A Two-Layer Steganography for Mosaic Images

Ji-Hwei Horng[1], Chin-Chen Chang[2,*], and Kun-Sheng Sun[1]
[1]Department of Electronic Engineering, National Quemoy University, Kinmen 89250, Taiwan
[2]Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan
[E-mail : horng@email.nqu.edu.tw, alan3c@gmail.com, s110940101@student.nqu.edu.tw]
[*]Corresponding Author : Chin-Chen Chang

Received October 19, 2020; revised March 17, 2021; accepted May 18, 2021; published September 30, 2021

Abstract

A lot of data hiding schemes have been proposed to embed secret data in the plain cover images or compressed images of various formats, including JPEG, AMBTC, VQ, etc. In this paper, we propose a production process of mosaic images based on three regular images of coffee beans. A primary image is first mimicked by the process to produce a mosaic cover image. A two-layer steganography is applied to hide secret data in the mosaic image. Based on the low visual quality of the mosaic cover image, its PSNR value can be improved about 1.5 dB after embedding 3 bpp. This is achieved by leveraging the newly proposed polarized search mask and the concepts of strong embedding and weak embedding. Applying steganography to the mosaic cover images is a completely new idea and it is promising.

Keywords: Data hiding, mosaic image, polarized mask, strong embedding, weak embedding.

1. Introduction

With the rapid growth of Internet multimedia transmission, many steganographic techniques have been proposed to hide secret data in digital images. In 2006, Zhang and Wang proposed the exploiting modification direction (EMD) method [1]. The EMD data hiding scheme embeds a 5-ary secret digit in a pair of pixels by adjusting at most one pixel with value 1. Later, some improved schemes [2-3] were proposed to raise its hiding capacity. In 2014, Chang et al. proposed a turtle shell matrix [4] that can help to hide an 8-ary digit per pixel pair. It is advantageous to hide 2^n-ary digit, since the secret digit can be converted directly from integer number of binary secret bits. Many related works were proposed [5-9] to improve its performance.

In 2015, Kurup et al. proposed an octagon shaped shell matrix [10] to raise the hiding capacity of turtle shell-based scheme from three bits per pixel pair to four bits. In 2019, Leng further expanded the geometric size of octagon shaped shell [11]. By applying his generalized design rule, different-sized octagon shaped shell matrices can be obtained to embed three to eight bits per pixel pair.

In 2008, Chang et al. proposed a Sudoku-based data hiding scheme [12]. By leveraging the high complexity of Sudoku matrix, this scheme greatly improves the security of reference matrix-based approach. Many methods were proposed to improve the hiding capacity or visual quality of stego image [13-16].

In recent years, many reversible data hiding techniques were proposed [17-20]. After extraction of secret data, the cover image can be restored. However, the reversibility severely degrades the hiding capacity. Another interesting novel topic is secret image sharing [21-23]. The secret data is embedded into multiple different cover images or repetitions of the same cover image. Without cooperation of all image shares, the secret cannot be extracted.

Instead of hiding data in plain image, some researchers turned to embeding secret data into AMBTC [24] or JPEG [25-26] compressed images. Although the embedding capacity is much less than that of the plain image embedding, these methods are more practical for applications.

In this paper, we propose a novel and interesting topic of mosaic image data hiding. A mosaic image is synthesized by small image tiles to mimic a given primary image. By replacing image tiles and modification of pixel values, we can embed two layers of secret data in a mosaic image. The rest of this paper is organized as follows. Section 2 introduces a reference matrix-based data hiding scheme, which is a commonly applied data hiding scheme in recent years. In Section 3, the proposed scheme, including synthesis of mosaic images, the first and second layers of data hiding, and the corresponding extraction process, is presented. Experimental results are given in Section 4. Conclusions are presented in Section 5.

2. The data hiding based the generalized octagon shaped shell matrix

A series of reference matrix-based data hiding schemes have been proposed to hide secret data in a cover image. In 2014, Chang et al. proposed a turtle shell reference matrix [4] as shown in **Fig. 1**. It is so ingeniously designed that any arbitrarily selected element can find all distinct integer values from 0 to 7 in its vicinity. Through guidance of the turtle shell matrix, three bits of secret data can be embedded to each pair of pixels.

In their scheme, the cover image is divided into mutually exclusive pixel pairs first. Then, the gray level values of each pixel pair are applied as coordinates and mapped to an element in the turtle shell matrix. According to the decimal value of three secret bits to be embedded, the nearest element with its value matched the secret value is selected as target. Then, the

coordinates of the matched element are recorded to the stego image. This procedure is repeated until all pixel pairs are processed. Since a target element can always be found in vicinity of the mapped element, the resulting stego image looks the same as the cover image. In other words, the difference between them cannot be distinguished by human eyes.

Later, the idea of turtle shell matrix was expanded to the octagon-shaped shell [10] and the generalized octagon-shaped shell [11] matrices as shown in **Fig. 2**. The embedding capacity of a pixel pair can be raised to four bits by compactly arranging sixteen distinct integer values of 0 to 15 in each of the consecutive octagons [10] as shown in **Fig. 2(a)**. Leng further expanded the size of an octagon to include 2^n elements [11] and thus it can be applied to embed n bits of secret data for each cover pixel pair. Of course, the search range of target element should be increased with the expansion of octagonal size. Thus, the displacement of coordinates is increased and the visual quality of stego images is also degraded at the same time.

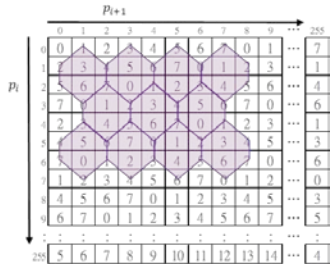
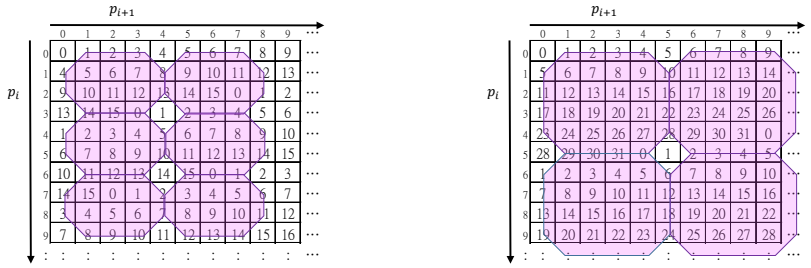


Fig. 1. The turtle shell reference matrix proposed in [1].



(a) 16-ary octagon shaped shell (b) 32-ary octagon shaped shell

Fig. 2. The generalized octagon shaped shell reference matrices.

3. The proposed scheme

In this paper, we propose a two-layer data hiding scheme for mosaic images. In this section, we first introduce our procedure of mosaic image production. Then, the data hiding scheme for the first layer and the second layer embedding and extraction are presented.

3.1 Synthesis of Mosaic Images

A mosaic image, also called a photomosaic, is an image that has been divided into tiled sections, each of which is replaced with an image tile that best represents the target section. When viewed at a low resolution, it looks like the primary image, while close examination reveals the texture of the image tiles.

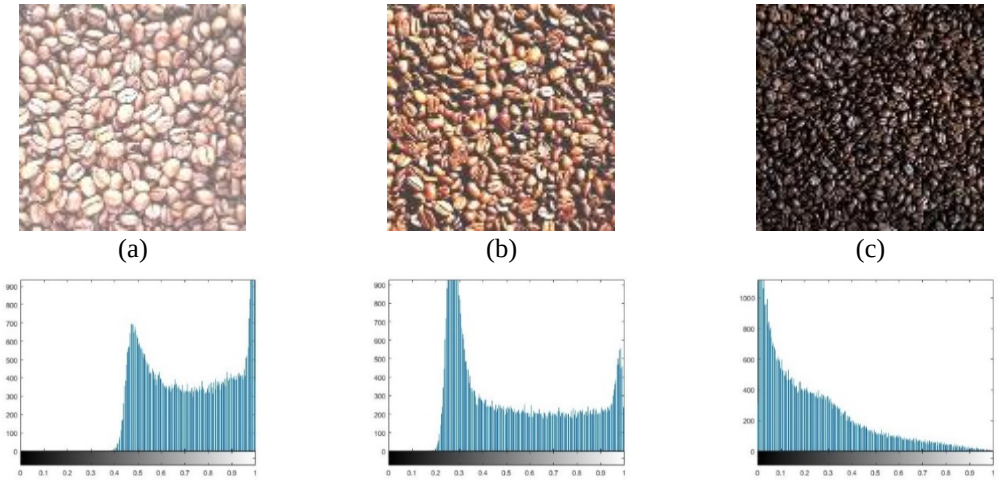
3.1.1 Library of image tiles

To produce mosaic images, a library of image tiles has to be set up first. In this paper, we apply three images of coffee beans with size 512×512 [27-29] as shown in **Figs. 3(a)-(c)** to

generate the elementary image tiles. Their corresponding histograms are given in **Figs. 3(d)-(f)**, which are mainly distributed in the ranges of high, median, and low, respectively.

By duplicating the coffee bean images and modifying their gray level with linear transformation, 7 images with different ranges of gray level distribution are produced. Each of which is then divided into 32×32 image tiles of size 16×16. It results in 7,168 tiles in the whole image library. Since all of the elementary image tiles come from the coffee bean images, there is only a single tone of brown color. To represent arbitrary images, we have to leverage their luminance. The color space of image tiles is converted from RGB to YIQ, where Y is the luminance representation of the image tiles.

The elementary image tiles are sorted by average luminance and the 256 tiles that best match the integer gray level values from 0 to 255 are selected as the candidate tiles. The gray level distribution of each candidate tile is then fine-tuned to optimize its average gray level. In this way, we can obtain a library of image tiles $U = \{B_i(x, y)|x, y = 0,1,2, \dots ,15; i = 0,1,2, \dots ,255\}$ that contains 256 members, the corresponding gray level value i of the image tile $B_i(x, y)$ is defined as *tile index*.



(a) (b) (c)

(d) (e) (f)

Fig. 3. Three images of coffee beans (a) - (c) with their histograms (d) - (f).

3.1.2 Tiling mosaic images

Based on the library of coffee bean image tiles, we can produce a mosaic image to mimic any arbitrarily given primary image. The input image is divided into blocks of size 16×16 first. Then, the color space is converted into YIQ domain. By averaging the Y values of each block, we can obtain an index matrix and tile the mosaic image according to it. The algorithm for generating a mosaic image to mimic a given primary image is provided as follows.

Algorithm: Generation of mosaic image	
Input:	primary image I_P of size $w \times h$, library of image tiles $U = \{B_i(x, y)\|x, y = 0,1, \dots ,15; i = 0,1, \dots ,255\}$.
Output:	mosaic image I_M of size $w \times h$.
1:	Divide I_P into $m \times n$ image blocks $P_{i,j}(x, y)$ of size 16×16, where $m = w/16, n = h/16$; (i, j) are the block coordinates; (x, y) are the pixel coordinates in a block.
2:	Initialize index matrix Q of size $m \times n$.

3:	For each image block $P_{i,j}$,
4:	Convert color space of $P_{i,j}$ to obtain luminance matrix $Y_{i,j}$.
5:	$Q(i,j) = \frac{1}{16\times16}\sum_{x,y} Y_{i,j}(x,y)$.
6:	$I_M(16\times(i-1)+x, 16(j-1)+y) = B_{Q(i,j)}(x,y)$, for $x, y = 0,1,2,\ldots,15$.
	End
7:	Output I_M.

Fig. 4(a) shows a true color image with size 1600×1792 of a tiger [27]. The resulted mosaic image is shown in **Fig. 4(b)**. Since it is produced with coffee bean images of a single tone, the mosaic image is basically brown color with a texture of coffee beans.

(a) (b)

Fig. 4. The mosaic image production: (a) an image of a tiger, (b) its corresponding mosaic image produced with the library of coffee bean image tiles.

3.2 The first layer of data hiding

To hide secret data into a mosaic image, we apply a modified version of the turtle shell matrix-based data hiding scheme [4] proposed by Chang et al. in 2014. Refer to **Fig. 5**, the red rectangular region of mosaic image "Tiger" in **(a)** is magnified as shown in **(b)**. By converting to the YIQ color space and averaging luminance of each image tile, we can obtain the index matrix. Refer to **Fig. 6**, the mosaic image **(a)** is a copy of **Fig. 5(b)**. Part of index values of **Fig.6(a)** is shown in **(c)**, where the corresponding ranges of pixel addresses are labeled in the gray axial frame.

The indices of the index matrix are divided into index pairs. Through the guidance of turtle shell matrix as shown in **Fig. 7**, each pair of indices can be applied to embed three bits of secret data. An example index pair of (153,128) in **Fig. 6(c)** maps to the red color element as shown in **Fig. 7(a)**. Suppose the three bits to be embedded are $(100)_2$. Thus, the matched target element is (152,128). The two coordinates are recorded to the index matrix of stego image as shown in **Fig. 6(d)** and their corresponding image tiles are posted to **Fig. 6(b)**. The second example is the green index pair (112,123) in **Fig. 6(c)**. The mapped element with its vicinity in the reference matrix is shown in **Fig. 7(b)**. Suppose the secret bits to be embedded are $(110)_2$. The matched target element is the yellow element. Then, its coordinates (113,124) and corresponding image tiles are recorded to **Figs. 6(d)** and **6(b)**, respectively. In this way, the index matrix is modified with the turtle shell-based data hiding scheme and the stego mosaic image is tiled according to the modified index matrix. The algorithm for the first layer of data hiding is summarized as follows.

(a) (b)

Fig. 5. A close examination: (a) the mosaic image "Tiger", (b) a close examination of the upper-left square region of (a).

(a) (b)

	1:16	17:32	33:48	49:64
1:16	153	128	112	123
17:32	151	135	120	119
33:48	150	148	144	134

(c)

	1:16	17:32	33:48	49:64
1:16	152	128	113	124
17:32	150	134	120	120
33:48	151	149	143	134

(d)

Fig. 6. A demonstration of the turtle shell matrix-based data hiding for mosaic images.

	126	127	128	129	130
151	7	0	1	2	3
152	2	3	4	5	6
153	4	5	6	7	0
154	7	0	1	2	3
155	1	2	3	4	5

(a)

	121	122	123	124	125
110	4	5	6	7	0
111	6	7	0	1	2
112	1	2	3	4	5
113	3	4	5	6	7
114	6	7	0	1	2

(b)

Fig. 7. Illustration of the data hiding process.

Algorithm: The first layer of data hiding for mosaic images

Input: mosaic image I_M of size $w \times h$,
library of image tiles $U = \{B_i(x,y) | x,y = 0,1,\dots,15; i = 0,1,\dots,255\}$,
binary secret stream S.

Output: stego mosaic image $\bar{I}_M$ of size $w \times h$.

1: Divide I_M into $m \times n$ image blocks $M_{i,j}(x,y)$ of size 16×16,
where $m = w/16, n = h/16$; (i,j) are the block coordinates;
(x,y) are the pixel coordinates in a block.

2: Initialize index matrix Q of size $m \times n$.

3: **For** each image block $M_{i,j}$,

4: $Q(i,j) = \frac{1}{16 \times 16} \sum_{x,y} M_{i,j}^{Y}(x,y)$,
where $M_{i,j}^{Y}(x,y)$ is the luminance of $M_{i,j}(x,y)$.
End
5: Initialize stego index sequence $\bar{Q}$.
6: Rearrange Q into $Q = \{(q_{k1}, q_{k2}) | k = 1,2, \ldots, (m \times n)/2\}$.
7: Construct turtle shell matrix $RM(0{:}\,255, 0{:}\,255)$ as shown in Fig. 1.
8: **For** each pair of indices (q_{k1}, q_{k2}),
9: Map (q_{k1}, q_{k2}) to $RM(q_{k1}, q_{k2})$.
10: Retrieve 3 bits from S and convert to decimal digit s_k.
11: Find the nearest element $RM(\bar{q}_{k1}, \bar{q}_{k2}) = s_k$.
12: Record $(\bar{q}_{k1}, \bar{q}_{k2})$ to $\bar{Q}$.
End
13: Rearrange $\bar{Q} = \{(\bar{q}_{k1}, \bar{q}_{k2}) | k = 1,2, \ldots, (m \times n)/2\}$ into matrix $\bar{Q}$.
14: Initialize $\bar{I}_M$.
15: **For** each index $\bar{Q}(i,j)$,
16: $\bar{I}_M(16 \times (i-1) + x, 16(j-1) + y) = B_{\bar{Q}(i,j)}(x,y)$,
for $x, y = 0,1,2, \ldots, 15$.
End
17: Output $\bar{I}_M$.

3.3 The second layer of data hiding

In the first layer of data hiding, we simply embed secret data by image tile replacement. The only type of modification is to replace an image tile with another image tile of close index value. In the second layer of data hiding, we treat the luminance of first layer-embedded mosaic image as an ordinary cover image and apply the octagon-shaped shell scheme [11] to embed secret data in the pixel level instead of the tile level. A major difference from the conventional embedding is that the visual quality of our stego image is evaluated based on the primary image but not the cover/mosaic image.

In the conventional reference matrix-based embedding, the search range of target element is centered at the element mapped by the cover pixel pair. In the new situation, we have to know the relationship between the current mosaic pixel pair and the primary pixel pair in advance. When we modify the mosaic pixel value toward the primary pixel value, the visual quality of stego image is improved after embedding. To gain this effect, we propose a *polarized search mask* for reference matrix.

The idea is illustrated in **Fig. 8**, where three different relationships between the mosaic pixel values and the primary pixel values are considered. The principle for mask setting is to include the element mapped by the cover pixel pair and drift toward the element mapped by the primary pixel pair. Based on the polarized mask, secret data can be embedded with a least modification of cover pixel values and a reduction of error with respect to their corresponding primary pixel values.

Why don't we stretch further toward the primary pixel value and gain an even better visual quality? The answer is that modification of pixel values result in the drift of average luminance and thus may lead to error decoding of the first layer data. Therefore, the total amount of deviations should be carefully controlled.

To permit more flexibility of embedding, we enlarge the polarized search mask to enclose more elements in the reference matrix. Thus, multiple matched target elements can be found to embed an assigned secret digit. An example is illustrated in **Fig. 9**, where the polarized mask is enlarged to 5×5. Suppose the three bits to be embedded are $(010)_2$. Three matched elements can be found in the polarized mask. Recording the coordinates of the element closest to the cover element is defined as *weak embedding* (WE), while recording the element closest to the primary element is defined as *strong embedding* (SE).

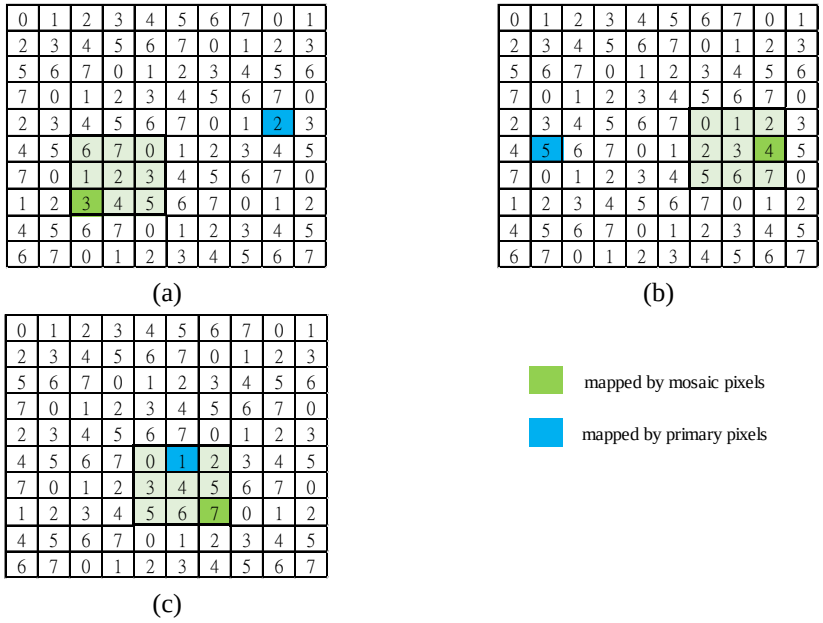


Fig. 8. The polarized search mask for turtle shell matrix with different conditions.

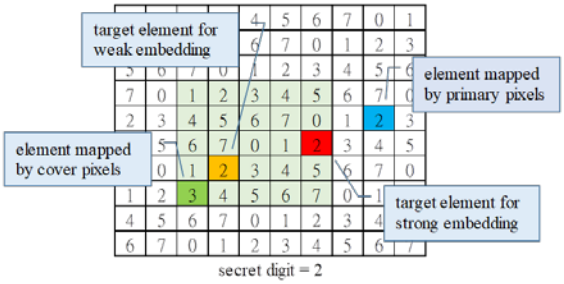


Fig. 9. An illustration of weak embedding and strong embedding.

Before deciding the embedding strategy, the relationship between a mosaic tile and its corresponding block in the primary image should be investigated first. The mosaic tile and its corresponding primary image block are denoted by $M(x, y)$ and $P(x, y)$, respectively. For each mosaic tile, we calculate its positive difference rate (*PD rate*) by

$$PD\ rate = \frac{1}{16 \times 16} \sum_{x,y} sign\ [P(x, y) - M(x, y)], \tag{1}$$

where the function $sign(\cdot)$ is defined by

$$sign(n) = \begin{cases} 1, & n > 0; \\ 0, & n \leq 0. \end{cases} \quad (2)$$

The *PD rate* is used to calculate the percentage of pixels in the tile. It requires positive adjustment of pixel value to approach the primary one. According to the *PD rate*, the image tiles are classified into three types as illustrated in **Fig. 10**. In the second layer of embedding, we propose three different strategies to process different types of mosaic tiles.

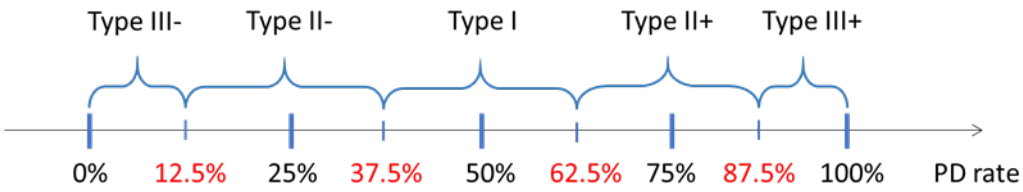


Fig. 10. Classification of embedding strategy.

According to the sign value of difference $sign\,[P(x,y) - M(x,y)]$, the pixel pairs in a mosaic tile can be divided into four groups of $(+,+),(+,-),(-,+)$, and $(-,-)$, where $P(x,y) > M(x,y)$ is denoted by '$+$'; while $P(x,y) \leq M(x,y)$ is denoted by '$-$'. For example, let the cover pixel pair be $(121,89)$ and its corresponding primary pixel pair be $(123,80)$. Since $123 > 121$ and $80 \leq 89$, this pixel pair is classified to the group of $(+,-)$.

For mosaic tiles of Type I, we apply enlarged version of polarized search mask with strong embedding (SE) to process all pixel pairs. Since the *PD rate* is around fifty percent, the positive and negative deviations caused by embedding tend to cancel out each other.

For mosaic tiles of Type II+, strong embedding (SE) is applied to the $(-,-)$ group of pixel pairs, while weak embedding (WE) is applied to the other groups. For complementary distribution of Type II-, SE is applied to the $(+,+)$ group of pixel pairs, while WE is applied to the other groups. The principle is to permit further stretch for group of lower population. Thus, the total amount of positive and negative deviations are balanced.

For mosaic tiles of Type III+, SE is applied to all pixel pairs in $(-,-)$ group and equal amount of pixel pairs in $(+,+)$ group by random selection. For the remainder, conventional embedding (CE) is applied. The equal amount of $(-,-)$ and $(+,+)$ pixel pairs tend to cancel out deviations, while the deviations produced by CE are symmetrically distributed. Analogue rules are applied to the complementary distribution of Type III-.

To ensure correct decoding of tile index, the average luminance of each stego mosaic tile should be within a predefined tolerance. For instance, the average luminance of a mosaic tile with index N should be within the range of $(N - 0.4, N + 0.4)$ after embedding. Compensation process should be taken when it is out of the tolerance range. Two possible ways of compensation are to weaken the embedding strength of over deviation group or to strengthen the embedding strength of under deviation group. The strength of embedding listed in the descending order are SE, WE, and CE. The algorithm for the second layer of data hiding is summarized as follows.

Algorithm: The second layer of data hiding for mosaic images

Input: primary image I_P of size $w \times h$,
first layer embedded stego mosaic image $\bar{I}_M$ of size $w \times h$,
binary secret stream S.

Output: fully embedded stego mosaic image $\bar{\bar{I}}_M$ of size $w \times h$.

Divide $\bar{I}_M$ into $m \times n$ image blocks $M_{i,j}(x, y)$ of size 16×16,
where $m = w/16$, $n = h/16$; (i, j) are the block coordinates;
(x, y) are the pixel coordinates in a block.

Initialize stego index matrix $\bar{Q}$ of size $m \times n$.

For each image block $M_{i,j}$,

Convert color space of $M_{i,j}$ to obtain luminance matrix $Y_{i,j}$.

$\bar{Q}(i,j) = \frac{1}{16\times16}\sum_{x,y} Y_{i,j}\,(x, y)$.

End

Initialize $\bar{\bar{I}}_M$.

For each image tile $M_{i,j}$ of $\bar{I}_M$,

$M_{i,j}(x, y) = \bar{I}_M(16 \times (i-1) + x, 16(j-1) + y)$,

$P_{i,j}(x, y) = I_P(16 \times (i-1) + x, 16(j-1) + y)$,
for $x, y = 0,1,2, \ldots, 15$.

$$PD\ rate = \frac{1}{16 \times 16}\sum_{x,y} sign\,[P^Y_{i,j}(x, y) - M^Y_{i,j}(x, y)].$$

Rearrange $M_{i,j}$ into sequence of pixel pairs
$M_{i,j} = \{(r_{k1}, r_{k2}) | k = 1,2,3, \ldots, (16 \times 16)/2\}$.

Group pixel pairs into $(+,+)$, $(+,-)$, $(-,+)$, and $(-,-)$.

Construct octagon-shaped shell matrix $RM(0{:}\,255, 0{:}\,255)$.

Type I ($37.5\% < PD\ rate \leq 62.5\%$):
Embed all pixel pairs with **SE**.

Type II+ ($62.5\% < PD\ rate \leq 87.5\%$):
Embed group $(-,-)$ with **SE**.
Embed the other groups with **WE**.

Type II- ($12.5\% < PD\ rate \leq 37.5\%$):
Embed group $(+,+)$ with **SE**.
Embed the other groups with **WE**.

Type III+ ($87.5\% < PD\ rate \leq 100\%$):
Embed group $(-,-)$, equal pairs of $(+,+)$ group with **SE**.
Embed the other pixel pairs with **CE**.

Type III- ($0\% < PD\ rate \leq 12.5\%$):
Embed group $(+,+)$, equal pairs of $(-,-)$ group with **SE**.
Embed the other pixel pairs with **CE**.

The embedding result is denoted by
$\hat{M}_{i,j} = \{(\hat{r}_{k1}, \hat{r}_{k2}) | k = 1,2,3, \ldots, (16 \times 16)/2\}$.

Calculate average luminance $Y = \frac{1}{16\times16}\sum \hat{M}_{i,j}$.

If $Y \notin (\bar{Q}(i,j) - 0.4,\ \bar{Q}(i,j) + 0.4)$,
Take compensation process.
End

Convert $\hat{M}_{i,j}$ back into matrix form $\hat{M}_{i,j}(x, y)$.

$\bar{\bar{I}}_M(16 \times (i-1) + x, 16(j-1) + y) = \hat{M}_{i,j}(x, y)$,
for $x, y = 0,1,2, \ldots, 15$.

End

Output $\bar{\bar{I}}_M$.

3.4 Extraction of secret data

Comparing with the embedding process, secret data extraction of the proposed scheme is relatively simple. The secret data embedded in the second layer is extracted first. Then, the index of each mosaic tile is decoded to extract the data embedded in the first layer. The secret data extraction algorithm is given as follows.

Algorithm: The secret data extraction

Input: stego mosaic image $\bar{\bar{I}}_M$ of size $w \times h$.

Output: binary secret stream S.

Second layer data extraction:

Duplicate $\bar{\bar{I}}_M$ and rearrange into series of pixel pairs $\hat{I}_M$.
  $\hat{I}_M = \{(p_{k1}, p_{k2}) | k = 1,2,3, \ldots, (w \times h)/2\}$.

Construct octagon-shaped shell matrix $RM(0{:}\,255, 0{:}\,255)$.

For each pixel pair (p_{k1}, p_{k2}),
  Map (p_{k1}, p_{k2}) to RM and retrieve $s_k = RM(p_{k1}, p_{k2})$.
  Convert decimal digit s_k to binary and record to S_2.

End

First layer data extraction:

Divide $\bar{\bar{I}}_M$ into $m \times n$ image blocks $\hat{M}_{i,j}(x, y)$ of size 16×16,
  where $m = w/16$, $n = h/16$; (i, j) are the block coordinates;
  (x, y) are the pixel coordinates in a block.

Initialize stego index matrix $\bar{Q}$ of size $m \times n$.

For each image block $\hat{M}_{i,j}$,

$$\bar{Q}(i,j) = \frac{1}{16 \times 16} \sum_{x,y} \hat{M}^{Y}_{i,j}(x, y),$$

  where $\hat{M}^{Y}_{i,j}(x, y)$ is the luminance of $\hat{M}_{i,j}(x, y)$.

End

Rearrange $\bar{Q}(i, j)$ into pixel pairs
  $\bar{Q} = \{(\bar{q}_{k1}, \bar{q}_{k2}) | k = 1,2, \ldots, (m \times n)/2\}$.

Construct turtle shell matrix $RM(0{:}\,255, 0{:}\,255)$.

For each pixel pair $(\bar{q}_{k1}, \bar{q}_{k2})$,
  Map $(\bar{q}_{k1}, \bar{q}_{k2})$ to RM and retrieve $s_k = RM(\bar{q}_{k1}, \bar{q}_{k2})$.
  Convert decimal digit s_k to binary and record to S_1.

End

Concatenate $S_1 \,||\, S_2 \rightarrow S$.

Output S.

4. Experimental results

To demonstrate the applicability of the proposed data hiding scheme, we use the platform of Microsoft Windows running on a personal computer. The data embedding and extraction algorithms are programmed with MATLAB language. Four test primary cover images[27] are shown in **Fig. 11**, which are (a) "Tiger" sized 1600×1792, (b) "Cat" sized 2288×3008, (c) "Sphinx" sized 3200×2080, and (d) "Squirrel" sized 3200×2144.

4.1 Main performance evaluations

The experimental results of "Tiger" are shown in **Fig. 12**, where **(a)** is the mosaic image produced by the method introduced in Section 3.1, **(b)** is the stego image produced by the first-layer data embedding, **(c)** to **(f)** are the final stego-images with different embedding rates in the second-layer. The applied embedding rates are 1.5, 2, 2.5, and 3 bits per pixel (bpp), respectively, i.e., the octagon-shaped shell matrices of 2^3, 2^4, 2^5, and 2^6-ary versions are applied in the reference matrix-based data hiding scheme. A zooming-in version of **Fig. 12** is given in **Fig. 13**. The region of interest is focused on an eye of the tiger. It is obvious that every details of the figures are constituted by the mosaic image tiles of coffee beans.

Tiger: 1600 × 1792

Cat: 2288 × 3008

Sphinx: 3200 × 2080

Squirrel: 3200 × 2144

Fig. 11. Four test images in our experiment.

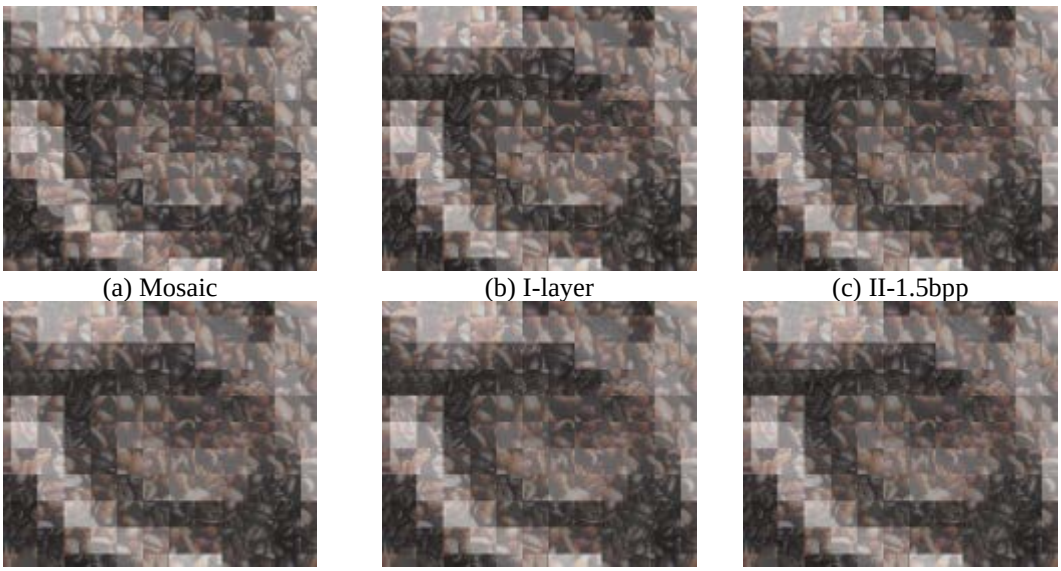
(a) Mosaic

(b) I-layer

(c) II-1.5bpp

(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

Fig. 12. The experimental results of "Tiger": (a) the produced mosaic image, (b) the stego-image with first-layer data embedded, and (c) to (f) the final stego-images with different embedding rates in the second-layer.



(a) Mosaic (b) I-layer (c) II-1.5bpp

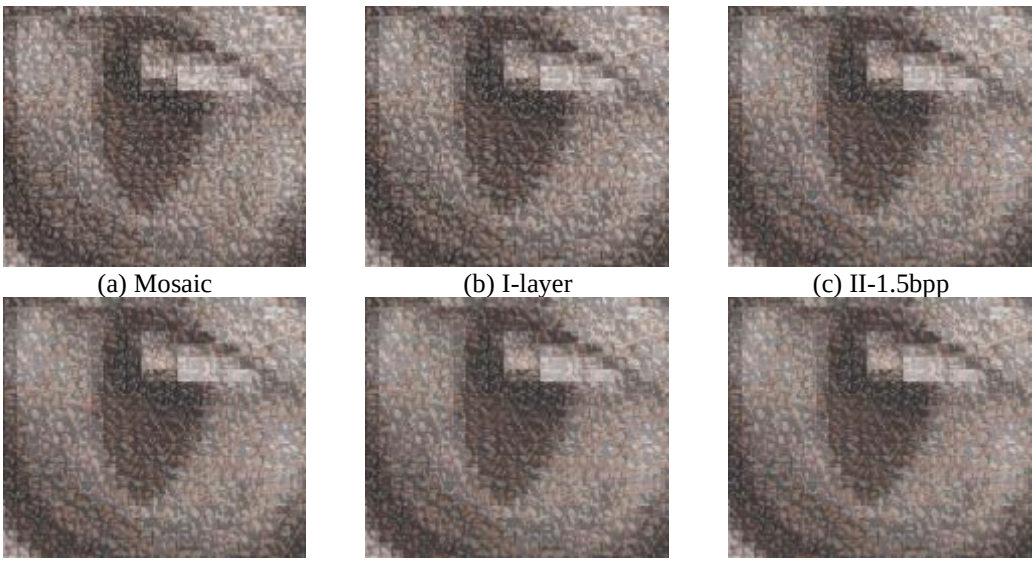
(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

Fig. 13. A zooming-in version of **Fig. 12** with the region of interest on an eye of the tiger.

(a) Mosaic (b) I-layer (c) II-1.5bpp

(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

Fig. 14. The experimental results of "Cat": (a) the produced mosaic image, (b) the stego-image with first-layer data embedded, and (c) to (f) the final stego-images with different embedding rates in the second-layer.



(a) Mosaic (b) I-layer (c) II-1.5bpp

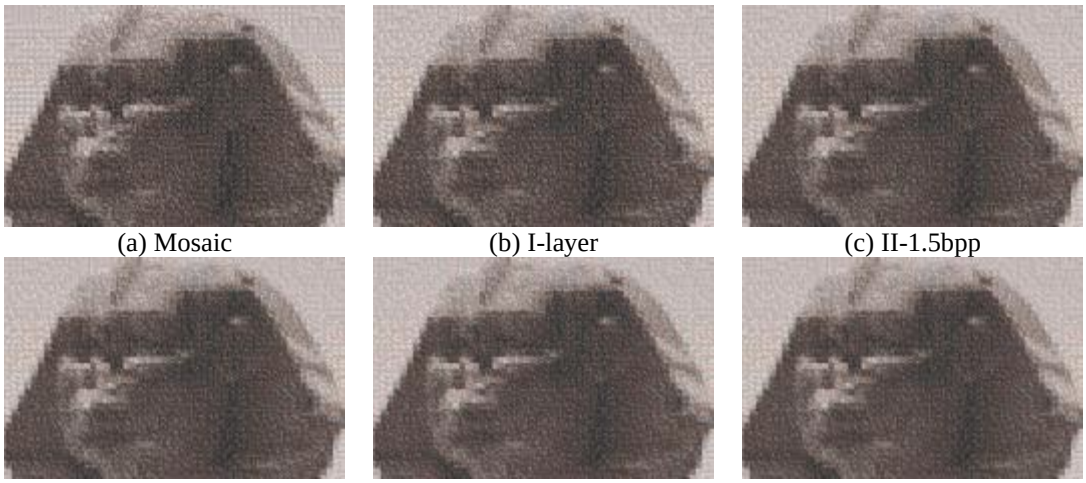
(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

Fig. 15. A zooming-in version of **Fig. 14** with the region of interest on an eye of the cat.

(a) Mosaic (b) I-layer (c) II-1.5bpp

(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

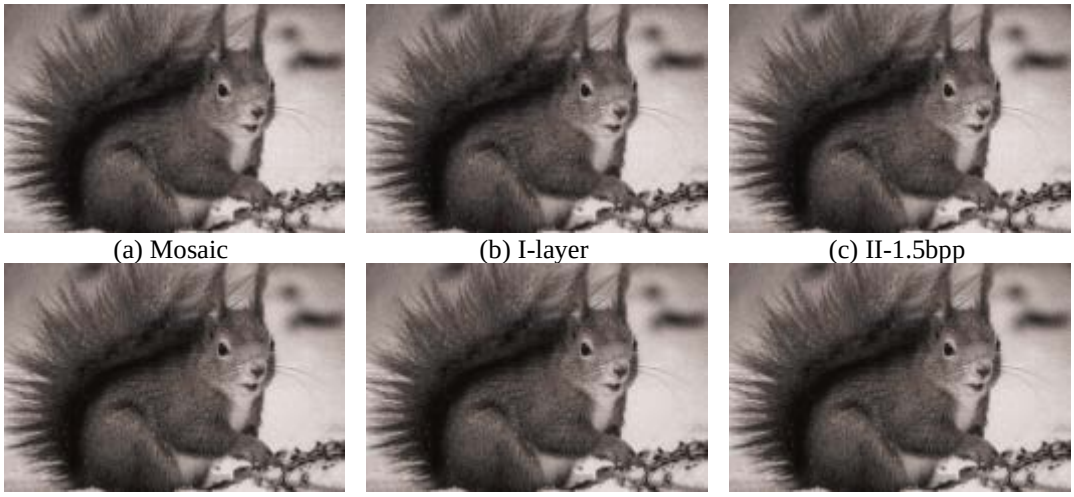
Fig. 16. The experimental results of "Sphinx": (a) the produced mosaic image, (b) the stego-image with first-layer data embedded, and (c) to (f) the final stego-images with different embedding rates in the second-layer.



(a) Mosaic (b) I-layer (c) II-1.5bpp

(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

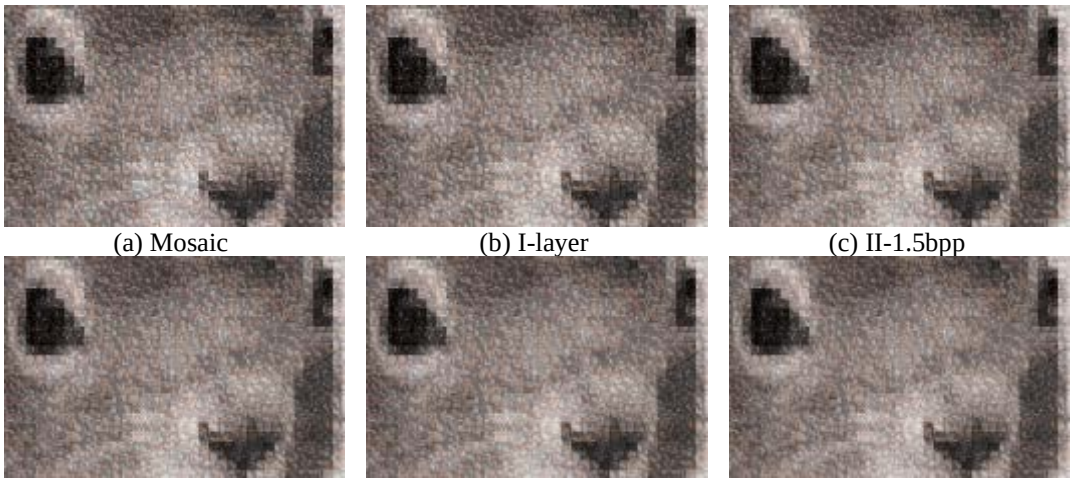
Fig. 17. A zooming-in version of **Fig. 16** with the region of interest on the head of Sphinx.



(a) Mosaic (b) I-layer (c) II-1.5bpp

(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

Fig. 18. The experimental results of "Squirrel": (a) the produced mosaic image, (b) the stego-image with first-layer data embedded, and (c) to (f) the final stego-images with different embedding rates in the second-layer.



(a) Mosaic (b) I-layer (c) II-1.5bpp

(d) II-2bpp (e) II-2.5bpp (f) II-3bpp

Fig. 19. A zooming-in version of **Fig. 18** with the region of interest on the face of the squirrel.

Similar results are obtained for the rest of three primary cover images, which are given in **Figs. 14** to **19**. The differences between the mosaic images of different embedding rates are imperceptible. To assess the quality of the stego-images, the peak signal to noise ratio (PSNR) is applied as the index. The PSNR is defined by

$$PSNR = 10\log_{10}\frac{255^2}{MSE}(\text{dB}), \tag{3}$$

$$MSE = \frac{1}{W \times H}\sum_{j=1}^{W}\sum_{i=1}^{H}(p'(i,j) - p(i,j))^2, \tag{4}$$

where $p'(i,j)$ and $p(i,j)$ represent the pixels of the stego-image and the cover mosaic image, respectively; W and H are the width and height.

The PSNR values of the mimic mosaic image and the first-layer data embedded stego-images are listed in **Table 1**. Note that, since the mosaic image tiles have only a single tone in color, the PSNR values are all measured with luminance of the images. As shown in the table, the PSNR values just slightly degrade after first-layer embedding for all of the four cases.

The PSNR values of the final stego-images are listed in **Table 2**, where PSNR values for four different embedding rates of the second-layer are given. It is interesting that the PSNR value gradually rises with the increasing embedding rate. The evolution of PSNR value from the first-layer embedding to the second-layer embedding with four different embedding rates is plotted in **Fig. 20**, where the evolutionary curves for all of the four mosaic cover images are given. **Table 3** lists the PSNR values for conventional reference matrix-based data hiding methods under different embedding rates. As the embedding rate raises from 1.5 to 3.0 bits per pixel, the PSNR value degrades about 10 dB. Since a wider search range is required to embed more secret bits in a cover pixel pair, the resulting modification of pixel value is greater. However, the embedding of our scheme is based on a mimic mosaic image. Under strong embedding, pixel values are modified in a way toward their original values. Therefore, the PSNR value increases with embedding rate. But this phenomenon gradually saturates at about 3 bits per pixel in our experiment. The total embedding capacity of the four mosaic cover images with different embedding rates are given in **Table 4**.

Table 1. The PSNR values, in dB, of the first-layer data embedded stego images.

Images	Mosaic	I-layer
Tiger	20.12	20.10
Cat	19.30	19.29
Sphinx	20.93	20.90
Squirrel	20.71	20.71

Table 2. The PSNR values of the final stego images with different embedding rates.

ER of II-layer	1.5bpp	2bpp	2.5bpp	3bpp
Tiger	20.90	21.27	21.55	21.67
Cat	20.01	20.35	20.61	20.73
Sphinx	21.77	22.17	22.47	22.57
Squirrel	21.56	21.95	22.23	22.33

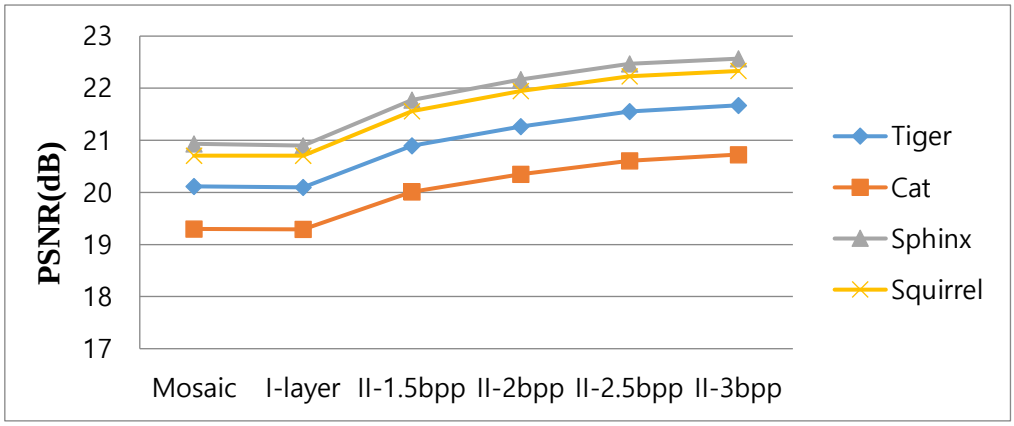


Fig. 20. PSNR values of the stego-images with different embedding rates.

Table 3. PSNR values of reference matrix-based methods under different embedding rates.

Method	PSNR	ER	Method	PSNR	ER
Lin et al.'s [16]	40.99	3	Liao et al.'s [15]	46.95	2
Hong's [14]	40.92	3	Hong's [14]	46.75	2
Leng's [11]	43.94	2.5	He et al.'s [13]	46.37	2
Xie et al.'s [8]	43.05	2.5	Lin et al.'s [16]	50.17	1.5
Lin et al.'s [16]	47.11	2	Chang et al.'s [4]	49.4	1.5

Table 4. Total embedding capacity for the four mosaic images with different embedding rates of the second-layer.

ER of II-layer	1.5bpp	2bpp	2.5bpp	3bpp
Tiger	4,317,600	5,751,200	7,184,800	8,618,400
Cat	10,363,782	13,804,934	17,246,086	20,687,238
Sphinx	10,023,000	13,351,000	16,679,000	20,007,000
Squirrel	10,331,400	13,761,800	17,192,200	20,622,600

4.2 Detailed data and parameter settings

In the second layer of data embedding, the octagon-shaped shell matrix [11] should be constructed according to the predefined embedding rate. The parameter settings for different embedding rates with their corresponding radix numbers are listed in **Table 5**, each provided parameter set (m, n, k) gives the best PSNR performance in comparison with the other octagon solutions of the same radix number system.

For the strong embedding (SE), the size of search mask is enlarged to allow a more flexible modification of pixel values. The experimental sizes of the enlarged masks for different embedding rates are given in **Table 6**.

Table 5. Parameter settings of octagon-shaped shell matrix for different ERs.

Radix	2^3	2^4	2^5	2^6
ER of II-layer	1.5bpp	2bpp	2.5bpp	3bpp
Parameters (m, n, k)	(3, 4, 1)	(4, 5, 1)	(6, 6, 1)	(8, 11, 3)

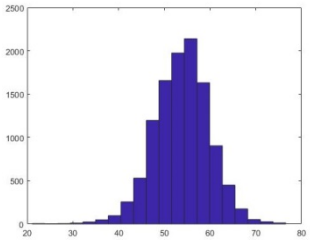
Table 6. Sizes of search mask for the SE with different ERs.

ER of II-layer	**1.5bpp**	**2bpp**	**2.5bpp**	**3bpp**
Size of search mask	5×5	7×7	9×9	11×11

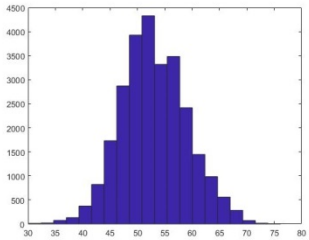
Upon embedding the second layer of secret data, the embedding strategy is decided by the positive difference (PD) rate of the image tile under processing. The actual distributions of PD rates for different mosaic cover images are plotted in **Fig. 21**. It is advantageous that most of the image tiles are populated around the PD rate of 50%, since we can apply the SE for all pixel pairs in this tile and effectively improve the visual quality of the stego-image. **Table 7** shows the type distributions of the image tiles. Over ninety percent of the image tiles belong to the type I for all of the four images. That is why the stego-image quality is improved after the second-layer of embedding.

Table 7. Type distribution of image tiles in percentage.

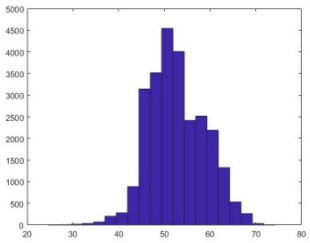
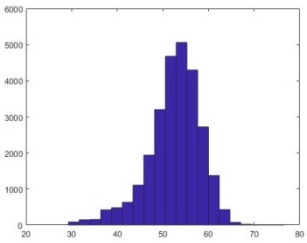
Type (image tile)	**II-**	**I**	**II+**
Tiger	0.9	92.3	6.6
Cat	0.5	93.1	6.4
Sphinx	0.7	92.7	6.6
Squirrel	1.9	96.6	1.5



(a) Tiger



(b) Cat

(c) Sphinx (d) Squirrel

Fig. 21. Distributions of the PD rates, in percentage, for the four mosaic cover images.

After the second-layer of data embedding, overflow or underflow of the average luminance is frequently encountered. The percentage of image tiles that result in overflow or underflow of luminance are listed in **Tables 8** to **11** for different ERs. All values are represented in percentage with respect to the total image tiles of the whole mosaic image. As ER increases, the percentage of violating tiles also increases. It seems to reach a saturation at fifty percent. Details of the compensation strategies are listed in **Table 12**. For example, when the current image tile under processing belongs to Type I and the average luminance after embedding is below the lowest bound, we compensate by consecutively alternating the SE of $(-,-)$ pixel pairs with WE until the luminance is back to the formal range. In our experiments, all of the stego-image tiles can be adjusted to the formal state by leveraging the proposed compensation strategy. In the worst case, the CE can always preserve the average luminance of an image tile and the steganography can be executed without exception.

Table 8. Over and Under-flows of average luminance for the ER of 1.5bpp.

II-1.5bpp	**Type**	**II-**		**I**		**II+**	
		details	total	details	total	details	total
Tiger	Over	0.20	0.21	32.63	39.17	0.09	2.04
	Under	0.01		6.54		1.95	
Cat	Over	0.07	0.08	29.49	38.12	0.13	1.76
	Under	0.01		8.63		1.63	
Sphinx	Over	0.15	0.18	29.42	38.19	0.76	2.69
	Under	0.03		8.77		1.93	
Squirrel	Over	0.21	0.26	34.48	41.55	0.06	0.5
	Under	0.05		7.07		0.44	

Table 9. Over and Under-flows of average luminance for the ER of 2bpp.

II-2bpp	**Type**	**II-**		**I**		**II+**	
		details	total	details	total	details	total
Tiger	Over	0.41	0.41	40.88	49.43	0.01	4.05
	Under	0.00		8.55		4.04	
Cat	Over	0.17	0.17	35.99	47.56	0.004	3.42
	Under	0.00		11.57		3.42	
Sphinx	Over	0.29	0.29	34.66	45.59	0.01	3.57
	Under	0.00		10.93		3.56	
Squirrel	Over	0.68	0.68	40.04	48.57	0.00	0.85
	Under	0.00		8.53		0.85	

Table 10. Over and Under-flows of average luminance for the ER of 2.5bpp.

II-2.5bpp	Type	II-		I		II+	
		details	total	details	total	details	total
Tiger	Over	0.35	0.38	47.29	56.03	0.04	3.70
	Under	0.03		8.74		3.66	
Cat	Over	0.17	0.18	41.88	53.66	0.10	2.96
	Under	0.01		11.78		2.86	
Sphinx	Over	0.27	0.29	38.74	50.90	0.07	3.82
	Under	0.02		12.16		3.75	
Squirrel	Over	1.18	1.19	48.93	56.55	0.03	0.67
	Under	0.01		7.62		0.64	

Table 11. Over and Under-flows of average luminance for the ER of 3bpp.

II-3bpp	Type	II-		I		II+	
		details	total	details	total	details	total
Tiger	Over	0.14	0.35	44.19	45.18	0.51	2.66
	Under	0.21		0.99		2.15	
Cat	Over	0.06	0.22	41.11	54.43	0.84	2.80
	Under	0.16		13.32		1.96	
Sphinx	Over	0.10	0.24	33.94	49.03	0.53	3.68
	Under	0.14		15.09		3.15	
Squirrel	Over	1.49	1.54	48.41	56.98	0.23	0.47
	Under	0.05		8.57		0.24	

Table 12. Compensation strategy for different situations.

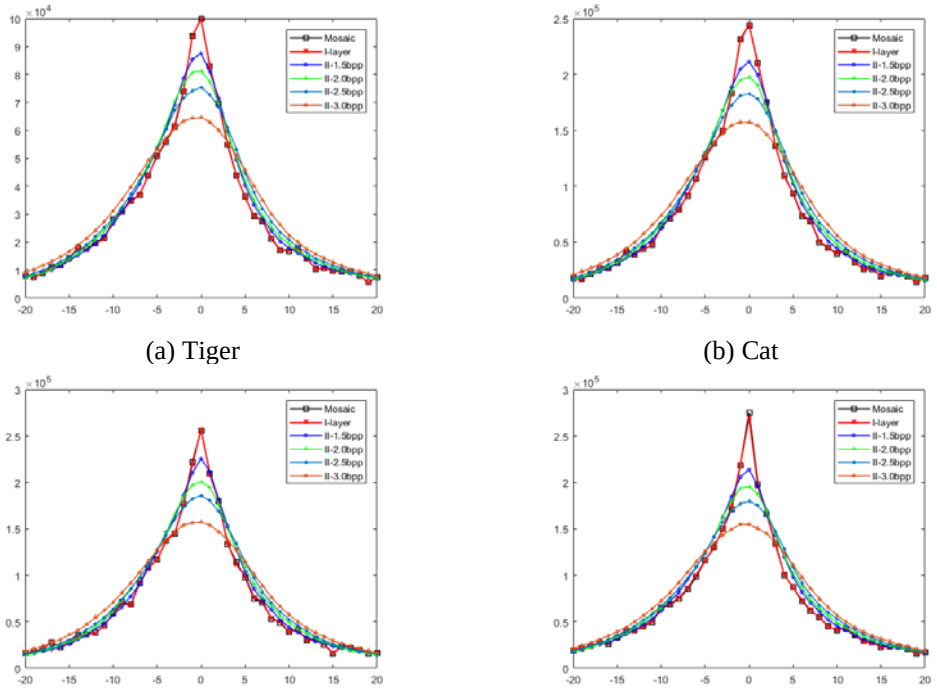
Tile type	Situation	Default	Compensation Strategy
Type II-	OverFlow	$(+,+)$ SE	$(+,+) \rightarrow$ WE
	UnderFlow	$(-,-)$ WE	$(-,-) \rightarrow$ CE
Type I	OverFlow	$(+,+)$ SE	$(+,+) \rightarrow$ WE
	UnderFlow	$(-,-)$ SE	$(-,-) \rightarrow$ WE
Type II+	OverFlow	$(+,+)$ WE	$(+,+) \rightarrow$ CE
	UnderFlow	$(-,-)$ SE	$(-,-) \rightarrow$ WE

4.3 Security analysis

To assess the security level of the proposed data hiding scheme, many steganalysis, including machine learning-based [30], deep learning-based [31], and pixel-value differencing [32], techniques are available. The reference matrix-based data hiding schemes usually divide cover image into pixel pairs and embed secret data in a pair-wise manner. The pixel-value differencing histogram steganalysis [32] is devised specifically for analyzing such data hiding schemes.

The pixel-value differencing histogram (PDH) for four series of stego images produced by our data hiding scheme are plotted in **Fig. 22**. In each figure, PDH for the mosaic cover image, the stego images of the first-layer embedding and the second-layer embedding of different embedding rates are given. Notice that the curves of the cover mosaic image and the stego image of the first-layer embedding are exactly aligned with each other. Since the first-layer

embedding only involves block-wise replacement of mosaic tiles, the differences between neighboring pixel values are preserved. For the stego images of complete (two-layers) embedding, the curve is gradually flattened as the embedding rate increases. This phenomenon is exactly the same as conventional reference matrix-based methods.



(a) Tiger (b) Cat

(c) Sphinx (d) Squirrel

Fig. 22. Pixel-value differencing histogram for the four series of stego images.

5. Conclusions

We use three images of coffee beans to generate a library of image tiles. The generating process includes partition in the space domain and linear transformation of pixel values. Then, the image tiles are applied to mimic the given primary cover images and produce the mosaic images.

A two-layer image steganography is proposed to hide secret data in the mosaic images. In the first layer of embedding, turtle shell matrix-based data hiding scheme is applied to the tile level of the cover image. Secret data is embedded by alternating between the image tiles of close luminance. In the second layer of embedding, octagon-shaped shell matrix is applied to hide secret data in the pixel level. The embedding rate can be adjusted by leveraging differently sized octagons.

We define a polarized search mask with the SE and WE techniques to make the modification of pixel values more flexible. In addition, a compensation strategy is designed to preserve the luminance order of image tiles from being disturbed, which is essential information for the extraction of first-layer data.

Experimental results show that the proposed mosaic production process can successfully mimic the given primary images. Based on the low PSNR values of the mosaic cover images, a large amount of secret data can be embedded. In the second layer of embedding, the PSNR

value of the stego-image with respect to the primary cover image increases with the increasing of embedding rate and reaches a saturation at 3 bpp. We hope this article can open a new direction of applying steganography to the mosaic images. Besides, we believe in the future, there will be more and more studies which present novel and promising methods in this issue.

References

[1] X. Zhang, S. Wang, "Efficient steganographic embedding by exploiting modification direction," *IEEE Communications Letters*, vol. 10, no. 11, pp. 781-783, Nov. 2006. Article (CrossRef Link)

[2] H. J. Kim, C. Kim, Y. Choi, et al., "Improved modification direction methods," *Comput. Math.Appl*, vol. 60, no. 2, pp. 319–325, 2010. Article (CrossRef Link)

[3] T. D. Kieu, C. C. Chang, "A steganographic scheme by fully exploiting modification directions," *Expert system with applications*, vol. 38, no. 8, pp. 10648-10657, 2011. Article (CrossRef Link)

[4] C. C. Chang, Y. Liu, T. S. Nguyen, "A novel turtle shell-based scheme for data hiding," in *Proc. of 10th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process. (IIHMSP)*, Kitakyushu, Japan, pp. 89–93, Aug. 2014. Article (CrossRef Link)

[5] Y. Liu, C. C. Chang, T. S. Nguyen, "High capacity turtle shell-based data hiding," *IET Image Process*, vol. 10, no. 2, pp. 130–137, 2016. Article (CrossRef Link)

[6] Q. Jin, Z. Li, C. C. Chang, et al., "Minimizing turtle-shell matrix based stego image distortion using particle swarm optimization," *Int. J. Netw. Secur.*, vol. 19, no. 1, pp. 154–162, 2017. Article (CrossRef Link)

[7] L. Liu, C. C. Chang, A. Wang, "Data hiding based on extended turtle shell matrix construction method," *Multimedia Tools Appl*, vol. 76, pp. 12233–12250, 2017. Article (CrossRef Link)

[8] X. Z. Xie, C. C. Lin, C. C. Chang, "Data hiding based on a two-layer turtle shell matrix," *Symmetry*, vol. 10, no. 2, pp. 47, 2018. Article (CrossRef Link)

[9] J. H. Horng, J. Lin, Y. J. Liu, C. C. Chang, "3D Multilayered Turtle Shell Models for Image Steganography," *Computer Modeling in Engineering & Sciences*, vol. 125, no. 2, pp. 879-906, 2020.Article (CrossRef Link)

[10] S. Kurup, A. Rodrigues, A. Bhise, "Data hiding scheme based on octagon shaped shell," in *Proc. of International Conference on Advances in Computing, Communication and Informatics*, Kerala, India, pp. 1982-1986, 2015. Article (CrossRef Link)

[11] H. S. Leng, "Generalized scheme based on octagon-shaped shell for data hiding in steganographic applications," *Symmetry*, vol. 11, no. 6, pp. 1-10, 2019. Article (CrossRef Link)

[12] C. C. Chang, Y. C. Chou, T. D. Kieu, "An information hiding scheme using Sudoku," in *Proc. of 3rd Int. Conf. Innov. Comput. Inf. Control (ICICIC)*, Dalian, China, pp. 17–22, Jun. 2008. Article (CrossRef Link)

[13] M. Z. He, Y. J. Liu, C. C. Chang, et al., "A mini Sudoku matrix-based data embedding scheme with high payload," *IEEE Access*, vol. 7, pp. 141414-141425, 2019. Article (CrossRef Link)

[14] W. Hong, "Adaptive image data hiding in edges using patched reference table and pair-wise embedding technique," *Information Sciences*, vol. 221, pp. 473–489, 2013. Article (CrossRef Link)

[15] X. Liao, S. Guo, J. Yin, H. Wang, X. Li, A. K. Sangaiah, "New cubic reference table based image steganography," *Multimed Tools Appl*, vol. 77, pp. 10033–10050, 2018. Article (CrossRef Link)

[16] J. Lin, J. H. Horng, Y. Liu and C. C. Chang, "An Anisotropic Reference Matrix for Image Steganography," *Journal of Visual Communication and Image Representation*, Vol. 74, 102969, January 2021. Article (CrossRef Link)

[17] T. S. Nguyen, C. C. Chang, "A reversible data hiding scheme based on the Sudoku technique," *Displays*, vol. 39, pp. 109-116, 2015. Article (CrossRef Link)

[18] C. C. Chang, C. T. Li, and K. Chen, "Privacy-preserving reversible information hiding based on arithmetic of quadratic residues," *IEEE Access*, vol. 7, no. 1, pp. 54117-54132, 2019. Article (CrossRef Link)

[19] C. C. Chang, C. T. Li, Y. Q. Shi, "Privacy-aware reversible watermarking in cloud computing environments," *IEEE Access*, vol. 6, no. 1, pp. 70720-70733, 2018. Article (CrossRef Link)
[20] Y. Shi, X. Li, X. Zhang, H. Wu and B. Ma, "Reversible data hiding: Advances in the past two decades," *IEEE Access*, vol. 4, pp. 3210-3237, 2016. Article (CrossRef Link)
[21] C. N. Yang, T. S. Chen, K. H. Yu and C. C. Wang, "Improvements of image sharing with steganography and authentication," *Journal of Systems and Software*, vol. 80, no. 7, pp. 1070-1076, 2007. Article (CrossRef Link)
[22] C. C. Chang, C. T. Li, "Algebraic secret sharing using privacy homomorphisms for IoT-based healthcare systems," *Math. Biosci. Eng*, vol. 16, no. 5, pp. 3367-3381, 2019. Article (CrossRef Link)
[23] K. Gao, J. H. Horng, Y. J. Liu, and C. C. Chang, "A Reversible Secret Image Sharing Scheme Based on Stick Insect Matrix," *IEEE Access*, vol. 8, pp. 130405-130416, 2020. Article (CrossRef Link)
[24] J. H. Horng, C. C. Chang, G. L. Li, "Steganography using Quotient Value Differencing and LSB Substitution for AMBTC Compressed Images," *IEEE Access*, vol. 8, pp. 129347-129358, 2020. Article (CrossRef Link)
[25] F. Huang, X. Qu, H. J. Kim and J. Huang, "Reversible data hiding in JPEG images," *IEEE Transactions on Circuits Systems for Video Technology*, vol. 26, no. 9, pp.1610–1621, 2016. Article (CrossRef Link)
[26] Y. Li, S. Yao, K. Yang, et al., "A High-imperceptibility and histogram-shifting data hiding scheme for JPEG images," *IEEE Access*, vol. 7, pp. 73573-73582, 2019. Article (CrossRef Link)
[27] https://www.pexels.com/zh-tw/.
[28] https://unsplash.com/.
[29] https://www.kdccoffee.com/.
[30] J. Fridrich and J. Kodovsky, "Rich Models for Steganalysis of Digital Images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868-882, June 2012. Article (CrossRef Link)
[31] Dengpan, Y., Shunzhi, J., Shiyu, L. et al., "Faster and transferable deep learning steganalysis on GPU," *J Real-Time Image Proc*, 16, 623–633, 2019. Article (CrossRef Link)
[32] Zhang, X., Wang, S., "Vulnerability of pixel-value differencing steganography to histogram analysis and modification for enhanced security," *Pattern Recognition Letters*, vol. 25, no. 3, 331-339, 2004. Article (CrossRef Link)

Ji-Hwei Horng received a B.S. degree from the Department of Electronic Engineering, Tamkang University, Taipei, Taiwan in 1990 and M.S. and Ph.D. degrees from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan in 1992 and 1996, respectively. He was a professor and Chairman of the Department of Electronic Engineering from 2006 to 2009 and the Dean of the College of Science and Engineering from 2011 to 2014 at National Quemoy University (NQU), Kinmen, Taiwan. Currently, he is the Vice President of Academic Affairs in NQU. His research interests include image processing, pattern recognition, information security, and artificial intelligence.

Chin-Chen Chang received a B.S. degree in Applied Mathematics, an M.S. degree in Computer and Decision Sciences from National Tsing Hua University, and a Ph.D. degree in Computer Engineering from National Chiao Tung University. He was with the National Chung Cheng University from 1989 to 2005. He is currently the Chair Professor of the Department of Information Engineering and Computer Science, Feng Chia University. Prior to joining Feng Chia University, he was an Associate Professor with Chiao Tung University, a Professor with National Chung Hsing University, and a Chair Professor at National Chung Cheng University. He has also been a Visiting Researcher at Tokyo University, Japan, and a Visiting Scientist at Kyoto University, Japan. During his service in Chung Cheng, he served as the Chairman of the Institute of Computer Science and Information Engineering, Dean of College of Engineering, Provost, and then the Acting President of Chung Cheng University and the Director of Advisory Office in Ministry of Education, Taiwan. He has won many research awards and has honorary positions in prestigious organizations both nationally and internationally. He is currently a Fellow of IEEE, U.K. Since his early years of career development, he has consecutively won awards, including the Outstanding Talent in Information Sciences of the R.O.C., the Acer Dragon Award of the Ten Most Outstanding Talents, the Outstanding Scholar Award of the R.O.C., the Outstanding Engineering Professor Award of the R.O.C., the Distinguished Research Awards of National Science Council of the R.O.C., the Top Fifteen Scholars in Systems and Software Engineering of the Journal of Systems and Software, and so on. On numerous occasions, he has been invited to serve as a Visiting Professor, Chair Professor, Honorary Professor, Honorary Director, Honorary Cairman, Distinguished Alumnus, Distinguished Researcher, and Research Fellow by various universities and research institutes. His current research interests include database design, computer cryptography, image compression, and data structures.

Kun-Sheng Sun received the B.S. degree in Department of Electronic Engineering from National Quemoy University, Kinmen, Taiwan, in 2020, where he is currently pursuing the M.S. degree. His current research interests include information security and information hiding.