Constructive Texture Steganography Based on Compression Mapping of Secret Messages

Fengyong Li[1, *], Zongliang Yu[1] and Chuan Qin[2]

Abstract: This paper proposes a new constructive texture synthesis steganographic scheme by compressing original secret messages. First, we divide the original message into multiple bit blocks, which are transferred to decimal values and compressed into small decimal values by recording their interval sign characters. Then, a candidate pattern is generated by combining the given source pattern and boundary extension algorithm. Furthermore, we segment the candidate pattern into multiple candidate patches and use affine transformation algorithm to locate secret positions on a blank canvas, which are used to hide the sign characters by mapping the candidate patches. Finally, we select the candidate patches with minimal mean square error to represent secret bits to generate stego image by image quilting. Extensive experiments demonstrate that compared with existing texture steganographic methods, our method has a better visual quality, higher embedding capacity and security performance, while maintaining strong anti-steganalysis capability.

Keywords: Constructive steganography, texture synthesis, compression mapping, patch stitching.

1 Introduction

Steganography [Fridrich (2009), Filler, Judas and Fridrich (2011), Li, Wu, Zhang et al. (2018)] aims to embed secret messages into digital media, such as digital images, video or audio files, for covert communication. In order to minimize the changes caused by secret information embedding, researchers employ coding methods to achieve fewer changes with embedding the same amount of secret information, such as wet paper coding [Fridrich, Goljan, Lisonek et al. (2005)] and double-layer steganography method [Zhang, Zhang and Wang (2007)]. Code-based Steganography [Westfeld (2001)] can theoretically be closer to the performance limitation of steganographic distortion, but, in

[1] College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China.
[2] School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, China.
* Corresponding Author: Fengyong Li. Email: fyli@shiep.edu.cn.
Received: 16 December 2019; Accepted: 17 March 2020.

practical applications, it is not satisfactory when resisting specific steganalysis with high-dimensional feature model [Li, Wu, Lei et al. (2015)]. So, some researchers try to develop secure steganography [Filler, Judas and Fridrich (2011)] with high anti-steganalysis capability, for example, coverless steganography [Zhang, Su, Li et al. (2019)] and deep learning-based secure steganography [Yang, Ruan, Huang et al. (2019), Meng, Cui and Yuan (2018)]. Nevertheless, since steganography mainly hides data into image elements (pixels or DCT (Discrete Cosine Transform) coefficients), the distribution of image modes usually change after performing steganography, even if for a low embedding capacity. This offers the opportunity for adversary to attack the methods.

Considering the aforementioned discussion, some researchers employ the textural similarity of texture image to directly generate stego image [Qian, Pan, Li et al. (2018), Zhou, Qiu, Li et al. (2018)]. The secret messages are mapped to some blocks of stego image. We term this scheme as "constructive steganography". In fact, constructive steganography hides secret message during reconstructing stego image. It directly generate stego image instead of modifying the cover image, and thus has a higher security performance.

Constructive steganography is usually documented two categories: texture constructive scheme and texture synthesis scheme. Regarding the texture constructive scheme, Qian et al. [Qian, Pan, Li et al. (2018)] hide the secret information by generating water-like textures, it can be visually perceived due to containing much unnatural texture. Regarding the texture synthesis scheme, Otori et al. [Otori and Kuriyama (2009)] firstly proposes a texture stitching scheme, which masks the local binary pixels that express secret information by using texture stitching method. However, this scheme may cause attacker's attention due to inferior masking mechanism. In order to avoid this problem, Wu et al. [Wu and Wang (2014)] hide secret information by using patch stitching. In this scheme, the source pattern is divided into overlapping patches and the secret message are embedded during reorganizing the approximate patches. Although Wu's method provides a high embedding payload, it has been verified by Zhou's comment [Zhou, Chen, Zhang et al. (2017)] to have unsafe performance and low visual quality for stego images. In addition, Du et al. [Du, Wang, Zhao et al. (2019)] employ quasi-affine transform on finite integer mesh to resist Zhou's attack method [Zhou, Chen, Zhang et al. (2017)] and further improve steganographic security. Qian et al. [Qian, Huang, Li et al. (2018)] design new boundary generation method to resist a specified attack aiming at patch stitching-based steganography. Unfortunately, these schemes still do not show enough superiority, especially for the visual quality, key security, and anti-steganalysis capability, keeping a large improvement gap.

Facing the aforementioned problems, we make the following novel contributions in the context of constructive texture steganography:

- We propose a new solution for constructive steganography, which can not only hide high payload without modifying the original texture image but also ensure that the stego images have a significant superior security performance.

- The proposed new solution achieves high embedding capacity by compression mapping for original secret messages. The stego images can be synthesized step by step by searching the candidate patches with the minor mean square error. Since the proposed scheme effectively narrows the visual gap among synthesized patches, the visual quality can be thus further improved.
- Comprehensive experiments are performed over different textural images and the experimental results demonstrate that our solution significantly outperforms existing texture steganography methods in terms of visual quality, embedding capacity, security performance and anti-steganalysis capability.

The rest of this paper is organized as follows. Section 2 introduces the related work. The detailed procedure of our proposed scheme is shown in Section 3. We perform comprehensive experiments to evaluate the performance of proposed scheme, and present the results and corresponding discussions in Section 4. Finally, Section 5 concludes the paper.

2 Related work

Texture synthesis is a technology that uses a source texture pattern to generate a larger sized pattern with similar appearance. Wu et al. [Wu and Wang (2014)] firstly designed a new method to hide secret message using patch stitching. In Wu's scheme, they defined the basic unit as a "patch", which contains a kernel region and a boundary region. The source pattern is divided into multiple non-overlapping kernels with $K_w \times K_h$ so that it can be recovered completely once these non-overlapping kernels are obtained. Then, they expanded each kernel into patch. If a kernel is located at the boundary of source pattern, the mirror symmetry method is used to implemented this processing. Finally, the extended source pattern were segmented into multiple candidate patches using the raster scanning. With a secret key, the data hider can randomly distribute all overlapping patches into a blank canvas. With image quilt technology [Efros and Freeman (2001)], the remaining patches in blank canvas were further filled by the patches that have the same ranks of mean square error (MSE) with the secret message. When the blank canvas are filled completely, it can be finally considered as stego image.

Data extraction is a reverse procedure of data embedding. With the secret key, the receiver first extracts all overlapping patches from the stego image. Then they extract each kernel from each patch to recover the source pattern. Subsequently, the overlapping patches and candidate patches can be generated from the source pattern again. By calculating the MSE values and ranking them, the recipient can find easily the ranks of the same patch between stego patch and candidate patch so that the secret bits can be extracted exactly.

Since Wu's scheme used the mirroring technique to extend the boundary patch of the source pattern, Zhou et al. [Zhou, Chen, Zhang et al. (2017)] searched patches in the stego images to recover the source pattern by comparing the similarity of the mirroring part. Using the same way of Wu's method, they can successfully extract the hidden messages.

Furthermore, in order to prevent the attack of Zhou's scheme, Du et al. used Quasi-Affine transformation on limited integer to generate redundant location, which are pasted by randomly selecting some candidate patches. Accordingly, even if the attacker successfully recover the source pattern, they cannot extract the secret message correctly because they do not know exactly which patches hide the secret messages.

In order to improve the security and visual quality, Qian et al. [Qian, Huang, Li et al. (2018)] designed a new method to generate boundary region. In addition, they extend the source pattern to generate more candidate patches, which are used to synthesis the steganography pattern, resulting in an improvement of visual quality of stego image.

Although the above mentioned schemes can achieve the covert communication of secret information by constructing the texture stego image, there still has an obvious improvement room, e.g., visual quality, key security, and anti-steganalysis capability. This work is to fill this gap.

3 Steganography based on compression mapping of secret messages

3.1 The framework of proposed scheme

The framework of proposed scheme contains two parts: message preprocessing and texture image extension. In the message preprocessing stage, we divide the original message into multiple bit blocks. The decimal value transferred by each bit block is further compressed by mapping them to different interval, which are denoted by different block sign characters. In texture image extension stage, given a small source pattern, we firstly construct the candidate pattern by combining source pattern and block matching algorithm. Then, source pattern is extended and is randomly hashes on some locations in a blank canvas (original stego pattern). Subsequently, we divide the candidate pattern into multiple overlapping patches and distribute them into the remaining locations of stego pattern by a key. Finally, the compressed messages are referred to select the candidate patches that equals to the same decimal number of mean square error, which are filled in the remaining blocks in stego pattern until the stego pattern is synthesized completely. Since the sign characters mapping the compressed messages are fully recorded, the original messages can be perfectly restored. The overall framework is shown in Fig. 1.

3.2 Compression mapping of secret messages

In this section, we first process the original messages. Assume that M is the bit length of original messages, we segment the original messages into N bit blocks.

$$N=\left\lceil \frac{M}{BP} \right\rceil \tag{1}$$

where BP is the number of bits embedded in each patch and can be transferred to a decimal interval [0, BP_d], where BP_d is the maximum decimal value that BP bits can represent. Notably, BP is not related to the patch size, but to the total amount of secret messages.

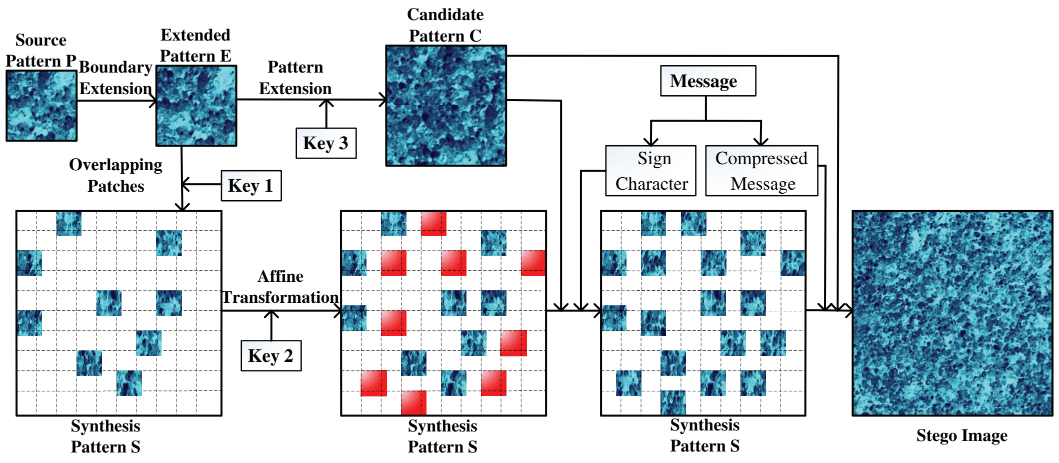

Figure 1: Overall framework of proposed method

With the same number of patches, the larger amount of secret messages, the more bits each patch needs to carry. Accordingly, if more secret messages need to be delivered, we can set a bigger *BP* for each patch.

We further divide the interval $[0, BP_d]$ evenly into four sub-intervals $[0, \alpha]$, $[\alpha+1, \beta]$, $[\beta+1, \gamma]$ and $[\gamma+1, BP_d]$. For any secret bit block, we can transfer it to a decimal value and map it into one of the above four sub-intervals.

$$T' = \begin{cases} T-\gamma & , \quad BP_d \geq T > \gamma \\ T-\beta & , \quad \gamma \geq T > \beta \\ T-\alpha & , \quad \beta \geq T > \alpha \\ T & , \quad \alpha \geq T \geq 0 \end{cases} \tag{2}$$

where T' represents the compressed secret messages. Since the compressed messages must be recovered correctly when they are extracted, the receiver needs to know which sub-interval secret bit block is mapped. Thus, we use two bits as "sign character" to record the sub-interval that one bit block is mapped.

$$l = \begin{cases} 11 & , \quad BP_d \geq T > \gamma \\ 10 & , \quad \gamma \geq T > \beta \\ 01 & , \quad \beta \geq T > \alpha \\ 00 & , \quad \alpha \geq T \geq 0 \end{cases} \tag{3}$$

where l stands for the actual sign character for each bit block.

We can use a simple example to explain this procedure. Assume that the length of each bit block BP=12, it thus can indicate a decimal interval [0, 4095]. We divide the interval [0, 4095] as four sub-intervals [0, 1023], [1024, 2047], [2048, 3071] and [3072, 4095].

Given a secret bit block “111011001011”, its decimal value can be represented as “3787”. We can use Eq. (2) to compress the decimal value “3787” to “716”, and subsequently map it to “11” as sign character by Eq. (3).

Notably, since the sign character can uniquely pin to the sub-interval that one bit block is mapped, it can thus guarantee that the secret messages are recovered accurately.

3.3 Texture extension

In this section, we show the procedure of candidate pattern construction. Given the source pattern P with size $S_w \times S_h$, we segment the source texture image into a number of non-overlapped kernel blocks, which have the same size $K_w \times K_h$, e.g., Fig. 2. Let KB represents the set containing all kernel blocks, each source patch can be indexed by their subscript kb_i. For example, given a source texture image with the size of $S_w \times S_h$=128×128, we can generate KB=16 kernel blocks if $K_w \times K_h$=32×32. Accordingly, all kernel blocks can be indexed sequentially by as kb_0, kb_1, …, kb_{15}. Then, we extend the boundary region on pattern P to construct pattern E with size $(S_w+2P_d)\times(S_h+2P_d)$, where P_d (P_d>0) is the width of extended boundary.

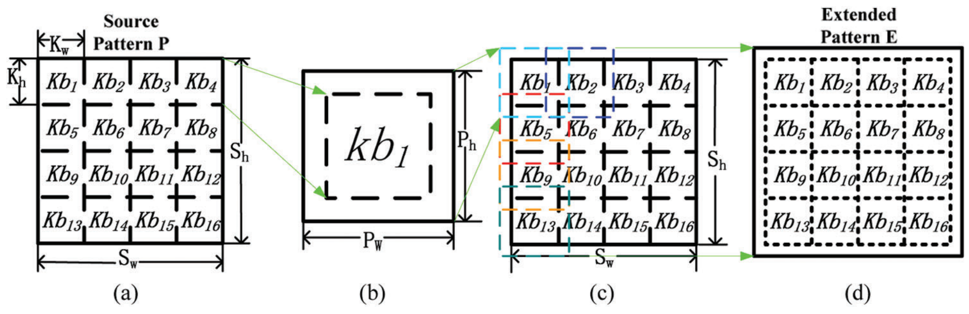

Figure 2: The construction procedure of extended pattern. (a) Source pattern, (b) Kernel, (c) Extended kernel, (d) Extended pattern

We use the window with size $P_w \times P_h$ to scan source pattern P (referring to Fig. 3). Multiple candidate patches are obtained and their total number can be calculated as follows.

$$CP_{\mathrm{n}}=(S_w-P_w+1)\times(S_h-P_h+1) \tag{4}$$

We further put source pattern P on the center position of a blank canvas to construct pattern E, whose size is $(S_w+2P_d)\times(S_h+2P_d)$. Obviously, there is a boundary sized P_d between pattern P and pattern E. In order to fill the blank between pattern P and blank canvas, we choose the most suitable candidate patch to synthesis pattern E. To be specific, we firstly calculates the mean square error (*MSE* for short) of the overlapped region between synthesized patch and each candidate patch. Since the total number of candidate patches are CP_n, there are thus CP_n *MSE* values.

$$MSE = \sum_{j \in OL_i} \left(p_j^c - p_j^s\right)^2 \tag{5}$$

where OL_i stands for pixels of overlapped region p_j^c indicates the pixel of candidate patch, and p_j^s is the pixel of synthesis region.

Then, we sort the CP_n *MSE* values in ascending order and select the candidate patch with the first rank as synthesized patch. This procedure can guarantee that the *MSE* between the selected candidate patch and the region that need to be filled is minimum. That is also to say, the most similar candidate blocks can ensure the synthesis steganographic image has the highest visual quality. Finally, the selected candidate patches are pasted on the blank region one by one to generate pattern *E*, which is used to synthesis stego pattern *S* and candidate pattern *C*. The complete procedure can be found in Fig. 4.

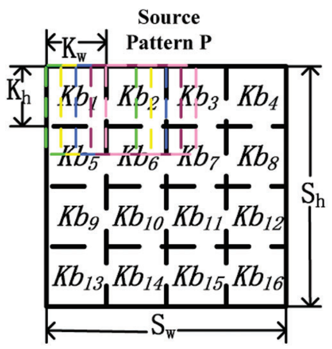

Figure 3: The procedure of raster scanning. A window with $P_w \times P_h$ is used to perform the scanning by pixel-by-pixel from left to right and up to down

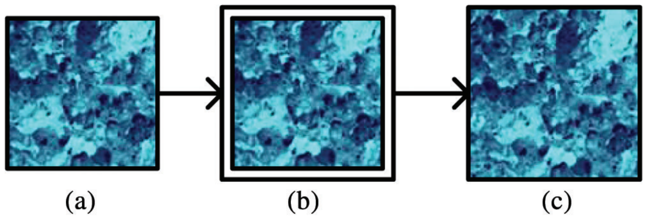

Figure 4: Source pattern *P* use boundary extension to generate extended pattern *E*. (a) Source Pattern P, (b) Boundary Extension, (c) Extended Pattern E

3.4 Affine transformation and sign character

With extended pattern *E*, we further divide it into SP_n overlapping patches by using the same raster scanning in Fig. 5. Each overlapping patch is $P_w \times P_h$.

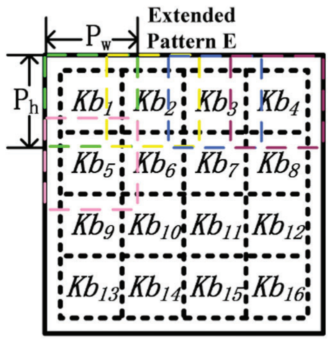

Figure 5: With the extended pattern E, we could get overlapping patches which have the same number compare to kernels divided by source pattern *P*

$$SP_n = \frac{(E_w - 2P_d) \times (E_h - 2P_d)}{K_w \times K_h} \tag{6}$$

where $E_w=(S_w+2P_d)$ and $E_h=(S_h+2P_d)$ are the width and height of pattern E, respectively. Thus,

$$SP_n = \frac{S_w \times S_h}{K_w \times K_h} \tag{7}$$

Given another blank canvas S with size $S'_w \times S'_h$, we use the same method as pattern E to split S into blocks with the same size.

$$SP'_n = \left\lfloor \frac{S'_w - P_w}{P_w - P_d} + 1 \right\rfloor \times \left\lfloor \frac{S'_h - P_h}{P_h - P_d} + 1 \right\rfloor \tag{8}$$

Subsequently, SP_n overlapping patches are randomly distributed into the blank canvas S by a secret key 1. Since S has a bigger size than pattern E, the block number of remaining blank region in S is $SP'_n – SP_n$. Furthermore, the affine translation with another secret key 2 (a, b, c, d, e, f) is used to randomly map some locations by Eq. (9) (Corresponding to the red patches in Fig. 6). These selected locations are used to embed the sign characters from secret messages, which have been explained in Section 3.2.

$$\begin{aligned}\begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \\ &= \begin{pmatrix} 1 & \dfrac{(a-1)}{c} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ c & 1 \end{pmatrix}\begin{pmatrix} 1 & \dfrac{(1-a)d}{c} \\ 0 & ad-bc \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}\end{aligned} \tag{9}$$

Denote the number of selected patches as TP_n, the final number of remaining blank patches can be then calculated as follows.

$$EP_n = SP'_n - SP_n - TP_n \tag{10}$$

3.5 Data embedding and data extracting

In the above three subsections, we have generated extended pattern E and pattern S, where pattern S contains some candidate patches that are used to hide the sign characters of secret messages (Corresponding to the red patches in Fig. 6). In this section, we further explain the procedure of data embedding and extraction.

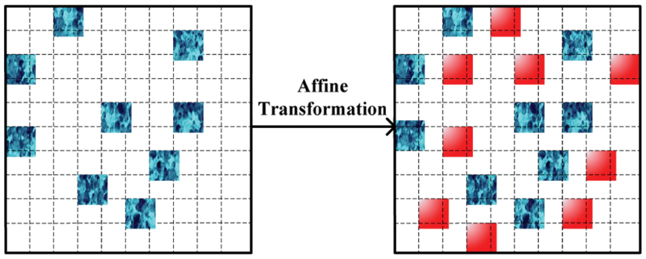

Figure 6: The locations selected by affine transformation. Red blocks represent the secret locations which are used to hide the sign character, while blue blocks are the candidate patches segmented by extended pattern E

3.5.1 Data embedding

We first process secret messages as two parts: compressed messages and their sign characters. According to Section 3.2, the number of compressed messages is N, the sign characters are thus $2N$. We describe the embedding procedure as follows.

Step 1: We extend the pattern E to another bigger candidate pattern with size $C_w \times C_h$, which is denoted as C. The actual processing is as follows. Divide E into multiple overlapping patches (referring to Fig. 3) and randomly distribute them into pattern C by a key 3. The remaining blocks can be filled with the candidate blocks, which are generated from pattern P by raster scanning (referring to Fig. 3). The size of pattern C is naturally bigger than pattern E.

Step 2: The pattern C is further divided into multiple candidate patches by raster scanning (referring to Fig. 3), which are used to synthesis pattern S by combining compressed messages and their sign characters. Denote the number of candidate patches as CP'_n,

$$CP'_n = (C_w - P_w + 1) \times (C_h - P_h + 1) \tag{11}$$

Then, denote that TP_n is the number of candidate patches that are used to hide the sign characters in S.

$$TP_n = \left\lceil \frac{2N}{L} \right\rceil = \left\lceil \frac{2N}{\lfloor \log_2 CP'_n \rfloor} \right\rceil \tag{12}$$

where $L=\lfloor \log_2 CP'_n \rfloor$ is the length of sign characters that each candidate patch can embed the maximum bit number.

Step 3: Following the raster scanning order, sort the CP'_n candidate patches from pattern C and denote their serial number as $c_1, c_2, \ldots, c_{CP'_n}$. Then, transfer L bits sign characters to a decimal value T_L. If $T_L=c_i$, $1 \le i \le CP'_n$, the corresponding c_i-th candidate patch in pattern C is selected to fill the red synthesized patch in S.

Step 4: According to the Eq. (10), the remaining EP_n synthesized patches in S can be use to embed the compressed messages, which contain N decimal values $\{T'_1, T'_2, \ldots, T'_N\}$ that are calculated by Eq. (2). For any synthesized patch, we replace it by using each candidate patches from pattern C, respectively, and then obtain CP'_n MSE values. Sort CP'_n MSE values with ascending order and denote their serial number as $\{s_1, s_2, \ldots, s_{CP'_n}\}$. If $T'_k=s_i$, $1 \le k \le N$ and $1 \le i \le CP'_n$, the corresponding s_i-th candidate patch in pattern C is selected as synthesized patch, which is used to carry the compressed message T'_k.

Step 5: Repeat the Step 4 until the pattern S is completely synthesized as stego image.

Notably, in these above steps, if $N<EP_n$, we fill the remaining EP_n–N synthesized patches by using the s_1-th candidate patch in pattern C each time, otherwise, we can solve this problem easily by extending the size of pattern S.

3.5.2 Data extracting

When the stego image S is obtained, the complete messages can be easily extracted by the following steps.

Step 1: According to the key 1, the positions of overlapping patches from source P in stego image S can be located exactly. Then, the source P can be recovered completely. With the source P, pattern C is also extended based on the procedure in Section 3.3.

Step 2: Furthermore, we use the key 2 and affine transformation to locate the synthesized patches that embed the sign characters of compressed messages. The sign characters can be extracted by combining the pattern C.

Step 3: The compressed messages $\{T'_1, T'_2, \ldots, T'_N\}$ are also sequentially extracted by calculating and sorting the MSE values. Finally, the original messages can be recovered correctly by combining the compressed messages and their sign characters.

4 Experimental results and discussion

4.1 Experimental setup

We test the proposed scheme by a series of experiments that are carried out with Brodatz Textures. Four kinds of texture source pattern with different texture distribution are shown in Fig. 7. In our experiments, we set the size of source pattern P, pattern C and the stego pattern S as 128×128, 191×191, and 528×528, respectively. The corresponding

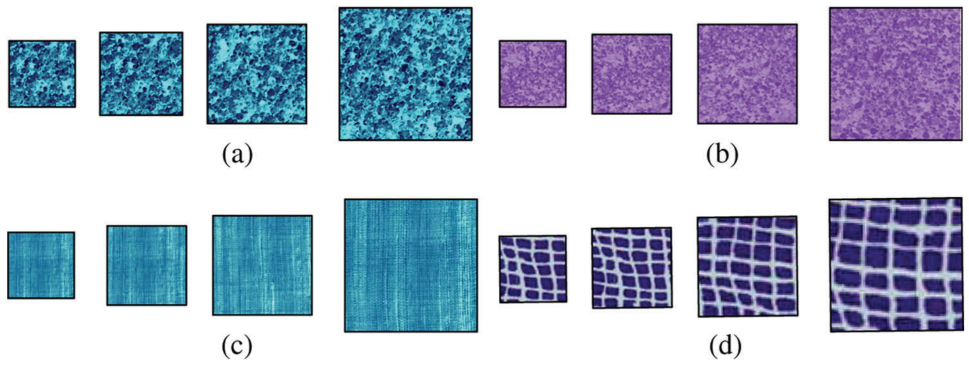

Figure 7: Four kinds of texture source patterns are used in our experiments. For different source pattern, the first picture is the source texture *P*, the second is the extended pattern *E*, the third is candidate pattern *C*, and the last pattern is synthesized pattern *S*

experiments are implemented over a computer with an Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz 1.80 GHz and 8 GB RAM.

4.2 Visual quality test for different texture image

In this experiment, we test the visual quality of synthesized stego images using our proposed scheme. For illustration purpose, we evaluate the visual quality of stego image by calculating the average MSE values (*AMSE* for short).

$$AMSE=\frac{1}{EP_n \times P_w \times P_h}\sum_{i=1}^{EP_n}\sum_{j\in OL_i}(p_j^c - p_j^s)^2 \tag{13}$$

where OL_i represents the pixels of overlapping regions. Generally speaking, the smaller *AMSE* value indicates the better visual quality of synthesized stego image. In addition, we consistently use the bits of per patch (*BP* for short) to measure the embedding capacity for different steganographic schemes, which can be calculated by Eq. (1).

We compare proposed scheme with Wu's scheme [Wu and Wang (2014)] and Qian's scheme [Qian, Huang, Li et al. (2018)] by simulating four different textural sources. Accordingly, the synthesized stego images generated by above four kinds of texture sources have different visual quality. To give an overall insight, the experiments are repeated 10 times, and the average results are shown in Fig. 8. As can be seen from this figure, whatever textural sources are used, our scheme always have an obvious lower *AMSE* values with the same embedding capacity. This demonstrates that proposed scheme can obtain significant better visual quality than that of other two schemes. This phenomenon can be explained easily. Proposed scheme employs compression mapping processing to map messages to a smaller decimal sub-interval (referring to Eq. (2)). This makes that the candidate patches with the minimal *MSE* value in pattern *C* are easier to be selected as

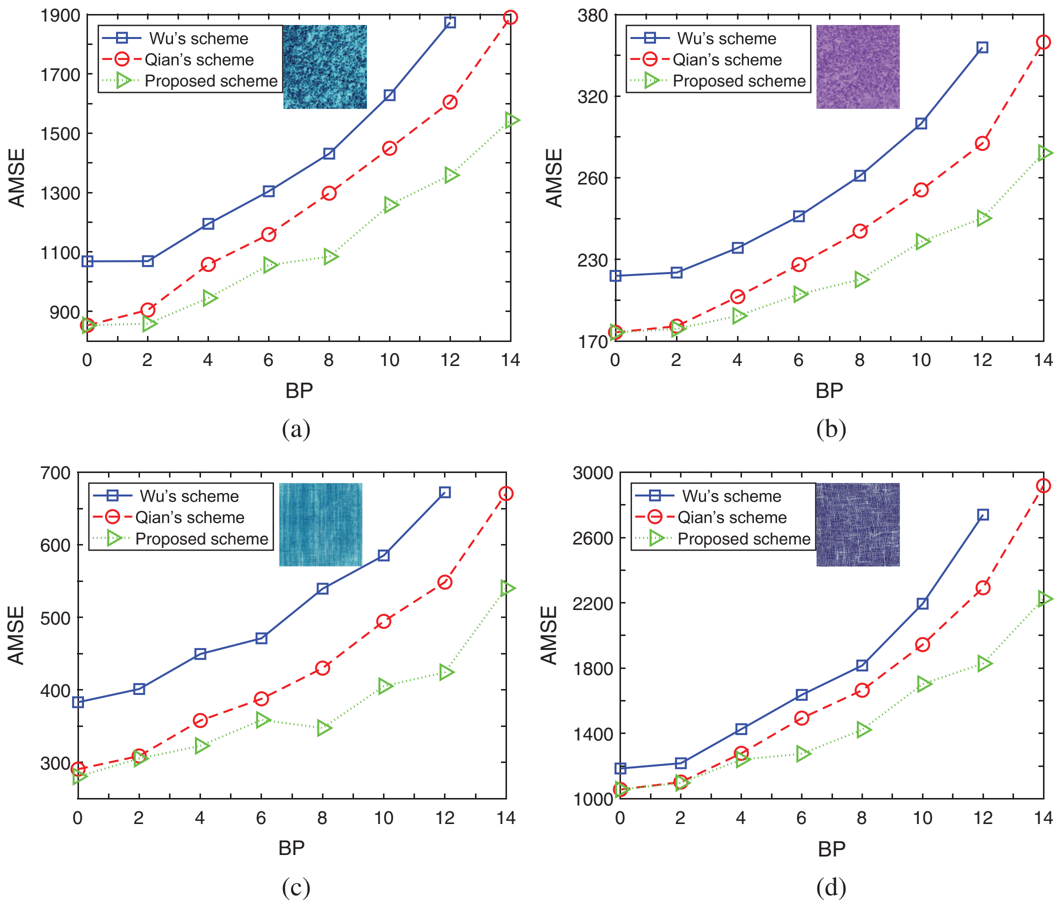

Figure 8: Relationship between embedding capacity and visual quality for different texture image. The abscissa stands for the number of bits in each patch, while the ordinate represents the average *MSE*

Table 1: AMSE performance over different textural images when *BP*=12

AMSE	**Fig. 7a**	**Fig. 7b**	**Fig. 7c**	**Fig. 7d**
Wu's scheme	1873.5	385.9	672.4	2740.4
Qian's scheme	1605.0	315.3	548.5	2292.4
Proposed scheme	1358.9	26.3	424.4	1828.4

the synthesized patch of *S*, resulting in a better visual quality for stego image. The actual experimental results with *BP*=12 can be found intuitively in Tab. 1.

In addition, we can observe an interesting phenomenon, that is, when *BP* is obvious small, proposed scheme presents an insignificant advantage comparing with other two schemes.

This is mainly because less secret messages implies that the pattern C is divided into less candidate patches, which results in the effect of compression mapping cannot be fully reflected, because the top ranked candidate patches are always selected when mapping compressed messages.

4.3 Comparison of embedding capacity

In this section, we compare the embedding capacity of our proposed scheme with other two steganographic schemes.

We first test the maximum embedding capacity BP_{max} when setting textural sources with different size, $T_w \times T_h$=1024×1024, and 1008×1008. The corresponding results are shown in Tabs. 2 and 3. From these two tables, we can observe that although proposed scheme improves the maximum embedding capacity comparing with Wu's scheme, i.e., the gain is approximately 2%, it has a clear gap between proposed scheme and Qian's scheme, approximately 14%. This is mainly because proposed scheme employs some synthesized patches in pattern S to hide the sign characters. This makes that the synthesized patches in S that can be used to carry the secret messages are relatively less than that of Qian's scheme. Nevertheless, we would like to stress that proposed scheme still has a superior performance under the same embedding capacity. Actually, this conclusion can be also verified by Fig. 8. In addition, Fig. 9 shows the embedding capacity under the same *AMSE* for different texture images. We can see that with the same *AMSE*, our proposed scheme has a higher embedding capacity than that of other schemes. Moreover, we also observe that Fig. 9d has a significant bigger *AMSE* value. This demonstrates that the more complex the image texture, the worse the visual quality of synthesized stego image.

Table 2: The maximum embedding capacity (bits) comparison for three steganographic schemes, proposed scheme, Wu's scheme and Qian's scheme

$S_w \times S_h$	Scheme	SP_n	SP'_n	TP_n	EP_n	BP_{max}
128×128	Wu's scheme	2601	64	0	2537	30444
	Qian's scheme	2601	64	0	2537	35518
	Proposed scheme	2601	64	317	2220	31080

In this experiment, the parameters are set to $T_w \times T_h$=1024×1024, $P_w \times P_h$=24×24 and P_d=4.

4.4 Security analysis for key

In our proposed scheme, three keys, Key 1, Key 2, Key 3, are used to improve the algorithm security, where Key 1 is used to randomly distribute the overlapping patches from extended pattern E into stego pattern S, Key 2 is used to implement affine transformation, while the Key 3 is used to control the extension procedure from pattern E to candidate pattern C. For Wu's scheme, since it only uses the Key 1 to randomly distribute the overlapping patches, an

Table 3: The maximum embedding capacity (bits) comparison for three steganographic schemes, proposed scheme, Wu's scheme and Qian's scheme

$S_w \times S_h$	Scheme	SP_n	SP'_n	TP_n	EP_n	BP_{max}
128×128	Wu's scheme	625	16	0	609	7308
	Qian's scheme	625	16	0	609	8526
	Proposed scheme	625	16	76	533	7462

In this experiment, the parameters are set to $T_w \times T_h$=1008×1008, $P_w \times P_h$=48×48 and P_d=8.

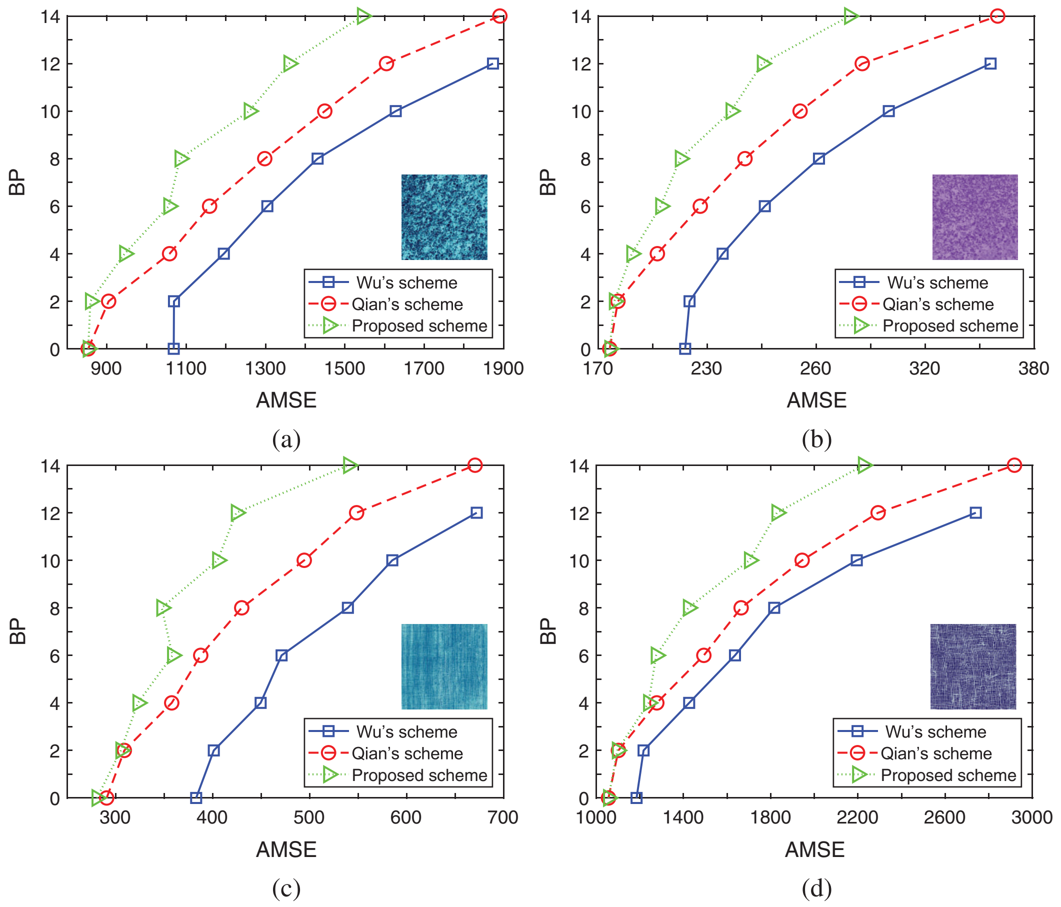

Figure 9: Relationship between visual quality and embedding capacity for different texture image. The ordinate stands for the number of bits in each patch, while the abscissa represents the average *MSE*

attacker can obtain easily source pattern from stego pattern and then recover the secret message once he gets this key. This conclusion has been validated by Zhou et al. [Zhou, Chen, Zhang et al. (2017)].

Additionally, comparing with Qian's scheme, proposed scheme employs another key to implement affine transformation, which can select some positions in synthesized stego pattern S to hide the sign characters produced by secret messages. Although these positions consume some embedding capacity comparing with Qian's scheme, it improves exactly security performance of algorithm, because even if an attacker obtains the Keys 1 and 3, he cannot distinguish which blocks are embedded with secret messages. Tab. 4 shows the qualitative analysis of key security for different steganographic schemes.

Table 4: Qualitative analysis of key security for different steganographic schemes

Attacking Cases	**Wu's scheme**	**Qian's scheme**	**Proposed scheme**
Key 1 lost	No	Yes	Yes
Key 1 and Key 2 lost	No	No	Yes

4.5 Test for anti-steganalysis performance

In this section, to evaluate anti-steganalysis capability, we use the classical quantitative steganalysis method, RS analysis, to give testing results. For RS analysis, the relative number of regular groups for mask M=[0 1 0 1] and $-M$=[0 −1 0 −1] are denoted as R_M and R_{-M}, respectively, while the relative number of singular groups for masks M and $-M$ are denoted as S_M and S_{-M}, respectively. If the suspicious image is not performed by steganography, $R_M \approx R_{-M}$ and $S_M \approx S_{-M}$. Otherwise, $R_M > S_M$ and $R_{-M} > S_{-M}$.

We carry a series of experiments to test the four parameters, R_M, R_{-M}, S_M and S_{-M}, with different embedding capacity BP. The corresponding results have been shown in Tab. 5. From this table, we can see that no matter what the embedded capacity is, the results always keep $R_M \approx R_{-M}$ and $S_M \approx S_{-M}$. Actually, our proposed scheme can guarantee high security because it embeds the secret messages by synthesizing different candidate patches, not modifying the pixel value. In addition, we can explain the phenomenon that the testing results have a slight fluctuations. This is because the image stitching technique is used to synthesize the stego image S in our scheme, although the block-based

Table 5: RS analysis for different embedding capacity

BP	R_M	S_M	R_{-M}	S_{-M}
Original	1609	541	1572	566
4BP	1601	543	1588	552
8BP	1593	547	1574	569
12BP	1603	533	1583	557

synthesis algorithm [Efros and Freeman (2001)] has been used to eliminate the pixel mismatch as much as possible, the slight mismatching still exist during the synthesis process, resulting in slight modifications for stego images.

4.5.1 Test for time complexity

To give a more insight, we test the time complexity of proposed scheme over different textural images. The corresponding experimental results have been shown in Tab. 6. From this table, we can observe that Wu's scheme has the lowest time complexity. This is because Wu's scheme directly selects the candidate patch that equals to the decimal value of *BP* secret bits, leading to low time complexity, but, an inferior visual quality. In contrast, our scheme and Qian's scheme construct a bigger candidate pattern to generate more candidate patches. More candidate patches imply that when the steganographic image is constructed, the optimal synthesis patch has a larger selection range. Accordingly, the generated stego image has a higher visual quality, but, higher time complexity.

Table 6: Time complexity comparison for different textural images

Time	**Fig. 7a**	**Fig. 7b**	**Fig. 7c**	**Fig. 7d**
Wu's scheme	17.94 s	21.19 s	20.94 s	23.53 s
Qian's scheme	72.63 s	68.02 s	62.67 s	89.34 s
Proposed scheme	92.93 s	84.42 s	82.76 s	94.01 s

5 Conclusions and future work

Constructive texture steganography has a much needed in the presence of data hiding, because the stego image can be constructed by synthesizing the image block directly, not modifying the pixels. This important requirement, however, is largely ignored in existing traditional steganography. We further improved the application by developing a new method, which designs the compression mapping method over secret messages. Our scheme can extract original secret messages accurately while keeping higher visual quality for stego images. The proposed scheme, verified with comprehensive evaluation, has high embedding capacity, security performance and good anti-steganalysis capability.

In the future, we plan to extend the work in two directions. First, we will investigate good mapping methods to further compress the original secret messages while keeping high visual quality. Second, we plan to extend our work to compressed image formats.

Funding Statement: This work was supported by Natural Science Foundation of China under Grants (U1736120, 61602295).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Du, Y.; Wang, Z.; Zhao, G.; Zhang, Y. (2019): A secure coverless texture synthesis information hiding scheme. *Computer Applications and Software*, vol. 6, pp. 53.

Efros, A. A.; Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 341-346.

Filler, T.; Judas, J.; Fridrich, J. (2011): Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 920-935. DOI 10.1109/TIFS.2011.2134094.

Fridrich, J. (2009): *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge: Cambridge University Press.

Fridrich, J.; Goljan, M.; Lisonek, P.; Soukal, D. (2005): Writing on wet paper. *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3923-3935. DOI 10.1109/TSP.2005.855393.

Li, F.; Wu, K.; Lei, J.; Wen, M.; Bi, Z. et al. (2015): Steganalysis over large-scale social networks with high-order joint features and clustering ensembles. *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 344-357. DOI 10.1109/TIFS.2015.2496910.

Li, F.; Wu, K.; Zhang, X.; Yu, J.; Lei, J. et al. (2018): Robust batch steganography in social networks with non-uniform payload and data decomposition. *IEEE Access*, vol. 6, pp. 29912-29925. DOI 10.1109/ACCESS.2018.2841415.

Meng, R.; Cui, Q.; Yuan, C. (2018): A survey of image information hiding algorithms based on deep learning. *Computer Modeling in Engineering & Sciences*, vol. 117, no. 3, pp. 425-454. DOI 10.31614/cmes.2018.04765.

Otori, H.; Kuriyama, S. (2009): Texture synthesis for mobile data communications. *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 74-81. DOI 10.1109/MCG.2009.127.

Qian, Z.; Huang, N.; Li, S.; Zhang, X. (2018): Constructive steganography using texture synthesis. *IETE Technical Review*, vol. 35, no. sup1, pp. 14-22. DOI 10.1080/02564602.2018.1475267.

Qian, Z.; Pan, L.; Li, S.; Zhang, X. (2018). Steganography by constructing marbling texture. *International Conference on Cloud Computing and Security*, pp. 428-439.

Westfeld, A. (2001): F5-a steganographic algorithm. *International Workshop on Information Hiding*, pp. 289-302.

Wu, K. C.; Wang, C. M. (2014): Steganography using reversible texture synthesis. *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 130-139.

Yang, J.; Ruan, D.; Huang, J.; Kang, X.; Shi, Y. Q. (2019): An embedding cost learning framework using GAN. *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 839-851. DOI 10.1109/TIFS.2019.2922229.

Zhang, S.; Su, S.; Li, L.; Zhou, Q.; Lu, J. et al. (2019): An image style transfer network using multilevel noise encoding and its application in coverless steganography. *Symmetry*, vol. 11, no. 9, pp. 1152. DOI 10.3390/sym11091152.

Zhang, X.; Zhang, W.; Wang, S. (2007): Efficient double-layered steganographic embedding. *Electronics Letters*, vol. 43, no. 8, pp. 482-483. DOI 10.1049/el:20070248.

Zhou, H.; Chen, K.; Zhang, W.; Yu, N. (2017): Comments on steganography using reversible texture synthesis. *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1623-1625. DOI 10.1109/TIP.2017.2657886.

Zhou, Q.; Qiu, Y.; Li, L.; Lu, J.; Yuan, W. et al. (2018): Steganography using reversible texture synthesis based on seeded region growing and LSB. *Computers, Materials & Continua*, vol. 55, no. 1, pp. 151-163.