Constructive Steganography Using Texture Synthesis

Zhenxing Qian, Nannan Huang, Sheng Li and Xinpeng Zhang

Shanghai Institute for Advanced Communication and Data Science, School of Communication & Information Engineering, Shanghai University, Shanghai 200444, People’s Republic of China

ABSTRACT
This paper proposes a constructive steganography approach using texture synthesis. With a source texture pattern, we first generate a reference pattern using a point-based texture synthesis. The reference pattern is further extended using a patch-based synthesis algorithm. We crop the extended pattern to generate an intermediate pattern. The reference pattern is then divided into overlapping patches. With a secret key, the reference patches are pseudo-randomly painted onto a blank canvas. To hide secret messages, we identify appropriate patches from the intermediate pattern to represent secret bits, and fulfill the blank regions with these patches using image quilting. The proposed method has a better performance than previous works. As we extend the source pattern to generate more candidate patches, more secret bits can be hidden into the stego image, and the quality of the synthesized image is better. Meanwhile, the proposed method has a capability of resisting a specified attack designed for texture synthesis-based steganography.

KEYWORDS
Covert transmission; Data hiding; Information hiding; Steganography; Texture synthesis; Watermarking

1. INTRODUCTION

The word steganography is a Greek word which means the covered writing. Steganography has a long history of over 2500 years since the birth. The earliest record of steganography can be traced back to ancient Greece such as writing messages on a wooden board and then waxing it to hide information. Steganography is a technology that conceals information into digital carriers to realize the covert transmissions [1,2]. Modern research on steganography began with the famous “prisoner’s problem” raised by Simmons. Since the late 1990s, steganography technique has attracted much attention of the academia.

Generally, a steganography system includes two parts: messages embedding and messages extraction. A sender uses an embedding algorithm to hide secret messages into a public multimedia to generate a stego, and transmits the stego to a receiver through a public channel. A receiver uses the extraction algorithm to recover the secret messages from the stego. In the past decades, many steganography methods have been proposed to embed information into images. LSB replacement [3] is a common and simple approach to embed information into an image. This algorithm replaces the least significant bits of the image coefficients with the secret bits. F5 is an algorithm that embeds secret information by reducing the absolute value of the DCT coefficient [4], in which the matrix coding and random scrambling are used to embed the secret information evenly into different cells. ZZW proposed a binary coding method by combining matrix coding and wet paper code, which can approach the theoretical bound of data hiding [5]. Filler et al. proposed an STC encoding framework that uses various distortion functions [6]. For a defined distortion function and the secret messages, the STC code can minimize the embedded distortion and enable a receiver extracting the messages without the synchronization distortion information. In these methods, secret messages are hidden into the images by modifying the pixels or coefficient values.

In recent years, a kind of steganography based on texture synthesis has been proposed. Different from the modification-based methods, some new approaches hide information during the procedures of image construction. We name this kind of data hiding as constructive steganography to distinguish it from the modification-based steganography. Pioneering works were done by Otori and Kuriyama [7,8]. They use the colored dots picked from a texture sample to represent the secret bits. After painting these dots onto a canvas, an image containing secret messages is generated by synthesizing the blank regions. Another constructive steganography algorithm was proposed in [9], in which the hidden messages written on a canvas are deformed to a texture image by a technique of Marbling [10]. In [11], Wu and Wang propose a steganography method using reversible
t e x t u r e s y n t h e s i s . T h i s m e t h o d p r o v i d e s a s m a l l e r l a r g e r embedding payload than previous works in [7] and [8]. Meanwhile, the extraction error can also be avoided. However, due to the mirror symmetry operations used during texture synthesis, an algorithm was proposed in [12] to combat this steganography. With this adversary method, the steganography behavior can be detected and the secret messages can be extracted. Another constructive steganography was proposed in [13], using texture synthesis to construct a robust data hiding. It is secure than the previous work at the expense of fewer payloads.

In this paper, we propose a new steganography method using texture synthesis. Different from the traditional works, we propose to construct an intermediate texture before data hiding. With the intermediate texture, more candidate patches can be used to carry more secret bits during texture synthesis. The proposed method has three achievements. First, the embedding payload is higher than previous arts. Second, the synthesized texture containing secret messages has a better quality. Third, the proposed approach has the capability of combating the attack proposed in [12]. This paper is organized as follows. In Section 2, we briefly introduce the related works. In Section 3, we detail the proposed method of constructive steganography. Section 4 provides the experimental results. The whole paper is concluded in Section 5.

2. RELATED WORKS

Texture synthesis is a technology that uses a texture pattern with limited contents to generate a larger sized image with similar appearance. The exemplar-based texture synthesis is a popular idea that resamples the source pattern to produce a synthesized image [14–16]. In [14] and [15], pixels in the source pattern are used to propagate the synthesis content. The pixel by pixel algorithm requires a lot of time to synthesize each region. In [16], Efros and Freeman propose a fast synthesis algorithm named “image quilting”. This approach constructs the new texture by finding appropriate patches from the source pattern. Accordingly, the neighboring patches are quilted by an optimal cut. An example of texture synthesis using the method in [16] is shown in Figure 1, in which (a) is a source pattern and (b) a synthesized image with a larger size.

Inspired by these methods, some steganography algorithms based on texture synthesis emerge. One representative work was presented by Wu and Wang in [11]. The main idea is to hide secret messages during the procedure of texture synthesis. With a source texture, the data hider first divides the pattern into non-overlapping blocks, which are named as kernels. Each kernel is then expanded into a patch, containing the kernel region and a boundary region. In the process of expansion, the boundary pixels are directly extracted from the source pattern. If a kernel is located at the boundary of the source texture, it should be expanded by mirror symmetry.

The data hider further pastes all kernels onto a blank canvas that is to be synthesized. The pasting positions are identified by a secret key. Next, the data hider uses the algorithm proposed in [16] to synthesize the rest regions in blank. Meanwhile, the secret messages are hidden during the process. He/she divides the source pattern into overlapping blocks using the raster scan. These blocks are used as candidates for synthesis. When synthesizing each blank block, the data hider calculates the mean square error (MSE) of the overlapped region between a candidate and a synthesized patch, and further ranks the MSE values. Accordingly, the index of each candidate is used to represent several bits of the secret messages. Hence, the secret bits can be embedded into the synthesized texture after texture synthesis.

Data extraction is a reverse procedure of the embedding. With the secret key, the receiver first extracts the kernels from the stego image and reconstructs the original texture pattern. Subsequently, the patches and candidates can be divided from the source again. After calculating and indexing the MSE values, the secret bits can be extracted exactly.

The steganography approach in [11] can provide a large embedding capacity. However, there is a security limitation, which was found by Zhou et al. [12]. Since the kernels on the boundary of the source texture are expanded by mirror symmetry, Zhou et al. [12] found that the original texture can be recovered by an adversary even the
secret key is not available. After comparing the similarity of an expanded boundary area with an inside boundary area, the adversary can search patches in the synthetic image and identify the patches from the four corners and boundaries of source pattern. Therefore, the boundary patches of original pattern can be recovered by identifying the optimal patch from left to right and top to bottom. Once the adversary padded a ring-shaped boundary of the source image, the remaining patches can be identified and pasted until the full source texture is recovered. Using the same way of data hiding, the adversary can extract the hidden messages.

3. PROPOSED METHOD

3.1 General Framework

The general framework of the proposed constructive steganography is depicted in Figure 2. The data hider first chooses a small texture pattern \( P \) as the source image. This texture pattern is then extended to construct a larger pattern \( E \). The data hider first constructs a reference pattern \( R \), which contains the content of \( P \). The reference pattern is then divided into overlapping patches. With an encryption key \( K \) the data hider pseudo-randomly paints these patches onto a blank canvas to generate an initial image \( I \). The data hider further crops some content from the extended pattern \( E \) to generate an intermediate pattern \( C \). With the candidate patches from \( C \), the data hider hides the secret message \( M \) during the procedure of synthesizing the blank regions on image \( I \). Finally, a stego image \( S \) is generated.

3.2 Texture Extension

Given a source pattern \( P \), we first construct a reference pattern \( R \) by extending the boundaries of \( P \) using a pixel-based texture synthesis. Let the size of \( P \) be \( w \times h \). The original pattern \( P \) is extended to a texture pattern sized \( w' \times h' \), where \( w' = w + 2d, h' = h + 2d \), and \( d > 0 \). As shown in Figure 3, the original pattern \( P \) is first painted onto the center of the blank canvas.

With the pixel-based synthesis algorithm in [15], we paint the blank region on the canvas. To synthesize each point \( o \) on the blank boundary, we choose a block \( \pi(o) \) around this point, in which \( o \) is centered at \( \pi(c) \). Next, we find an optimal candidate block \( \pi_b \) from the original pattern \( P \) using Equation (1), such that the block \( \pi(o) \) is close to the optimal candidate.

\[
\pi_b = \arg \min_{\pi_c} g \ast d(\pi(o), \pi_c).
\] (1)
In Equation (1), $g$ is a Gaussian kernel, “*” the convolution operator, $d(\cdot)$ the function calculating the Euclid distance, and $\pi_b$ the optimal candidate block. The central pixel in $\pi_b$ is then painted onto the position of $o$. After synthesizing all pixels on the blank region, we obtain a reference pattern $R$.

We further create a new blank canvas and paint the reference $R$ onto the center of the canvas. Denote the size of the new canvas as $w_c \times h_c$, where $w_c$ and $h_c$ are much larger than $w'$ and $h'$, respectively. Next, we divide the original pattern $P$ into overlapping patches in raster scan order. Let the sizes of each patch be $a \times b$. Hence, there are $m = (w - a + 1)(h - b + 1)$ patches. We use these patches as candidates for patch-based texture synthesis. With the block-based synthesis algorithm in [16], we fulfill the blank regions on the new canvas using the candidates. For every location of the blank regions, we search from all $m$ candidates for an optimal one.

For example, when synthesizing the blank region next to a painted block $A$ (the left rectangle), as shown in Figure 4(a), we find a patch $B$ (the right rectangle) from $m$ candidates such that the mean square error of the overlap (MSEO) is the smallest. The MSEO between two patches can be calculated by

$$\text{MSEO}(A, B) = \sum_{c=r,g,b} \| A_c^o - B_c^o \|^2 / 3,$$

where $A_o$ and $B_o$ represent the overlapped regions in $A$ and $B$, respectively, $c$ stands for the channels red, green, and blue of the overlapped regions, and $\| \cdot \|^2$ is the $L_2$ norm operator.

Subsequently, we calculate the cumulative minimum error on the overlap using Equation (1) in [16] to find the minimum cost path, i.e. the boundary cut. This path of the cut is used as the boundary of a new patch, which is then pasted onto the blank region. As the example shown in Figure 4(b), the content in $B$ on the right hand of the boundary cut forms a new patch. After synthesizing all blank regions in the new canvas, we generate the extended pattern $E$.

In Equation (1), $g$ is a Gaussian kernel, “*” the convolution operator, $d(\cdot)$ the function calculating the Euclid distance, and $\pi_b$ the optimal candidate block. The central pixel in $\pi_b$ is then painted onto the position of $o$. After synthesizing all pixels on the blank region, we obtain a reference pattern $R$.

We further create a new blank canvas and paint the reference $R$ onto the center of the canvas. Denote the size of the new canvas as $w_c \times h_c$, where $w_c$ and $h_c$ are much larger than $w'$ and $h'$, respectively. Next, we divide the original pattern $P$ into overlapping patches in raster scan order. Let the sizes of each patch be $a \times b$. Hence, there are $m = (w - a + 1)(h - b + 1)$ patches. We use these patches as candidates for patch-based texture synthesis. With the block-based synthesis algorithm in [16], we fulfill the blank regions on the new canvas using the candidates. For every location of the blank regions, we search from all $m$ candidates for an optimal one.

For example, when synthesizing the blank region next to a painted block $A$ (the left rectangle), as shown in Figure 4(a), we find a patch $B$ (the right rectangle) from $m$ candidates such that the mean square error of the overlap (MSEO) is the smallest. The MSEO between two patches can be calculated by

$$\text{MSEO}(A, B) = \sum_{c=r,g,b} \| A_c^o - B_c^o \|^2 / 3,$$

where $A_o$ and $B_o$ represent the overlapped regions in $A$ and $B$, respectively, $c$ stands for the channels red, green, and blue of the overlapped regions, and $\| \cdot \|^2$ is the $L_2$ norm operator.

Subsequently, we calculate the cumulative minimum error on the overlap using Equation (1) in [16] to find the minimum cost path, i.e. the boundary cut. This path of the cut is used as the boundary of a new patch, which is then pasted onto the blank region. As the example shown in Figure 4(b), the content in $B$ on the right hand of the boundary cut forms a new patch. After synthesizing all blank regions in the new canvas, we generate the extended pattern $E$.

$$\text{MSEO}(A, B) = \sum_{c=r,g,b} \| A_c^o - B_c^o \|^2 / 3,$$

where $A_o$ and $B_o$ represent the overlapped regions in $A$ and $B$, respectively, $c$ stands for the channels red, green, and blue of the overlapped regions, and $\| \cdot \|^2$ is the $L_2$ norm operator.

Subsequently, we calculate the cumulative minimum error on the overlap using Equation (1) in [16] to find the minimum cost path, i.e. the boundary cut. This path of the cut is used as the boundary of a new patch, which is then pasted onto the blank region. As the example shown in Figure 4(b), the content in $B$ on the right hand of the boundary cut forms a new patch. After synthesizing all blank regions in the new canvas, we generate the extended pattern $E$.

$$\text{MSEO}(A, B) = \sum_{c=r,g,b} \| A_c^o - B_c^o \|^2 / 3,$$

where $A_o$ and $B_o$ represent the overlapped regions in $A$ and $B$, respectively, $c$ stands for the channels red, green, and blue of the overlapped regions, and $\| \cdot \|^2$ is the $L_2$ norm operator.

Subsequently, we calculate the cumulative minimum error on the overlap using Equation (1) in [16] to find the minimum cost path, i.e. the boundary cut. This path of the cut is used as the boundary of a new patch, which is then pasted onto the blank region. As the example shown in Figure 4(b), the content in $B$ on the right hand of the boundary cut forms a new patch. After synthesizing all blank regions in the new canvas, we generate the extended pattern $E$.

$$\text{MSEO}(A, B) = \sum_{c=r,g,b} \| A_c^o - B_c^o \|^2 / 3,$$

where $A_o$ and $B_o$ represent the overlapped regions in $A$ and $B$, respectively, $c$ stands for the channels red, green, and blue of the overlapped regions, and $\| \cdot \|^2$ is the $L_2$ norm operator.

Subsequently, we calculate the cumulative minimum error on the overlap using Equation (1) in [16] to find the minimum cost path, i.e. the boundary cut. This path of the cut is used as the boundary of a new patch, which is then pasted onto the blank region. As the example shown in Figure 4(b), the content in $B$ on the right hand of the boundary cut forms a new patch. After synthesizing all blank regions in the new canvas, we generate the extended pattern $E$.

$$\text{MSEO}(A, B) = \sum_{c=r,g,b} \| A_c^o - B_c^o \|^2 / 3,$$

where $A_o$ and $B_o$ represent the overlapped regions in $A$ and $B$, respectively, $c$ stands for the channels red, green, and blue of the overlapped regions, and $\| \cdot \|^2$ is the $L_2$ norm operator.

Subsequently, we calculate the cumulative minimum error on the overlap using Equation (1) in [16] to find the minimum cost path, i.e. the boundary cut. This path of the cut is used as the boundary of a new patch, which is then pasted onto the blank region. As the example shown in Figure 4(b), the content in $B$ on the right hand of the boundary cut forms a new patch. After synthesizing all blank regions in the new canvas, we generate the extended pattern $E$.
We use these patches as candidates to paint the \( m_s - m_r \) blank blocks in \( I \), and hide the secret messages during the process.

At each position of the blank block to be synthesized, we calculate MSEO between the painted patch with each one in \( \{p_1, p_2, \ldots, p_m\} \). Without loss of generality, we suppose there has been a painted block \( A \) on \( I \). When painting a blank block next to \( A \), we calculate the MSEO between \( A \) and each \( p_i \). Denote the maximum MSEO value as

\[
e_{\text{max}} = \text{maximize } \text{MSEO}(A, p_i), \quad i = 1, 2, \ldots, m,
\]

where MSEO stands for the MSE of the overlap.

As described in [16], we define an error tolerance \( \delta \), where \( 0 < \delta \leq 1 \). This parameter is used to generate diverse appearances for different synthesizers. With the parameter, we find \( l \) patches \( \{p_{s(1)}, p_{s(2)}, \ldots, p_{s(l)}\} \) from \( \{p_1, p_2, \ldots, p_m\} \), such that

\[
\text{MSEO}(A, p_{s(1)}) \leq \text{MSEO}(A, p_{s(2)}) \leq \cdots \leq \text{MSEO}(A, p_{s(l)}) \leq \delta \cdot e_{\text{max}}.
\]

These patches are used to represent secret bits. Since there are \( l \) candidates, we can hide \( \lceil \log_2 l \rceil \) bits when painting the blank block next to \( A \). According to the secret bits to be hidden, we choose the corresponding candidate from \( \{p_{s(1)}, p_{s(2)}, \ldots, p_{s(l)}\} \) and fulfill the blank block next to \( A \) using the boundary cut-based synthesis algorithm in [16].

In this way, the secret message \( M \) is hidden into the stego image \( S \) after sequentially synthesizing all blank blocks in \( I \). The synthesis of each blank block is realized by finding the boundary cut on the overlap. Assuming that each blank block on \( I \) has \( l_i \) candidates \( (i = 1, 2, \ldots, m_s - m_r) \), the embedding capacity is therefore equal to

\[
C_e = \sum_{i=1}^{m_s-m_r} \lceil \log_2 l_i \rceil.
\]

After receiving the stego image \( S \), a recipient can extract the hidden message using the shared key \( K \), which is a reverse process of data hiding. With the secret key, the recipient identifies \( m_r \) reference patches located inside the stego image. With these extracted patches, the recipient reconstructs the reference image \( R' \) using the way depicted in Figure 5. The image \( R' \) is slightly different from \( R \) on the borders because of image quality during data hiding. However, contents of the kernels in all reference patches do not change during data hiding.

Therefore, the original source texture \( P \) can be losslessly recovered.

Using the same algorithm depicted in the last subsection, the recipient can extend \( P \) into \( R \) and generate the image \( C \). The recipient further divides the \( C \) into overlapping blocks \( \{p_1, p_2, \ldots, p_m\} \) in raster scan order. From the stego image \( S \), the recipient also identifies \( m_s - m_r \) patches containing secret messages. For each patch, the recipient generates a category of candidate patches from \( \{p_1, p_2, \ldots, p_m\} \) using the error tolerance \( \delta \). Accordingly, the relationships between the secret bits and candidates can be generated. Consequently, the recipient can extract the hidden message from the synthesized patches.

4. EXPERIMENTAL RESULTS

We have conducted a series of experiments to verify the proposed method. Figure 6 shows an example of experimental results, in which (a) is the original texture pattern sized \( 128 \times 128 \), and (b) the extended pattern sized \( 228 \times 228 \) and the intermediate pattern is sized \( 191 \times 191 \), as shown in Figure 6(c). Figure 6(d) is a \( 528 \times 528 \) stego image containing 1519 secret bits, in

\[
\text{Figure 6: A group of experimental results: (a) is the source texture, (b) the extended texture, (c) the intermediate texture, and (d) the stego image}
\]

\[
\text{Figure 7: Another group of experimental results: (a) is the source texture, (b) the extended texture, (c) the intermediate texture, and (d) the stego image}
\]
Figure 8: The third group of experimental results: (a) is the source texture, (b) the extended texture, (c) the intermediate texture, and (d) the stego image.

which we use the parameter $\delta = 0.5$. Figure 7 shows another group of results using the same parameters as Figure 6. The stego image contains 1824 secret bits. Figure 8 shows the third group of results, in which (a) is the original pattern sized $128 \times 128$, (b) the extended pattern sized $258 \times 258$, and (c) the intermediate pattern containing $221 \times 221$ pixels. Figure 8(d) shows the stego image sized $608 \times 608$ containing 1679 bits, in which we use $\delta = 0.35$. Figure 9 shows the last group of results, in which (a) is the original pattern sized $128 \times 128$, (b) the extended pattern sized $288 \times 288$, and (c) the intermediate pattern containing $271 \times 271$ pixels. Figure 9(d) shows the stego image sized $608 \times 608$ containing 2698 bits, in which we use $\delta = 0.6$.

Figure 9: The last group of experimental results: (a) is the source texture, (b) the extended texture, (c) the intermediate texture, and (d) the stego image.

To evaluate the image quality of the synthetized image, we use Equation (3) to calculate the mean of MSEO values (MMSEO) in a synthesized image.

$$\text{MMSEO} = \frac{1}{(m_s - m_r) \times p_w \times p_h} \sum_{i=1}^{m_s - m_r} \text{MSEO}_i.$$  (3)

A smaller MMSEO value represents a better quality of texture synthesis. We also define the embedding rate ($\text{bpp, bit-per-patch}$) as

$$C_{\text{ave}} = \frac{C_e}{n_b},$$  (4)

where $C_e$ is the amount of hidden bits and $n_b$ the number of patches.

Figure 10: MMSEO values with respect to embedding capacity.
Figure 10 shows four groups of MMSEO results using different source patterns, corresponding to embedding rates ranging from 0 to 12 bpp. The sizes of the original texture pattern, the extended pattern, the intermediate pattern, and the stego image are 128 × 128, 228 × 228, 191 × 191, and 528 × 528, respectively. The results show that the image quality gets better when the embedding rate decreases. We also compare the proposed method with the method proposed by Wu et al. in [11]. The results show that the proposed method outperforms [11]. With the same embedding rate, the stego image generated by the proposed method has a smaller MMSEO value.

Figure 11 provides four groups of experimental results to compare the proposed method with [11]. The original pattern is sized 128 × 128, the extended pattern 228 × 228, the intermediate pattern 191 × 191, and the stego 528 × 528. The results show that the proposed method provides a better payload than [11] when using the same embedding rates.

The parameter δ also impacts the efficiency of data hiding. Table 1 shows the embedding capacity and quality corresponding to different δ. We use the fixed image sizes, i.e. the original pattern is 128 × 128, the intermediate pattern 191 × 191, and δ = 0.4. We also use 30 source patterns to implement the data hiding. The results show that the embedding capacity gets higher when the image size is larger.

The size of the intermediate pattern also affects the embedding capacity. We also use 30 images to calculate the average capacity. Figure 12 shows that embedding capacity is higher when the intermediate pattern is larger, when using a fixed parameter δ.

| δ   | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 1   |
|-----|-----|-----|-----|-----|-----|-----|
| C_{ave} | 4.0 | 7.6 | 10.8 | 12.7 | 13.6 | 14  |
| MMSEO | 501.5 | 599.9 | 717.0 | 815.3 | 900.7 | 947.9 |

| w_s × h_s | 488 | 568 | 648 | 728 | 808 | 888 |
|-----------|-----|-----|-----|-----|-----|-----|
| C_e       | 893.7 | 1274.4 | 1730.9 | 2222.9 | 2790.4 | 3403.7 |

Figure 12: Comparison of embedding capacity with respect to the threshold

which we use the original pattern is 128 × 128, the intermediate pattern is 191 × 191, and δ = 0.4. We also use 30 source patterns to implement the data hiding. The results show that the embedding capacity gets higher when the image size is larger.
While the method in [11] can be attacked by [12] according to the mirror symmetry, the proposed method can resist this attack. We use 4176 synthetic images from Brodatz Texture Database and the same settings as [12] to verify the security of information hiding. Table 3 shows the attacking results using [12], indicating that without the key the source patterns cannot be extracted from each stego using the proposed method. Therefore, an adversary cannot detect the information hiding behavior during texture synthesis.

5. CONCLUSION

This paper presents a constructive algorithm based on texture synthesis. We extend a source pattern to generate a reference pattern and an intermediate pattern. With these patterns, a stego image is constructed to carry secret messages. Experimental results show that the proposed method has three achievements, including a higher embedding capacity, the better quality of the synthesized image, and the capability of resisting attacks. In the future works, we will extend the constructive embedding into real images.

DISCLOSURE STATEMENT

No potential conflict of interest was reported by the authors.

FUNDING

This work was supported by the Natural Science Foundation of China [grant number U1536108, 61572308, U1636206, U1736213, 61602294, 61525203, and 61502009], the Shanghai Dawn Scholar Plan [grant number 14SG36] of Shanghai Municipal Education Commission, the Shanghai Excellent Academic Leader Plan [grant number 16XD1401200] of Shanghai Science and Technology Committee, and the Shanghai Sailing Program under [grant number 16YF1404100] of Shanghai Science and Technology Committee.

REFERENCES

1. N. Provos and P. Honeyman, “Hide and seek: An introduction to steganography,” IEEE Secur. Priv., Vol. 1, no. 3, pp. 32–44, 2003.

2. Z. Qian and X. Zhang, “Reversible data hiding in encrypted images with distributed source encoding,” IEEE Trans. Circ. Syst. Vid. Tech., Vol. 26, no. 4, pp. 636–46, 2016.

3. N. Johnson and S. Jajodia, “Exploring steganography: Seeing the unseen,” IEEE Comput., Vol. 31, no. 2, pp. 26–34, 1998.

4. A. Westfeld, “F5 – A steganographic algorithm,” in International Workshop on Information Hiding, I. S. Moskowitz, Ed. Berlin, Heidelberg: Springer, 2001, pp. 289–302.

5. W. Zhang, X. Zhang, and S. Wang, “Near-optimal codes for information embedding in gray-scale signals,” IEEE Trans. Inform. Theory, Vol. 56, no. 3, pp. 1262–70, 2010.

6. J. Fridrich, T. Filler, and J. Judas, “Minimizing additive distortion in steganography using syndrome-trellis codes,” IEEE Trans. Inform. Foren. Sec., Vol. 6, no. 1, pp. 920–35, 2011.

7. H. Otori and S. Kuriyama, “Data-embeddable texture synthesis,” in Proceedings of the 8th International Symposium on Smart Graphics, Kyoto, Japan, 2007, pp. 146–57.

8. H. Otori and S. Kuriyama, “Texture synthesis for mobile data communications,” IEEE Comput. Graph. Appl., Vol. 29, no. 6, pp. 74–81, 2009.

9. J. Xu, X. Mao, and X. Jin, “Hidden message in a deformation-based texture,” Visual Comput., Vol. 31, no. 12, pp. 1653–69, 2015.

10. X. Jin, S. Chen, and X. Mao, “Computer-generated marbling textures: A GPU-based design system,” IEEE Comput. Graph. Appl., Vol. 27, no. 2, pp. 78–84, 2007.

11. K. C. Wu and C. M. Wang, “Steganography using reversible texture synthesis,” IEEE Trans. Image Process., Vol. 24, no. 1, pp. 130–9, Jan. 2015.

12. H. Zhou, K. Chen, W. Zhang, and N. Yu, “Comments on ‘Steganography using reversible texture synthesis’,” IEEE Trans. Image Process., Vol. 26, no. 4, pp. 1623–25, Apr. 2017.

13. Z. Qian, H. Zhou, W. Zhang, and X. Zhang, “Robust steganography using texture synthesis,” in Twelfth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIIH-MSP 2016, Kaohsiung, Taiwan, Nov., 2016, pp. 25–33.

14. L. Y. Wei and M. Levoy, “Fast texture synthesis using tree-structured vector quantization,” in Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000, New Orleans, Louisiana, USA, Jul. 2000, pp. 479–88.

15. A. A. Efros and T. K. Leung, “Texture synthesis by non-parametric sampling,” in Proceedings of 7th IEEE International Conference on Computer Vision, Kerkyra, Greece, Sep. 1999, pp. 1033–38.

16. A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, Aug. 2001, pp. 341–46.
Authors

Zhenxing Qian received both the BS and the PhD degrees from University of Science and Technology of China (USTC) in 2003 and 2007, respectively. Since 2009, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. He has published over 80 peer-reviewed papers on international journals and conferences. His research interests include information hiding, image processing, and multimedia security.

Email: zxqian@shu.edu.cn

Nannan Huang received the BS degree in School of Communication and Information Engineering, Shanghai University. She is currently working towards the Master’s degree at Shanghai University. Her research interests include information hiding and image processing.

Email: 1163684253@qq.com

Sheng Li received the PhD degree at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2013. From 2013 to 2016, he was a research fellow in Rapid Rich Object Search (ROSE) Lab, Nanyang Technological University. He is currently with the faculty of the School of Communication and Information Engineering, Shanghai University, China. His research interests include biometric template protection, pattern recognition, multimedia forensics, and security. He is the recipient of the IEEE WIFS Best Student Paper Silver Award.

Email: shengli@shu.edu.cn

Xinpeng Zhang received the BS degree in computational mathematics from Jilin University, China, in 1995, and the ME and PhD degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. His research interests include information hiding, image processing, and digital forensics.

Corresponding author: Email: xzhang@shu.edu.cn