opportunities, in: J. Dittmann, J. Fridrich (Eds.), Proceedings of the 9th ACM Multimedia & Workshop, pages 3–14, Dallas, TX, September 20–21, 2007.
[9] P. Hao, Q. Shi, Matrix factorizations for reversible integer mapping, IEEE Trans. Signal Process. 49 (10) (2001) 2314–2324.
[10] W. Hong, Adaptive reversible data hiding method based on error energy control and histogram shifting, Opt. Commun. 285 (2011) 101–108.
[11] A.D. Ker, Steganalysis of embedding in two least-significant bits, IEEE Trans. Inf. Forensics Secur. 2 (1) (2007) 46–54.
[12] J. Kodovsky, J. Fridrich, Calibration revisited, in: J. Dittmann, S. Craver, J. Fridrich (Eds.), Proceedings of the 11th ACM Multimedia & Security Workshop, pages 6374, Princeton, NJ, September, 2009, 2009.
[13] J. Kodovsky, J. Fridrich, V. Holub, Ensemble classifiers for steganalysis of digital media, IEEE Trans. Inf. Forensics Secur. 7 (2) (April, 2012) 432–444.
[14] M. Kumar, R. Newman, J3: high payload histogram neutral JPEG steganography, Eighth Annual conference on Privacy, Security and Trust, August 2010, pp. 46–53.
[15] X. Li, B. Yang, D.F. Cheng, Zeng T.Y., A generalization of LSB matching, IEEE Signal Process Lett. 16 (2) (2009) 69–72.
[16] C.C. Lin, P.F. Shiu, High capacity data hiding scheme for DCT-based images, J. Inf. Hiding Multimedia Signal Process. 1 (3) (2010) 220–240.
[17] C.C. Lin, An information hiding scheme with minimal image distortion, Comput. Stand. Interfaces 33 (2011) 477–484.
[18] Y.K. Lin, High capacity reversible data hiding scheme based upon discrete cosine transformation, J. Syst. Softw. 85 (10) (2012) 2395–2404.
[19] T. Pevný, T. Filler, P. Bas, Break our steganographic system, http://boss.gipsa-lab.grenoble-inp.fr.
[20] H. Qi, Q. Huang, W. Gao, D. Zhao, Error-resistance and low-complexity integer inverse discrete cosine transform, Signal Process. Syst. 61 (2) (2010) 231–239.
[21] K. Solanki, A. Sarkar, B. Manjunath, YASS: yet another steganographic scheme that resists blind steganalysis, IH'07 Proceedings of 9th International on, Information, 2007, pp. 16–31.
[22] J. Tian, Reversible data embedding using a difference expansion, IEEE Trans. Circuits Syst. Video Technol. 13 (8) (2003) 890–896.
[23] H.W. Tseng, C.C. Chang, High capacity data hiding in JPEG-compressed images, Informatica 15 (1) (2004) 127–142.
[24] K. Wahid, G. Jullien, Error-free computation of 8 × 8 2-D DCT and IDCT using two-dimensional algebraic integer quantization, ARITH'05: Proceedings of the 17th IEEE Symposium on Computer Arithmetic, June, 2005, pp. 214–221.
[25] A. Westfeld, High capacity despite better steganalysis (F5—a steganographic algorithm), in: I.S. Moskowitz (Ed.), Information Hiding, 4th International Workshop, volume 2137 of Lecture Notes in Computer Science, Springer-Verlag, New York, 2001, pp. 289–302.
[26] D.C. Wu, W.H. Tsai, A steganographic method for images by pixel-value differencing, Pattern Recogn. Lett. 24 (9–10) (2003) 1613–1626.
[27] C.H. Yang, C.Y. Weng, H.K. Tso, S.J. Wang, A data hiding scheme using the varieties of pixel-value differencing in multimedia images, J. Syst. Softw. 84 (2011) 669–678.
[28] C.H. Yang, S.C. Wu, S.C. Huang, Y.K. Lin, Huffman-code strategies to improve MFCVQ-based reversible data hiding for VQ indexes, J. Syst. Softw. 84 (3) (2011) 388–396.

Yih-Kai Lin received the PhD degree in the Electrical Engineering from National Taiwan University, Taiwan, in 2003. He is presently an associate professor of Computer Science at National Pingtung University of Education. His current research interests include data compression, pattern recognition, computer graphics and image processing.