Steganography Using Reversible Texture Synthesis

Kuo-Chen Wu and Chung-Ming Wang, *Member, IEEE*

***Abstract*—We propose a novel approach for steganography using a reversible texture synthesis. A texture synthesis process re-samples a smaller texture image which synthesizes a new texture image with a similar local appearance and arbitrary size. We weave the texture synthesis process into steganography to conceal secret messages. In contrast to using an existing cover image to hide messages, our algorithm conceals the source texture image and embeds secret messages through the process of texture synthesis. This allows us to extract secret messages and the source texture from a stego synthetic texture. Our approach offers three distinct advantages. First, our scheme offers the embedding capacity that is proportional to the size of the stego texture image. Second, a steganalytic algorithm is not likely to defeat our steganographic approach. Third, the reversible capability inherited from our scheme provides functionality which allows recovery of the source texture. Experimental results have verified that our proposed algorithm can provide various numbers of embedding capacities, produce a visually plausible texture images, and recover the source texture.**

***Index Terms*—Data embedding, example-based approach, reversible, steganography, texture synthesis.**

I. Introduction

IN the last decade many advances have been made in the area of digital media, and much concern has arisen regarding steganography for digital media. Steganography [1] a singular method of information hiding techniques. It embeds messages into a host medium in order to conceal secret messages so as not to arouse suspicion by an eavesdropper [2]. A typical steganographic application includes covert communications between two parties whose existence is unknown to a possible attacker and whose success depends on detecting the existence of this communication [3]. In general, the host medium used in steganography includes meaningful digital media such as digital image, text, audio, video, 3D model [4], etc. A large number of image steganographic algorithms have been investigated with the increasing popularity and use of digital images [5], [6].

This work was supported in part by the Minister of Science and Technology, Taiwan under Grant 101-2221-E-005-091-MY3.

The authors are with the Department of Computer Science and Engineering, National Chung Hsing University, 250 Kuo Kuang Road, Taichung 40227, Taiwan (e-mail: phd9501@cs.nchu.edu.tw; cmwang@cs.nchu.edu.tw).
Corresponding author: Chung-Ming Wang.

Most image steganographic algorithms adopt an existing image as a cover medium. The expense of embedding secret messages into this cover image is the image distortion encountered in the stego image. This leads to two drawbacks. First, since the size of the cover image is fixed, the more secret messages which are embedded allow for more image distortion. Consequently, a compromise must be reached between the embedding capacity and the image quality which results in the limited capacity provided in any specific cover image. Recall that image steganalysis is an approach used to detect secret messages hidden in the stego image. A stego image contains some distortion, and regardless of how minute it is, this will interfere with the natural features of the cover image. This leads to the second drawback because it is still possible that an image steganalytic algorithm can defeat the image steganography and thus reveal that a hidden message is being conveyed in a stego image.

In this paper, we propose a novel approach for steganography using reversible texture synthesis. A texture synthesis process re-samples a small texture image drawn by an artist or captured in a photograph in order to synthesize a new texture image with a similar local appearance and arbitrary size. We weave the texture synthesis process into steganography concealing secret messages as well as the source texture. In particular, in contrast to using an existing cover image to hide messages, our algorithm conceals the source texture image and embeds secret messages through the process of texture synthesis. This allows us to extract the secret messages and the source texture from a stego synthetic texture. To the best of our knowledge, steganography taking advantage of the reversibility has ever been presented within the literature of texture synthesis.

Our approach offers three advantages. First, since the texture synthesis can synthesize an arbitrary size of texture images, the embedding capacity which our scheme offers is proportional to the size of the stego texture image. Secondly, a steganalytic algorithm is not likely to defeat this steganographic approach since the stego texture image is composed of a source texture rather than by modifying the existing image contents. Third, the reversible capability inherited from our scheme provides functionality to recover the source texture. Since the recovered source texture is exactly the same as the original source texture, it can be employed to proceed onto the second round of secret messages for steganography if needed. Experimental results have verified that our proposed algorithm can provide various numbers of embedding capacities, produce visually plausible texture images, and recover the source texture. Theoretical

analysis indicates that there is an insignificant probability of breaking down our steganographic approach, and the scheme can resist an RS steganalysis attack [7].

The remainder of this paper is organized as follows: in Section II, we review the texture synthesis techniques. In Section III, we detail our algorithm including embedding and extracting procedures. We describe experimental results and theoretical analysis in Section IV, followed by our conclusions and future work presented in the final section.

II. Related Works

Texture synthesis has received a lot of attention recently in computer vision and computer graphics [8]. The most recent work has focused on texture synthesis by example, in which a source texture image is re-sampled using either pixel-based or patch-based algorithms to produce a new synthesized texture image with similar local appearance and arbitrary size.

Pixel-based algorithms [9], [10], [11] generate the synthesized image pixel by pixel and use spatial neighborhood comparisons to choose the most similar pixel in a sample texture as the output pixel. Since each output pixel is determined by the already synthesized pixels, any wrongly synthesized pixels during the process influence the rest of the result causing propagation of errors.

Otori and Kuriyama [12], [13] pioneered the work of combining data coding with pixel-based texture synthesis. Secret messages to be concealed are encoded into colored dotted patterns and they are directly painted on a blank image. A pixel-based algorithm coats the rest of the pixels using the pixel-based texture synthesis method, thus camouflaging the existence of dotted patterns. To extract messages the printout of the stego synthesized texture image is photographed before applying the data-detecting mechanism. The capacity provided by the method of Otori and Kuriyama depends on the number of the dotted patterns. However, their method had a small error rate of the message extraction.

Patch-based algorithms [14], [15] paste patches from a source texture instead of a pixel to synthesize textures. This approach of Cohen *et al.* and Xu *et al.* improves the image quality of pixel-based synthetic textures because texture structures inside the patches are maintained. However, since patches are pasted with a small overlapped region during the synthetic process, one needs to make an effort to ensure that the patches agree with their neighbors.

Liang *et al.* [16] introduced the patch-based sampling strategy and used the feathering approach for the overlapped areas of adjacent patches. Efros and Freeman [17] present a patch stitching approach called "image quilting". For every new patch to be synthesized and stitched, the algorithm first searches the source texture and chooses one candidate patch that satisfies the pre-defined error tolerance with respect to neighbors along the overlapped region. Next, a dynamic programming technique is adopted to disclose the minimum error path through the overlapped region. This declares an optimal boundary between the chosen candidate patch and the synthesized patch, producing visually plausible patch stitching.

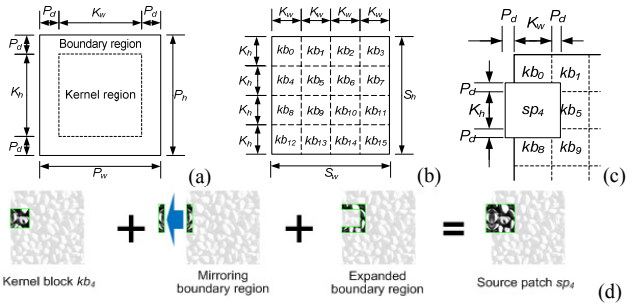

Fig. 1. Patch, kernel blocks, and source patch. (a) The diagram of a patch. The central part of a patch is the kernel region; the other part around the kernel region is the boundary region. (b) An illustration of non-overlapped kernel blocks subdivided from the source texture. (c) The diagram of source patches derived by the expanding process using kernel blocks. (d) The boundary mirroring and expanding for a source patch.

Ni *et al.* [18] proposed an image reversible data hiding algorithm which can recover the cover image without any distortion from the stego image after the hidden data have been extracted. Histogram shifting is a preferred technique among existing approaches of reversible image data hiding because it can control the modification to pixels, thus limiting the embedding distortion, and it only requires a small size location map, thereby reducing the overhead encountered. The current state-of-the-art for reversible image data hiding is the general framework presented by Li *et al.* [19].

To the best of our knowledge, we were unable to disclose any literature that related patch-based texture synthesis with steganography. In this paper, we present our work which takes advantage of the patch-based methods to embed a secret message during the synthesizing procedure. This allows the source texture to be recovered in a message extracting procedure, providing the functionality of reversibility. We detail our method in the next section.

III. Proposed Method

We illustrate our proposed method in this section. First, we will define some basic terminology to be used in our algorithm. The basic unit used for our steganographic texture synthesis is referred to as a "patch." A patch represents an image block of a source texture where its size is user-specified. Fig. 1(a) illustrates a diagram of a patch. We can denote the size of a patch by its width (P_w) and height (P_h). A patch contains the central part and an outer part where the central part is referred to as the kernel region with size of $K_w \times K_h$, and the part surrounding the kernel region is referred to as the boundary region with the depth (P_d).

Next, we describe the concept of the kernel block. Given a source texture with the size of $S_w \times S_h$ we can subdivide the source texture into a number of non-overlapped kernel blocks, each of which has the size of $K_w \times K_h$, as shown as Fig. 1(b). Let KB represent the collection of all kernel blocks thus generated, and $||KB||$ represent the number of elements in this set. We can employ the indexing for each source patch kb_i, i.e., $KB=\{kb_i | i = 0 \text{ to } ||KB||-1\}$. As an example, given a source texture with the size of $S_w \times S_h = 128 \times 128$, if we set the size $K_w \times K_h$ as 32×32, then we can generate $||KB||=16$ kernel blocks. Each element in KB

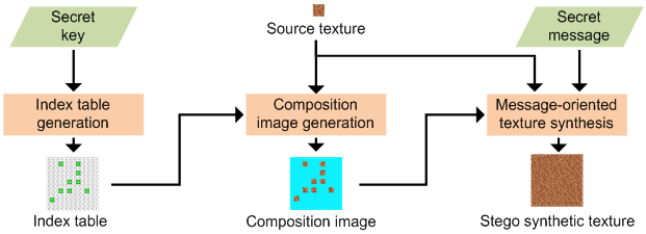

Fig. 2. The flowchart of the three-process message embedding procedure.

can be identified as $\{kb_0, kb_1, \ldots, kb_{15}\}$.

We can expand a kernel block with the depth P_d at each side to produce a source patch. The expanding process will overlap its neighbor block. Fig. 1(c) indicates the boundary region of source patch sp_4 when we expand the kernel block kb_4 to overlap the kernel blocks kb_0, kb_1, kb_5, kb_8, and kb_9. If a kernel block is located around the boundary of a source texture, we operate the boundary mirroring using the kernel block's symmetric contents to produce the boundary region, as shown in Fig. 1(d) for the kernel block kb_4.

Similar to the kernel block, we can denote SP as the collection of all source patches and SP_n=||SP|| as the number of elements in the set SP. We can employ the indexing for each source patch sp_i, i.e., SP={sp_i| i = 0 to ||SP||-1}.

Given a source texture with the size of S_w×S_h, we can derive the number of source patches SP_n using (1) if a kernel block has the size of K_w×K_h. In our paper, we assume the size of the source texture is a factor of the size of the kernel block to ease the complexity.

$$SP_n = \frac{S_w}{K_w} \times \frac{S_h}{K_h} \tag{1}$$

Our steganographic texture synthesis algorithm needs to generate candidate patches when synthesizing synthetic texture. The concept of a candidate patch is trivial: we employ a window P_w×P_h and then travel the source texture (S_w×S_h) by shifting a pixel each time following the scan-line order. Let CP={cp_i|i=0, 1, …, CP_n-1} represent the set of the candidate patches where CP_n=||CP|| denotes the number of elements in CP. We can derive CP_n using (2).

$$CP_n = \|CP\| = (S_w - P_w + 1) \times (S_h - P_h + 1) \tag{2}$$

When generating a candidate patch, we need to ensure that each candidate patch is unique; otherwise, we may extract an incorrect secret message. In our implementation, we employ a flag mechanism. We first check whether the original source texture has any duplicate candidate patches. For a duplicate candidate patch, we set the flag on for the first one. For the rest of the duplicate candidate patches we set the flag off to ensure the uniqueness of the candidate patch in the candidate list.

A. Message Embedding Procedure

In this section we will illustrate the message embedding procedure. Fig. 2 shows the three processes of our message embedding procedure. We will detail each process in the following sections.

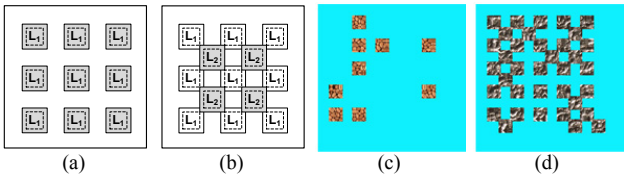

Fig. 3. An illustration of two types of priority locations to paste the source patches: (a) the first-priority locations L1 and (b) the second-priority locations L2; (c) 9 source patches using L1 resulting in sparse distribution; (d) 36 source patches using L1 first, and then employing the L2 leading to the dense distribution.

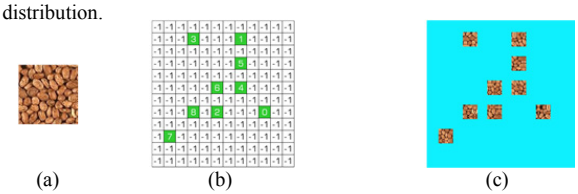

Fig. 4. An illustration of composition image; (a) the source texture (96×96), (b) the index table after patch distribution, (c) the composition image (488×488) by referring (a) and (b).

1) Index Table Generation Process

The first process is the index table generation where we produce an index table to record the location of the source patch set SP in the synthetic texture. The index table allows us to access the synthetic texture and retrieve the source texture completely. Such a reversible embedding style reveals one of the major benefits our proposed algorithm offers.

We first determine the dimensions of the index table (T_{pw}×T_{ph}). Given the parameters T_w and T_h , which are the width and the height of the synthetic texture we intend to synthesize, the number of entries in this index table can be determined using (3) where TP_n denotes the number of patches in the stego synthetic texture. For simplicity, we chose appropriate parameters for T_w, T_h, P_w, P_h, and P_d, so that the number of entries is an integer. As an example, if T_w×T_h=488×488, P_w×P_h=48×48, and P_d=8, then we can generate an index table (12×12) containing 144 entries.

$$TP_n = T_{pw} \times T_{ph} = \left\lfloor \frac{(T_w - P_w)}{(P_w - P_d)} + 1 \right\rfloor \times \left\lfloor \frac{(T_h - P_h)}{(P_h - P_d)} + 1 \right\rfloor \tag{3}$$

When we distribute source texture to achieve the manner of reversibility, the source patches can be distributed in a rather sparse manner if the synthetic texture has a resolution that is much larger than that of the source texture, as shown in Fig. 3(a). On the contrary, the source patches may be distributed in a rather dense manner if the synthetic texture has a resolution that is slightly larger than that of the source texture, as shown in Fig. 3(b). For the patch distribution, we avoid positioning a source texture patch on the borders of the synthetic texture. This will encourage the borders to be produced by message-oriented texture synthesis, enhancing the image quality of the synthetic texture. We further define the first-priority position L_1 and the second-priority position L_2, for two types of priority locations where ||L_1|| and ||L_2||, dervied in (4), represent the number in the first-priority and second-priority positions, respectively.

$$\begin{cases} \|L_1\| = \left\lfloor \frac{T_{pw}-2}{2} \right\rfloor \times \left\lfloor \frac{T_{ph}-2}{2} \right\rfloor \\ \|L_2\| = \left\lfloor \frac{T_{pw}-2}{2} \right\rfloor \times \left\lfloor \frac{T_{ph}-2}{2} \right\rfloor \end{cases} \quad (4)$$

Fig. 4 illustrates the composition image where nine source patches are pasted. As an example, an index table $T_{pw} \times T_{ph}$=12×12 contains 144 entries so $\|L_1\|$=25 and $\|L_2\|$=25.

Given the number of patches SP_n subdivided from the source texture, the strategy of patch distribution is to distribute patches perfectly on the first-priority positions before posting patches on the second-priority positions. Based on the resolution of the synthetic texture, we will have two cases: the sparse distribution and dense distribution. These are described below.

When the number of source patches is less than or equal to the number of the first-priority positions ($SP_n \leq \|L_1\|$), the patch will be distributed sparsely. In the security issue section we describe some mathematical analyses of our algorithm. The analysis shows that the total number of patterns that the sparse distribution can offer is $C_{SP_n}^{\|L_1\|} \times SP_n!$. On the contrary, when the number of source patches is greater than the first-priority position ($SP_n > \|L_1\|$), the patch will be distributed densely. A mathematical analysis shows that the total number of patterns that the dense distribution can offer is $C_{SP_n-\|L_1\|}^{\|L_2\|} \times SP_n!$.

The index table has the initial values of -1 for each entry, which shows that the table is blank. Now, we need to re-assign values when we distribute the source patch ID in the synthetic texture. In our implementation, we employ a random seed for patch ID distribution, which increases the security of our steganographic algorithm making it more difficult for malicious attackers to extract the source texture. As a result, the index table will be scattered with different values as shown in Fig. 4(b) where we have nine source patches (no. 0 to 8) and 135 blank locations with the initial value of "-1". In this index table, the entries with non-negative values indicate the corresponding source patch ID subdivided in the source texture, while these entries with the value of -1 represent that the patch positions will be synthesized by referring to the secret message in the message-oriented texture synthesis. Taking the above condition into consideration, we can now use the random seed R_s to disarrange the ID of the source patches subdivided in the source texture. For example, if there are nine source patches (SP_n=9) and the synthetic texture is synthesized with a total number of 144 patches (TP_n=144), we can distribute the disarranged nine IDs of the source patches resulting in a sparse distribution. Secret messages will be encoded in the remaining 135 blank locations during the message-oriented texture synthesis.

2) Patch Composition Process

The second process of our algorithm is to paste the source patches into a workbench to produce a composition image. First, we establish a blank image as our workbench where the size of the workbench is equal to the synthetic texture. By referring to the source patch IDs stored in the index table, we then paste the source patches into the workbench. During the pasting process, if no overlapping of the source patches is encountered, we paste the source patches directly into the workbench, as shown in Fig. 3(c). However, if pasting locations cause the source patches to overlap each other, we employ the image quilting technique [17] to reduce the visual artifact on the overlapped area.

TABLE I
A FUNDAMENTAL OF THREE DIFFERENCES BETWEEN THE CONVENTIONAL TEXTURE SYNTHESIS AND OUR MESSAGE-ORIENTED TEXTURE SYNTHESIS.

	Conventional Texture Synthesis	Message-oriented Texture Synthesis
Shape of the Overlapped Area	L-shape only	Five different shapes
Candidate Selection	A threshold with random selection	Referring to the secret message
Synthesized Results	A pure large texture	A large texture containing source texture and messages

3) Message-oriented Texture Synthesis Process

We have now generated an index table and a composition image, and have pasted source patches directly into the workbench. We will embed our secret message via the message-oriented texture synthesis to produce the final stego synthetic texture.

The three fundamental differences between our proposed message-oriented texture synthesis and the conventional patch-based texture synthesis are described in Table I. The first difference is the shape of the overlapped area. During the conventional synthesis process, an L-shape overlapped area is normally used to determine the similarity of every candidate patch. In contrast, the shape of the overlapped area in our algorithm varies because we have pasted source patches into the workbench. Consequently, our algorithm needs to provide more flexibility in order to cope with a number of variable shapes formed by the overlapped area.

The second difference lies in the strategy of candidate selection. In conventional texture synthesis, a threshold rank is usually given so that the patch can be randomly selected from candidate patches when their ranks are smaller than the given threshold. In contrast, our algorithm selects "appropriate" patches by taking into consideration secret messages. Finally, the output of the conventional texture synthesis is a pure synthetic texture. However, our algorithm produces a much different synthetic texture. The source texture being converted into a number of source patches has been pasted as part of the contents in the large synthetic texture. In addition, the output large texture has been concealed with the secret message.

While the conventional texture synthesis algorithm has an "L-shape" overlapped area, our algorithm may acquire another four shapes of the overlapped area, as shown in Fig. 5. Assume that the texture synthesis is carried on using the scan-line order. The texture area reveals a typical "L-shape" of an overlapped area, as shown in Fig. 5(a). However, when a nearby pasted source patch has occupied the right side of the working location, this leads to a "downward U-shape" of the overlapped area (Fig. 5(b)). In addition, if a nearby pasted source patch has occupied the bottom side, this leads to a "rightward U-shape" of the overlapped area (Fig. 5(c)). If a nearby pasted source patch has occupied the lower right corner of the working location, this leads to a disjointed overlapped area containing an "L-shape" and a small but isolated part (Fig. 5(d)). Finally, if two nearby

Fig. 5. Five different shapes of the overlapped area may occur during our message-oriented texture synthesis algorithm. Clear textures inside the green square frame represent textures that have been synthesized or textures that are overlapped with the pasted source patches. The blank area inside the green square frame represents the working location which is going to be synthesized.

pasted source patches have occupied the right and bottom side of the working location, this will contribute to an "O-shape" of the overlapped area (Fig. 5(e)).

For each candidate patch within the candidate list, one of the five shapes of overlapped area described above will occur when referring to the synthesized area in the working location. Thus, we can compute the mean square error (MSE) of the overlapped region between the synthesized area and the candidate patch. After all MSEs of the patches in the candidate list are determined, we can further rank these candidate patches according to their MSEs. Fig. 6 demonstrates an example of a candidate list where four candidate patches have different MSEs. The candidate patch which has the smallest MSE indicates that this candidate patch is the most similar to the synthesized area in the working location.

Once the ranks of all candidate patches are determined, we can select the candidate patch where its rank equals the decimal value of an n-bit secret message. In this way, a segment of the n-bit secret message has been concealed into the selected patch to be pasted into the working location. In our implementation, we employ a simple but effective image quilting technique [17] to reduce the visual artifact encountered in the overlapped area.

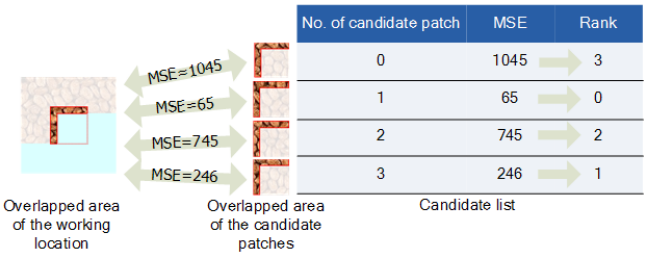

Fig. 6. The MSEs and rank of patches in the candidate list for the working location.

TABLE II
THE EMBEDDING CAPACITY PROVIDED BY OUR ALGORITHM

$BPP_{max} = \lfloor log_2[(S_w - P_w + 1) \times (S_h - P_h + 1)] \rfloor$
$SP_n = \frac{S_w}{P_w - (2 \times P_d)} \times \frac{S_h}{P_h - (2 \times P_d)}$
$TP_n = \left\lfloor \frac{(T_w - P_w)}{(P_w - P_d)} + 1 \right\rfloor \times \left\lfloor \frac{(T_h - P_h)}{(P_h - P_d)} + 1 \right\rfloor$
$EP_n = TP_n - SP_n$
$TC = \text{BPP} \times EP_n$

BPP_{max}: the maximal capacity in bits that can be concealed at each patch.
S_w: width of source texture, S_h: height of source texture.
P_w: width of a patch, P_h: height of a patch.
SP_n: the number of source patches subdivided in the source texture.
P_d: boundary depth of a patch.
TP_n: number of patches in the synthetic texture.
T_w: width of synthetic texture, T_h: height of synthetic texture.
EP_n: number of embeddable patches in the stego synthetic texture.
TC: total embedding capacity.

B. Capacity Determination

The embedding capacity is one concern of the data embedding scheme. Table II summarizes the equations we described to analyze the embedding capacity our algorithm can offer. The embedding capacity our algorithm can offer is related to the capacity in bits that can be concealed at each patch (BPP, bit per patch), and to the number of embeddable patches in the stego synthetic texture (EP_n). Each patch can conceal at least one bit of the secret message; thus, the lower bound of BPP will be 1, and the the maximal capacity in bits that can be concealed at each patch is the upper bound of BPP, as denoted by BPP_{max}. In contrast, if we can select any rank from the candidate list, the upper bound of BPP will be $\lfloor \log_2(CP_n) \rfloor$. The total capacity ($TC$) our algorithm can offer is shown in (5) which is the multiplciation of BPP and EP_n. The number of the embeddable patches is the difference between the number of patches in the synthetic texture (TP_n) and the number of source patches subdivided in the source texture (SP_n).

$$TC = \text{BPP} \times EP_n = \text{BPP} \times (TP_n - SP_n) \quad (5)$$

Suppose we provide a source texture $S_w \times S_h$ =128×128, and we intend to generate a synthetic texture $T_w \times T_h$=488×488. We specify the patch size $P_w \times P_h$=48×48 and the boundary depth P_d =8 pixels. This will lead to the range of the BPP between 1 and 12. We can produce SP_n=16 source patches and TP_n=144 patches on the stego synthetic texture. Thus, there are EP_n =128 embeddable patches. If we take the BPP=12, then the total embedding capacity is TC=1536 bits.

C. Source Texture Recovery, Message Extraction, and Message Authentication Procedure

The message extracting for the receiver side involves generating the index table, retrieving the source texture, performing the texture synthesis, and extracting and authenticating the secret message concealed in the stego synthetic texture. The extracting procedure contains four steps, as shown in Fig. 7.

Given the secret key held in the receiver side, the same index table as the embedding procedure can be generated. The next step is the source texture recovery. Each kernel region with the size of $K_w \times K_h$ and its corresponding order with respect to the size of $S_w \times S_h$ source texture can be retrieved by referring to the index table with the dimensions $T_{pw} \times T_{ph}$. We can then arrange kernel blocks based on their order, thus retrieving the recovered source texture which will be exactly the same as the source texture. In the third step, we apply the composition image generation to paste the source patches into a workbench to produce a composition image by referring to the index table. This generates a composition image that is identical to the one produced in the embedding procedure.

The final step is the message extraction and authentication step, which contains three sub-steps. The first sub-step constructs a candidate list based on the overlapped area by referring to the current working location. This sub-step is the

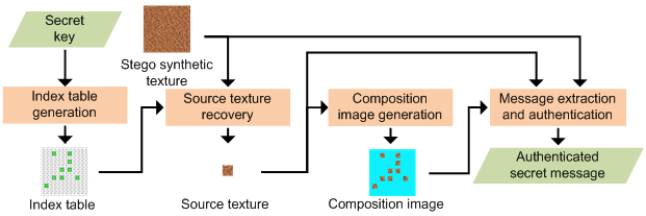

Fig. 7. The flowchart of the four-step message extracting procedure.

same as the embedding procedure, producing the same number of candidate lists and their corresponding ranks.

The second sub-step is the match-authentication step. Given the current working location *Cur(WL)* on the workbench, we refer to the corresponding stego synthetic texture at the same working location *Stg(WL)* to determine the stego kernel region $SK_w \times SK_h$. Then, based on this stego kernel region, we search the candidate list to determine if there is a patch in the candidate list where its kernel region is the same as this stego kernel region. If this patch is available, we refer to it as the matched patch, and denote it as $MK_w \times MK_h$. Clearly, we can locate the rank R of the matched patch, and this rank represents the decimal value of the secret bits we conveyed in the stego patch when operating the texture synthesis in the message embedding procedure. However, if we cannot disclose any matched patch in the candidate list where the kernel region is the same as the stego kernel region, it means that the stego kernel region has been tampered with, leading to a failure of the message authentication. In this way, we can authenticate and extract all of the secret messages that are concealed in the stego synthetic texture patch by patch.

Our method is resistant against malicious attacks as long as the contents of the stego image are not changed. With some side information, for example, our scheme can survive the attacks of the image mirroring or image rotation by 90, 180, or 270 degrees. Nevertheless, if malicious attacks lead to alteration of the contents of the stego texture image, the message authentication step will justify the authenticity of the secret messages.

IV. Experimental Results And Analysis

A. Results of the Embedding Capacity

We collect our experimental results on a personal computer with an i7-2600 3.4GHz CPU and 4GB memory. We adopt four source textures for the results of our collection. Table III presents the total embedding capacity our algorithm can provide when different resolutions of the synthetic texture are produced by concealing various BPPs. It is interesting to point out that given a fixed number of BPP, the larger the resolutions of the source texture $S_w \times S_h$ (96×96 vs. 192×192), the smaller the total embedding capacity (TC) our algorithm will offer (6160 bits vs. 5890 bits for 10 BPP). This is because the larger source texture will contain more source patches SP_n (9 vs. 36) that we need to paste which cannot conceal any secret bits. This will reduce the number of embeddable patches (EP_n) on the composition image (616 vs. 589), thus reducing the total embedding capacity. Nevertheless, we can employ larger BPP (11 vs. 14) in order to convey more secret messages (6776 bits vs. 8246 bits). The maximal capacity provided by our algorithm is 34398 bits.

TABLE III
Total Embedding Capacity in Bits our Algorithm can Provide

Synthetic texture size: $T_w \times T_h$ =1008×1008; Patch size: $P_w \times P_h$ =48×48; Boundary depth: P_d =8					
$S_w \times S_h$	SP_n	EP_n	TC (5BPP)	TC (10 BPP)	TC (BPP_{max})
96×96	9	616	3080	6160	6776
128×128	16	609	3045	6090	7308
192×192	36	589	2945	5890	8246
$T_w \times T_h$ =1024×1024, $P_w \times P_h$ =24×24, P_d =4					
96×96	36	2565	12825	25650	30780
128×128	64	2537	12685	25370	32981
192×192	144	2457	12285	24570	34398

TABLE IV
Computing Time (second)

Capacity	Pure	4 BPP	5 BPP	8 BPP	10 BPP	12 BPP
Rope net (192×192)	1562	1680	1557	1680	1541	1680
Metal (192×192)	1671	1816	1665	1768	1644	1816
Peanuts (96×96)	141	141	141	141	136	N/A
Ganache (128×128)	385	402	385	402	385	411

Patch size: $P_w \times P_h$=48×48, boundary depth: P_d =8

We compare our embedding capacity with the work presented by Otori and Kuriyama [13]. In their algorithm, a data-coded texture image of 480×640 pixels contains 5×5 or 10×10 dotted patterns, thus concealing the capacity in the range of 200 to 800 bits. In contrast, our scheme conveys the capacity ranging from 12285 to 34398 bits in a stego texture synthesis image of 1024×1024 pixels. Consequently, the capacity we offer varies from 4.50 to 50.39 times more than our counterparts'. Besides, our scheme extracts the secret messages correctly, while their scheme exhibits a small error rate when extracting secret messages.

Fig. 8 compares stego synthetic textures with different capacities. We also present the pure synthetic texture which does not convey any secret message. No significant visual difference exists between the two stego synthetic textures and the pure synthetic texture. In addition, no significant visual difference can be perceived when comparing two stego synthetic textures embedded by 5 BPP vs. 10 BPP.

The computing times of Fig. 8 are shown in Table IV. The range of the computing time is 6.8% to 8.7% more than that needed for pure texture synthesis. Nevertheless, our algorithm is flexible enough to provide a different embedding capacity by simply altering the patch size, satisfying the desirable capacity or texture quality demanded by users. Also, our algorithm can retrieve the source texture, making possible the second round of message embedding using the recovered source texture.

B. Image Quality Comparison

We use several mechanisms to determine the image quality of the stego synthetic texture. We define the first measurement, which is called the mean squared error of the overlapped area (MSEO) to determine the image quality of the synthetic results. MSEO reflects the similarity between the candidate patch and the synthesized area where we will specifically operate the

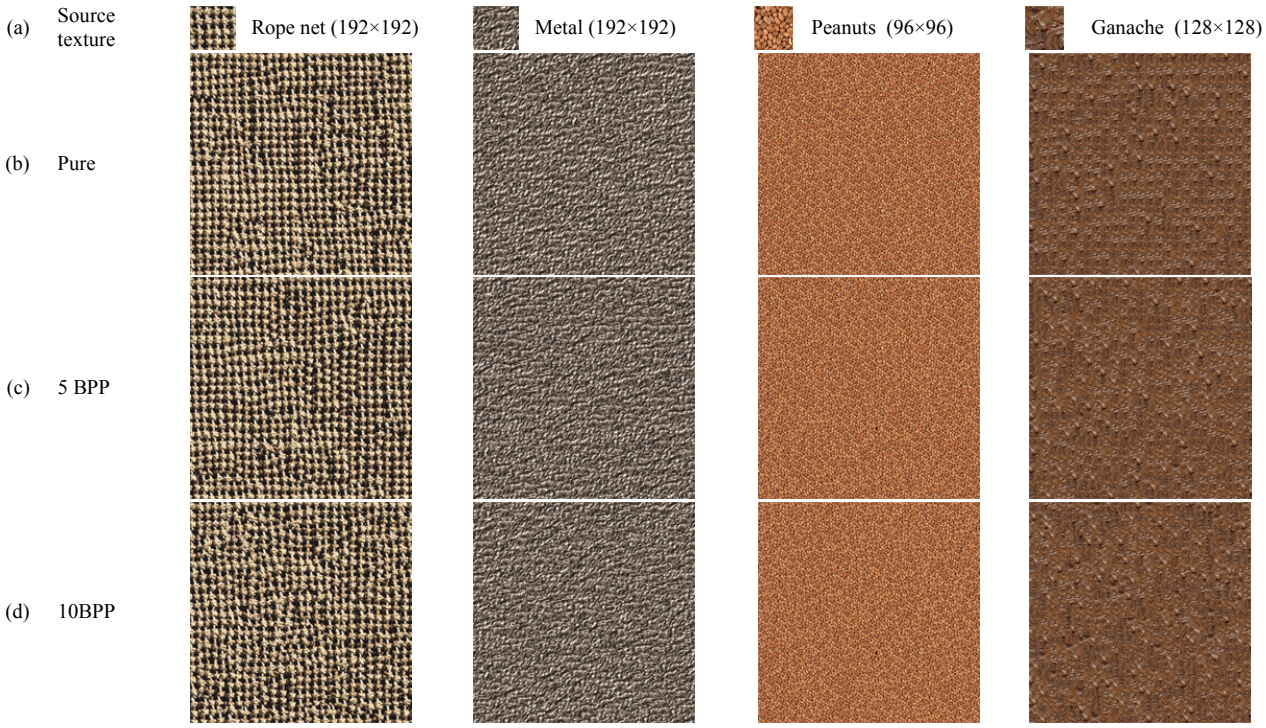

Fig. 8. Comparison results for the pure synthetic texture and two stego synthetic textures with different BPP: (a) source textures, (b) pure synthetic textures, (c) stego textures with 5 BPP, (d) stego textures with 10 BPP. No significant visual differences exist for these three synthetic textures. The patch size is $P_w \times P_h$ =48×48, the boundary depth is P_d =8, and the resolutions of the stego synthetic textures are $T_w \times T_h$ =1008×1008.

image quilting technique during the message-oriented texture synthesis process. Consequently, the MSEO has a non-zero value even in the case of the pure patch-based texture synthesis. If the MSEO produces a small value, it implies that the synthetic texture shows a high image quality of the overlapped areas. Obviously, the lower the MSEO value, the higher quality of the synthetic texture image. The equation of MSEO is shown in (6), where OL_i stands for the overlapped area of the working location, i, $\mathrm{p}_j^{\mathrm{c}}$ stands for the pixel j of the candidate patch in OL_i, and $\mathrm{p}_j^{\mathrm{s}}$ stands for the pixel j of the synthesized area in OL_i.

$$\mathrm{MSEO} = \frac{1}{(EP_n)\times P_w \times P_h} \sum_{i=1}^{EP_n} \sum_{j \in \mathrm{OL}_i} (\mathrm{p}_j^{\mathrm{c}} - \mathrm{p}_j^{\mathrm{s}})^2 \qquad (6)$$

The MSEO can be calculated during the texture synthesis procedure. When we synthesize a patch, we accumulate the squared errors of all the pixels in the overlapped area between the synthesized area and the selected candidate patch. When the texture synthesis procedure is finished, we can divide the summation of squared errors by the patch size and the number of the synthesized patches to produce the MSEO. It comes as no surprise that the MSEO is dependent on the capacity per patch.

Table V compares the MSEO to the embedding capacity for the four test source textures. When we consider the same patch size, the MSEO increases as we convey more secret bits per patch. These MSEO values coincide with the image quality when visualizing the stego synthetic texture. For a larger patch size, the boundary depth increases leading to a larger MSEO. Taking the test source texture "peanuts" as an example, the MSEO is 1034.6 when conveying 5 bits of secret message per patch, while it shows the MSEO of 1236.9 when conveying 10 bits per patch. Fig. 8(c) shows a slightly higher visual image quality than Fig. 8(d).

The second scheme we adopt for measuring the image quality is the Pearson Product Moment Correlation (PPMC) [20]. It is employed as a measure of how well two variables are related. The Pearson coefficient values that this scheme produces are between 1 and -1. A result of 1 means that there is a perfect positive correlation between the two variables, while a result between 0.5 and 1.0 represents a high correlation.

Table VI shows average Pearson coefficients across three R-G-B channels measured under the base of the histogram produced for each channel. In Case-A and Case-B, we reveal the Pearson coefficients between the stego synthetic textures and the pure synthetic texture. We report in Case-C the Pearson coefficients between two stego synthetic textures with different BPPs. The Pearson coefficients shown in Case-A are very close to 1.0. This represents the high correlation between the pure and stego synthetic textures. In Case-B, although the Pearson coefficient values slightly decrease along with the increase of the embedding capacity per patch, they retain the feature of a close relation between the pure and stego synthetic textures. The Pearson coefficients in Case-C are very close as well, which indicates that the stego synthetic textures conveying 5 BPP remain highly correlated to those conveying 10 BPP. All the Pearson coefficients shown in this table exhibit that our steganographic algorithm is effective in producing stego synthetic textures of high image quality in comparison to pure synthetic textures produced by conventional texture synthesis.

Next, we employ the SSIM (Structural SIMilarity) index [21] to quantify the similarity between the pure and stego synthetic textures. The SSIM is an image quality assessment method for

TABLE V
A COMPARISON OF MSEO WITH RESPECT TO EMBEDDING CAPACITY

	Patch size: $P_w \times P_h$ =24×24 Boundary depth: P_d =4			Patch size: $P_w \times P_h$ =48×48 Boundary depth: P_d =8		
	Pure	5 BPP	10 BPP	Pure	5 BPP	10 BPP
Rope net	651.2	899.4	1327.6	936.3	1224.2	1565.3
Metal	595.6	795.2	1096.4	837.4	1132.3	1391.1
Peanuts	547.8	776.9	1053.1	918.2	1034.6	1236.9
Ganache	184.5	219.3	318.3	365.3	483.1	556.2

TABLE VI
COMPARISON OF PEARSON CORRELATION COEFFICIENTS

	Case-A	Case-B	Case-C
	Pure vs. 5 BPP	Pure vs. 10 BPP	5 BPP vs. 10 BPP
Rope net	0.9991	0.9991	0.9987
Metal	0.9990	0.9984	0.9995
Peanuts	0.9980	0.9974	0.9993
Ganache	0.9991	0.9955	0.9980

Patch size: $P_w \times P_h$=48×48, boundary depth: P_d =8.

TABLE VII
COMPARISON OF SSIM INDEX

	Patch size: $P_w \times P_h$ =24×24 Boundary depth: P_d =4 $T_w \times T_h$ =504×504		Patch size: $P_w \times P_h$ =48×48 Boundary depth: P_d =8 $T_w \times T_h$ =488×488	
Pure vs.	5 BPP	10 BPP	5 BPP	10 BPP
Rope net	0.31	0.27	0.34	0.28
Metal	0.28	0.26	0.28	0.26
Peanuts	0.08	0.08	0.08	0.07
Ganache	0.18	0.16	0.17	0.15

measuring the change in luminance, contrast, and structure in an image. The SSIM index is in the range of [-1, 1] and when it equals to 1, the two images are identical. Table VII shows the SSIM indices for different patch sizes. The SSIM index values are close despite their concealing different embedding capacities. The SSIM algorithm is highly sensitive to translation, scaling and rotation. This explains the low index values shown in Table VII because the patch-based method generates an entirely new synthetic texture image from a number of patches in the source texture resulting in a different level of image translation and scaling.

C. Results of Different Message Probabilities

The secret messages consist of bit stream "0" or "1." The experimental results shown so far consider that bit "0" or "1" has equal probability of appearance. However, equal probability may not be the case for some kinds of information to be served as a secret message. We generalize our scheme by considering the probability of appearance.

Let p represent the probability of appearance of secret bit "1." If we consider two bits of the secret message, then the probability for bit patterns "00" is $(1-p)\times(1-p)$. Let $ER(p,2)$ represent the expected rank of the selected candidate during our message-oriented texture synthesis with BPP=2. Then, $ER(p,2)$ can be derived as a function of probability p, as shown in (7).

TABLE VIII
THE PROBABILITY P TO THE MSEO OF THE STEGO SYNTHETIC TEXTURE

p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Rope net	1173	1289	1337	1448	1497	1509	1670	1691	1743
Metal	1133	1235	1301	1348	1379	1396	1417	1462	1473
Peanuts	1035	1116	1168	1202	1227	1246	1267	1290	1318
Ganache	456	507	535	539	553	555	561	577	582

Patch size: $P_w \times P_h$=48×48, boundary depth: P_d =8, capacity: 10 BPP

TABLE IX
THE MSEO USING THE PROBABILITY BALANCING STRATEGY

p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Rope net	1556	1568	1556	1546	1625	1570	1628	1598	1624
Metal	1397	1380	1418	1397	1405	1386	1397	1402	1381
Peanuts	1238	1241	1232	1238	1241	1238	1239	1237	1234
Ganache	554	560	555	553	560	563	556	560	557

Patch size: $P_w \times P_h$=48×48, boundary depth: P_d =8, capacity: 10 BPP

$$ER(p,2) = 0\times(1-p)(1-p) + 1\times(1-p)p + 2\times p(1-p) + 3\times p^2 = 3p \quad (7)$$

$$ER(p,k) = (2^k - 1)p \quad (8)$$

In general, when we consider BPP=k bits of the secret message, we can express the expected rank of a selected candidate as a more general form, as $ER(p, k)$ shown in (8). For example, if k=10 and p=0.5, ER(0.5,10)=511.5. However, if p=0.7, then ER(0.7,10)=716.1. If we select the larger rank, the larger MSEO would be expected. Consequently, the image quality of the synthetic texture is likely to be worse than the one produced by p=0.5. In general, if p is smaller, we are likely to select a better candidate of patch, which leads to producing a better stego synthetic texture. Thus, our algorithm is general enough to cope with the situation when given a variety of p values.

Table VIII shows the MSEO with the different probability p. The statistics show that this larger probability causes the increase of MSEO, as expected. This is due to the fact that when the secret message has the larger probability p, the candidate patch is likely to be selected at the rear part of the candidate list, leading to larger MSEO.

To improve the probability of getting better ranks we can conduct the "probability complement" process to decrease the MSEO. If the probability p is larger than 0.5, we adopt the probability of 1-p for message embedding. This can be realized by flipping every bit of the secret message to its reverse style. During the extraction stage, we extract the secret bit as usual, and then flip it again to produce the correct message. With this mechanism, we can ensure that the probability used for message embedding is always less than or equal to 0.5, which encourages our scheme to produce a stego synthetic texture with a smaller MSEO. Clearly, we need to adopt a secret key to record whether or not we have conducted any flipping operation.

If we are not able to determine the probability of the secret message, such as the example of the message being from a streaming source, then we can use a strategy referred to as "probability balancing" to produce a nearly even probability of p=0.5. In particular, we can conduct an XOR operation between the previous message clip M'_{i-1} and the current message clip M_i. For example, if we want to embed M_i to patch i, we do the XOR

operation and produce M_i' where $M_i' = M_i \oplus M_{i-1}'$. Then, we embed M_i' to patch i. In the extraction procedure, we recover the secret clip M_i with another XOR operation, where $M_i = M_i' \oplus M_{i-1}$. With this mechanism, we can produce a stego synthetic texture where its image quality is close to the one produced by p=0.5, free from the effect of message probability.

Table IX shows the effect of "probability balancing." The results show that all of the MSEOs are close to those produced with the probability p=0.5 regardless of using different message probabilities.

D. Security Issue

In this section we discuss the probability of breaking our steganographic algorithm in order to produce a correct secret message. We assume the resolution of the stego synthetic texture is known by the eavesdropper.

We explain our first level of security. For four possible squared sizes of the source texture ($S_w \times S_h$), the probability of breaking down is Pb_i=0.25. The probability of breaking down five possible squared sizes of the patch ($P_w \times P_h$) is Pb_p=0.2. Finally, the probability of breaking down ten possible BPPs from 2-bit to 12-bit is Pb_c =0.1. The total probability that the eavesdropper can break down the security of level one is $Pb_1^{on}=Pb_i \times Pb_p \times Pb_c=5 \times 10^{-3}$.

Next, we discuss the second level of security. An eavesdropper might be able to perceive the index table. If the length of the seed used to generate the index table is a 128-bit sequence, it is highly impossible that this seed will be revealed. The eavesdropper may try to speculate about the contents of the index table. For the case of the sparse distribution, the number of source patches (SP_n) is less than or equal to the number of the first-priority position ($\|L_1\|$). Thus, the total number of patterns offered is $C_{SP_n}^{\|L_1\|} \times SP_n!$. The probability of revealing the correct order of patches in the index table is shown in (9).

$$Pb_2^{sp} = \frac{(\|L_1\| - SP_n)!}{\|L_1\|!} \tag{9}$$

For the case of the dense distribution, the number of source patches is larger than the number of the first-priority position ($SP_n > \|L_1\|$). Therefore, the total number of permutation patterns that the dense distribution can offer is $C_{SP_n-\|L_1\|}^{\|L_2\|} \times SP_n!$. Thus, the probability of revealing the correct order of these IDs in the index table is shown in (10).

$$Pb_2^{de} = \frac{(SP_n - \|L_1\|)! \times (\|L_2\| - SP_n + \|L_1\|)!}{\|L_2\|! \times SP_n!} \tag{10}$$

We present two examples to provide more insight for the sparse distribution. Suppose $T_w \times T_h$ =488×488, $P_w \times P_h$=48×48, P_d=8, and $S_w \times S_h$=128×128, where the notations of these variables can be referred to Table II. The eavesdropper can retrieve the number of source patches as SP_n=16, the dimensions of the index table as $T_{pw} \times T_{ph}$=12×12, the number of first-priority positions $\|L_1\|$=25, and the number of second-priority positions $\|L_2\|$=25. An eavesdropper can realize that the case of the sparse distribution is being employed. The probability that the index table is broken down is $P_2^{sp} \cong 2.35 \times 10^{-20}$. Even though the security of level one has collapsed, the probability that an eavesdropper can retrieve source texture will be $Pb_{bk}=Pb_1^{on} \times Pb_2^{sp}=1.18 \times 10^{-22}$.

For the dense distribution we employ a larger source texture $S_w \times S_h$=192×192. An eavesdropper can retrieve SP_n=36, $T_{pw} \times T_{ph}$=12×12, $\|L_1\|$=25, and $\|L_2\|$=25. The probability that an index table has broken down is $P_2^{de} \cong 8.70 \times 10^{-49}$. The overall probability that an eavesdropper can retrieve source texture will be $Pb_{bk}=Pb_1^{on} \times P_2^{de}=4.35 \times 10^{-51}$. Both examples illustrate that even though the security of level one has collapsed, there is insignificant probability that our steganographic system can be attacked to retrieve the source texture, and consequently, the secret message. The analysis indicates that our steganographic system is secure against a malicious attack.

TABLE X
THE RS STEGANALYSIS USING "ROPE NET" SOURCE TEXTURE

Channels	Capacity	*Rm+*	*Rm-*	*Sm+*	*Sm-*
R channel	pure	30.05	35.84	21.59	17.01
	5 BPP	30.48	35.77	21.66	16.99
	10 BPP	29.72	35.77	21.45	16.82
G channel	pure	30.51	31.13	20.50	19.69
	5 BPP	30.70	31.28	20.49	19.71
	10 BPP	30.94	31.26	20.51	19.97
B channel	pure	32.59	31.34	19.50	20.68
	5 BPP	31.99	31.55	19.44	20.19
	10 BPP	32.32	31.87	19.86	20.43

E. Steganalysis

We have determined so far that our scheme is secure. However, we need to conduct steganalysis, the art and science of detecting hidden messages using steganography. While there are a number of steganalysis algorithms developed, we employ the RS steganalytic scheme [7] since this algorithm is well-known, having been adopted for most steganalysis attacks.

In the RS detection method, the relative number of regular groups for masks M=[0 1 1 0] and $-M$=[0 -1 -1 0] is denoted as (R_M, R_{-M}), respectively. Similarly, the relative number of singular groups for masks M and $-M$ is denoted as (S_M, S_{-M}), respectively. If the magnitude of R_M is equal to that of R_{-M}, or the same equivalence happens to the singular group (S_M, S_{-M}), the embedded image will pass the RS steganalysis. Otherwise, it would reveal the presence of the secret message.

In detection processing, we compute the absolute difference of the regular group $|R_M - R_{-M}|$ and that of the singular group $|S_M - S_{-M}|$ with masks M=[0 1 1 0] and $-M$=[0 -1 -1 0]. A small difference usually means that the stego image is more secure to the RS steganalysis.

Table X shows the detection results for the source texture "rope net" in three primary color channels. Even though each patch is concealed with different amounts of secret bits, the difference between regular group R_M and R_{-M} is similar to the values shown for the pure synthetic texture. This similar difference invariance reflects for singular groups S_M and S_{-M}. These statistics indicate our message-oriented texture synthesis

algorithm can resist the RS detection attack. This is due to the fact that the secret bits are concealed by selecting different ranks of candidate patches rather than by modifying their contents.

V. Conclusions And Future Work

This paper proposes a reversible steganographic algorithm using texture synthesis. Given an original source texture, our scheme can produce a large stego synthetic texture concealing secret messages. To the best of our knowledge, we are the first that can exquisitely weave the steganography into a conventional patch-based texture synthesis. Our method is novel and provides reversibility to retrieve the original source texture from the stego synthetic textures, making possible a second round of texture synthesis if needed. With the two techniques we have introduced, our algorithm can produce visually plausible stego synthetic textures even if the secret messages consisting of bit "0" or "1" have an uneven appearance of probabilities. The presented algorithm is secure and robust against an RS steganalysis attack. We believe our proposed scheme offers substantial benefits and provides an opportunity to extend steganographic applications.

One possible future study is to expand our scheme to support other kinds of texture synthesis approaches to improve the image quality of the synthetic textures. Another possible study would be to combine other steganography approaches to increase the embedding capacities.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. We would like to express our special thanks of gratitude to Dr. Evelyne Pickett for her enthusiastic assistance in improving the clarity of this article.

References

[1] N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," *Computer*, vol. 31, no. 2, pp. 26-34, 1998.
[2] N. Provos and P. Honeyman, "Hide and seek: an introduction to steganography," *Security & Privacy, IEEE*, vol. 1, no. 3, pp. 32-44, 2003.
[3] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding-a survey," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1062-1078, 1999.
[4] Y.-M. Cheng and C.-M. Wang, "A high-capacity steganographic approach for 3D polygonal meshes," *The Visual Computer*, vol. 22, no. 9, pp. 845-855, 2006.
[5] S.-C. Liu and W.-H. Tsai, "Line-based cubism-like image—A new type of art image and its application to lossless data hiding," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 5, pp. 1448-1458, 2012.
[6] I.-C. Dragoi and D. Coltuc, "Local-prediction-based difference expansion reversible watermarking," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1779-1790, 2014.
[7] J. Fridrich, M. Goljan, and R. Du, "Detecting LSB steganography in color, and gray-scale images," *MultiMedia, IEEE*, vol. 8, no. 4, pp. 22-28, 2001.
[8] Y. Guo, G. Zhao, Z. Zhou, and M. Pietikäinen, "Video texture synthesis with multi-frame LBP-TOP and diffeomorphic growth model," *IEEE Trans. Image Process.*, vol. 22, no. 10, pp. 3879-3891, 2013.
[9] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000, pp. 479-488.
[10] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. of the Seventh IEEE International Conference on Computer Vision*, 1999, pp. 1033-1038.
[11] C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun, "Multiscale texture synthesis," ACM *Trans. Graph.*, vol. 27, no. 3, pp. 1-8, 2008.
[12] H. Otori and S. Kuriyama, "Data-embeddable texture synthesis," in *Proc. of the 8th International Symposium on Smart Graphics*, Kyoto, Japan, 2007, pp. 146-157.
[13] H. Otori and S. Kuriyama, "Texture synthesis for mobile data communications," *IEEE Comput. Graph. Appl.*, vol. 29, no. 6, pp. 74-81, 2009.
[14] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen, "Wang Tiles for image and texture generation," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 287-294, 2003.
[15] K. Xu, D. Cohen-Or, T. Ju, L. Liu, H. Zhang, S. Zhou, and Y. Xiong, "Feature-aligned shape texturing," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1-7, 2009.
[16] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Trans. Graph.*, vol. 20, no. 3, pp. 127-150, 2001.
[17] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001, pp. 341-346.
[18] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354-362, 2006.
[19] X. Li, B. Li, B. Yang, and T. Zeng, "General framework to histogram-shifting-based reversible data hiding," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2181-2191, 2013.
[20] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59-66, 1988.
[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600-612, 2004.

Kuo-Chen Wu is a Ph.D. student in the Department of Computer Science and Engineering at National Chung Hsing University, Taiwan. He received the M.Sc. degree from the Department of Computer Science and Engineering, National Chung Hsing University, Taiwan in 2006. His current research interests include computer graphics, color science, image processing, watermarking, and steganography.

Chung-Ming Wang (M'96) received the B.S. degree in Applied Mathematics from the National Chung Hsing University, Taiwan, in 1984, and the Ph.D. degree in Computer Science from the University of Leeds, Leeds, U.K., in 1992. From August 1986 to August 1988, he was a system analyst in the President Enterprise Cooperation and the PAL Company. He is currently a professor in the Department of Computer Science and Engineering, National Chung Hsing University, Taiwan. His research interests include computer graphics, color science, virtual reality, multimedia systems, watermarking, and steganography. Dr. Wang has won three Dragon Thesis Awards, funded by the Acer Computers, and the two Outstanding Paper Awards from the Computer Society of the Republic of China. He is a member of ACM and IEEE Computer Society.