Abstract—An approach for steganography using reversible texture synthesis, where instead of using the existing cover image, the algorithm synthesizes a new texture image using source texture and embeds messages by the process of texture synthesis. The size of new image can be user specified. We can extract the embedded secret message and original source texture image from the final stego synthetic texture. This technique of steganography using reversible texture synthesis provide some advantages like the embedding capacity, which is directly proportional to the size of stego image. Also it provides the ability to recover the source texture from stego texture image.

Keywords—steganography; texture synthesis; data embedding; dct:dwt

I. INTRODUCTION

Significance of steganography is due to the exponential growth and secret communication of potential PC users over the internet. Steganography is a singular strategy of information hiding. A typical application of steganography includes covert communications between two parties, whose presence is unknown to an attacker and whose success depends on detecting the presence of this communication. Usually steganography is achieved in text, image, audio, video or multimedia content for military communication, copyright, authentication and many other purposes.

Image steganography utilizes redundancy of digital image to embed the secret data. Several strategies have been proposed for image based steganography. LSB steganography is the simplest one. Most image steganographic algorithms adopt a current image as cover medium. Secret messages embedded into this cover image are the image distortion experienced in the stego image, which leads to two drawbacks.

1. Cover image size is fixed, so hiding more secret messages result in more image distortion.
   Hence a compromise must be reached between the image quality and embedding capacity.
2. Image steganalytic algorithm can reveal the hidden message, when stego image contains some distortion.

II. RELATED WORK

Texture synthesis has received a lot of attention in computer vision and computer graphics. A source texture image can be re-sampled by either pixel-based or patch-based algorithms to create a synthesized texture.

Pixel-based algorithms [1] 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. Otori and Kuriyama [2] 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.

Patch-based algorithms [3] paste patches from a source texture instead of a pixel to synthesize textures. This improves the quality of pixel-based synthetic textures, since the texture structures inside the patches are maintained. But patches are pasted with a small overlapped region, so one needs to make an effort to ensure that the patches agree with their neighbors. A patch-based sampling strategy [4] was introduced, which use feathering approach for the overlapped areas of adjacent patches.

Data hiding techniques mainly falls into two categories, spatial domain and frequency domain methods. For spatial domain method, the secret messages will be embedded directly in the pixels of cover image. In case of frequency domain, the cover image will first be changed from spatial domain...
to frequency domain before embedding the secret messages. Some of the recent approaches for steganography in both spatial and frequency domain include.

**2.1. Line-based Cubism-like image in data hiding**

Aesthetic data hiding is a new method of data hiding by the utilization of art image created by some art image generation algorithm. An art image created from source image based on cubism properties is called line-based cubism like art image [5].

The line-based Cubism-like image generation process includes two major stages. They are prominent line extraction and region re-coloring. In the first stage, apply edge detection and Hough transform in the source image in order to extract the line segments. Then perform short line segment filtering and nearby line merging. In the second stage, regions in the image are created by extending the line segments to the image boundary to divide the image space. Then the regions are re-colored by the average region colors and the boundaries of the regions are whitened. The final image is the desired line based cubism like image.

There are two stages in data hiding in line-based Cubism-like images. They are data string randomization and data embedding.

Steps in embedding a data string into a Cubism-like image include:

1. Randomize and segment the data string, M by the following steps.
   1.1. Transform M into binary string, and randomize the order of its bits using random number generator function f1, with secret key Ks as the seed.
   1.2. Append an ending pattern into the binary string to keep its length a multiple of three.
   1.3. Divide it into a sequence of 3-digit segments.

2. Generate an art image and compute the areas and average colors of each region. Then randomize the order of the regions using function f2, and take the resulting sequence for data hiding.

3. Calculate the maximum data embedding capacity, Qi. Take an unprocessed region and compute Qi by the following steps with the initial value of Qi set to be zero.
   3.1. Let (Nr0, Nr1), (Ng0, Ng1), and (Nb0, Nb1) denote the numbers of 0s and 1s embeddable in R, G, and B color channels respectively.
   3.2. Take an 3-digit segment, drgdgb of message and compute (Nr0, Nr1), (Ng0, Ng1), and (Nb0, Nb1) for dr, dg and db respectively, in the following way
      i. if \( d_r = 0 \), increment \( N_{r0} \) by 1; else, increment \( N_{r1} \) by 1;
      ii. if \( d_g = 0 \), increment \( N_{g0} \) by 1; else, increment \( N_{g1} \) by 1;
      iii. if \( d_b = 0 \), increment \( N_{b0} \) by 1; else, increment \( N_{b1} \) by 1;
   3.3. If all of the following three inequalities hold:
      \[-A'/2 \leq Nr1-Nr0 < A'/2;\]
      \[-A'/2 \leq Ng1-Ng0 < A'/2;\]
      \[-A'/2 \leq Nb1-Nb0 < A'/2;\]
      Then increase \( Q_i \) by 3; else, \( Q_i \) have reached maximum data embedding capacity for the region. Here \( A' \) is the area of the region.
   3.4. If the message is not exhausted, then go to Step 3.2.

4. Perform the following steps to embed data
   4.1. Randomize the order of pixels in the region using function f3
   4.2. Embed an unembedded segment, drgdgb of message into an unprocessed pixel with color values (Cr, Cg, Cb) by the following steps.
      i. Obtain new color values by modifying the original ones in the following way for \( h = r, g \) and \( b \):
         Increment \( C_h \) by 1 if \( d_h = 1 \);
Decrement $C_h$ by 1 if $d_h = 0$;  
Do nothing to $C_h$ if $d_h = d$ (an ending pattern digit).  
ii. Re-color the pixel by the new color values and decrement $Q_i$ by 3.  
iii. If $Q_i$ is not equal to zero, then go to Step 4.2.

5. Repeat Steps 3 and 4 if message is not exhausted.

6. Re-color each region, which has not been used for data embedding, by the following steps.  
6.1. Create a random digit string $B$ using function $f_4$, and $B$ is called a camouflage string.  
6.2. Perform Steps 3 through 5 to re-color the pixels to embed camouflage string $B$.

7. Take the final image as the desired stego-image.

The data extraction process is exactly a reverse version of the proposed data hiding process. This approach with cubism-like image distract the hacker’s attention to the message data embedded in them. Also by using the three measures of randomization of the input message data and the processing order of them with a secret key enhances the security of this approach.

2.2. Steganography using 32X32 Interpolated Quantization Table and DCT  
In this approach, a frequency based image steganography using Discrete Cosine Transform (DCT) and JPEG is used [6]. Instead of utilizing $8 \times 8$ pixel blocks with the $8 \times 8$ pixel quantization table, a larger block of size $32 \times 32$ is used with a corresponding $32 \times 32$ quantization table generated by cubic interpolation technique. While maintaining the image quality and the size of JPEG stego-image, this method reduces the computation time and increases the capacity of the secret messages.

Here a larger block size is used to decrease the computational time. But when a larger block size is used there is also a constraint in computer memory space required more for computation during the process. Therefore, blocks of size $32 \times 32$ are used to make JPEG image steganography. By utilizing $32 \times 32$ pixel blocks, a corresponding $32 \times 32$ pixel quantization table is required in the quantization. But, there is no standard quantization table for this size. Therefore $32 \times 32$ pixel quantization table is made by utilizing interpolation technique.

2.2.1. Interpolated Quantization Table  
In order to create $32 \times 32$ pixel blocks, the standard $8 \times 8$ pixel quantization table is first chosen to be the base quantization table (Table 1). Then, cubic spline interpolation method is applied to create $16 \times 16$ pixel interpolated quantization table by estimating the unknown data between two adjacent pixels. Then, $32 \times 32$ pixel interpolated quantization table is acquired by utilizing the $16 \times 16$ pixel interpolated quantization table as the base quantization table. Finally, the $32 \times 32$ pixel table is modified using Chang’s technique [7], so that the values in the center part of the table will be changed to 1.

<table>
<thead>
<tr>
<th>Table 1. Standard Quantization Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>49</td>
</tr>
<tr>
<td>72</td>
</tr>
</tbody>
</table>
2.2.2. Embedding Algorithm
1) Partition cover image into non-overlapping 32×32 pixel blocks.
2) Apply DCT transformation to each block, starting from left to right and top to bottom.
3) Quantize each DCT coefficient block by partitioning each block with 32 × 32 pixel quantization table and then rounding off.
4) Embed the secret messages into each quantized DCT coefficient block. The secret messages are embedded into the two least significant bits of each quantized coefficient in the middle part of the quantization table that has value ‘1’ (in the middle frequency part) beginning from left to right and top to bottom.
5) Employ JPEG encoding algorithms. In this step, the stego-image JPEG is obtained.

At first, a cover image is transformed into frequency domain by utilizing DCT and then secret messages are embedded after the quantization process. This model can reduce the computation time and increase the capacity while maintaining the size and image quality.

2.3. Steganography based on Huffman Encoding and DWT.
This approach of image steganography is based on DWT, where DWT is used to transform the original image (cover image) from spatial to frequency domain [8]. Initially, on a gray level cover image of size M × N, two dimensional Discrete Wavelet Transform (2-D DWT) is performed, and on the secret messages/image huffman encoding is performed before embedding. Then embed each bit of Huffman code of secret message/image in the high frequency coefficients resulted from Discrete Wavelet Transform. By preserving the wavelet coefficients in the low frequency sub band, the image quality is improved.

2.3.1. Embedding Secret Message / Image
The embedding scheme of secret message/image includes five steps. They are:
1. Using Haar wavelet transform, decompose the cover image.
2. Huffman encoding: Perform Huffman encoding on the secret message or image to convert it into a bits stream.
3. 3-bit block preparation: Huffman code is divided into 3-bits blocks and thus form a decimal value ranging from 0 to 7.
4. Bits replacement: One sub-band is selected for embedding the secret message. The 3 LSB positions of wavelet coefficients are replaced by the 3 bits of Huffman encoded bit.
5. Apply the Haar inverse DWT (IDWT) on the DWT transformed image, including the modified sub band to generate a new image which contains secret message.

2.3.2. Extraction of the Secret Message / Image
The stego-image is received in spatial domain. In order to transform the stego-image from spatial domain to frequency domain DWT is applied. The extracting algorithm is given as follows.
Extraction Algorithm:
1. Apply DWT to the stego-image.
2. Extract the size of encoded bit stream by collecting the 3 least significant bits from 1st four DWT coefficients in each subband.
3. Collect the 3 least significant bits of all of the DWT coefficients inside each sub bands.
4. Step 3 is repeated until the size of the array becomes equivalent to the size extracted in step 2. 
5. The Huffman table is constructed by extracting 3 bits from the LSB of all of the DWT coefficients inside every sub-band excluding the coefficients utilized in step 2 and step 3.
5. Using the Huffman table obtained in step 5, the array obtained in step 3 is decoded.
6. End.
III. PERFORMANCE ANALYSIS

For performance analysis, different methods of steganography are analyzed. Spatial domain and frequency domain methods are the two different types of steganographic methods. The spatial domain technique include line-based cubism like image in data hiding, which is reversible. That is the cover image can be losslessly recovered from the stego image. While the frequency domain techniques include steganography using 32×32 interpolated QT (QuantizationTable) and DCT, and steganography based on Huffman encoding and DWT, and they does not have the reversible capability. That is, they cannot recover the original cover image. The PSNR value for different steganographic techniques are given in Table 2. The embedding capacity is given in bits per pixel (bpp). In line-based cubism-like image in data hiding, the embedding capacity is proportional to ‘a’, that is the color shifting. The embedding capacity is 1bpp, when a=1. The embedding capacity is doubled when a=2 and so on. There is no obvious degradation in the image quality as ‘a’ increases from 1 to 8.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Domain</th>
<th>Reversible</th>
<th>Embedding Capacity (512 × 512) image</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line-based Cubism-like image</td>
<td>Spatial</td>
<td>Possible</td>
<td>∞ a</td>
<td>Above 50dB</td>
</tr>
<tr>
<td>Using 32 × 32 QT and DCT</td>
<td>Frequency</td>
<td>Not Possible</td>
<td>0.95 bpp</td>
<td>Above 36dB</td>
</tr>
<tr>
<td>Using Huffman and DWT</td>
<td>Frequency</td>
<td>Not Possible</td>
<td>0.98 bpp</td>
<td>Above 54dB</td>
</tr>
</tbody>
</table>

IV. STEGNOGRAPHY USING REVERSIBLE TEXTURE SYNTHESIS

The basic unit employed in steganographic texture synthesis is referred to as “patch”. An image block of source texture is represented as patch and its size is user-defined. A patch contains the central part and an outer part, where the central part is known as the kernel region, and the part surrounding the kernel region is referred to as the boundary region. A source texture is divided into a number of non-overlapped kernel blocks and each kernel block is expanded with a depth at each side to produce a source patch. The expanding process will overlap its neighboring blocks and boundary mirroring is done if a kernel block is found around the boundary of a source texture.

When synthesizing the synthetic texture, steganographic texture synthesis algorithm [9] needs to generate candidate patches. Candidate patches are produced by utilizing a mask of source patch size and then scan the source texture by shifting a pixel each time along the scan line order. There are two procedures in steganography using reversible texture synthesis.

• Message Embedding Procedure
• Message Extracting Procedure

4.1 Message embedding procedure

The three processes of message embedding procedure are index table generation, patch composition process and message-oriented texture synthesis. The flowchart of the message embedding procedure is shown in Figure 1.

4.1.1. Index Table Generation

The first process in message embedding procedure is the index table generation, where an index table is produced to record the location of corresponding source patch set in the synthetic texture. The
index table is used to access the synthetic texture and retrieve the source texture completely. In initial condition, the index table is blank, then each entry will be having a value of −1 and the table is reassigned with values, when source patch ID’s are distributed in the synthetic texture. A random seed for patch ID distribution can be used, which increases the security of steganographic algorithms making it more difficult for malicious attackers to extract the source texture.

4.1.2. Patch Composition Process
The second process of message embedding procedure is the patch composition process, which paste the source patches into the workbench to produce a composition image. First, set a blank image as workbench where the size of workbench is equal to the size of synthetic texture. Then paste the source patches into the workbench by referring to the source patch ID’s stored in the index table. If no overlapping of the source patches is encountered during the pasting process, then paste the source patches directly into the workbench. If pasting locations cause the source patches to overlap each other, image quilting technique is employed to reduce the visual artifact on the overlapped area.

4.1.3. Message-Oriented Texture Synthesis Process
An index table and a composition image are generated, and then the source patches are pasted directly into the workbench. The secret message is embedded via message oriented texture synthesis, which will produce the stego synthetic texture.

4.2. Message extracting procedure
At the receiver side, the message extracting includes generating the index table, retrieving the source texture, performing the texture synthesis, and extracting and authenticating the secret message hidden in the stego synthetic texture. The flowchart of the message extracting procedure is shown in Figure 2.

The receiver has the secret key, so the same index table as the embedding procedure can be produced. The next step is to recover the source texture. By referring the index table, each kernel
region and its relating order with respect to the size of source texture can be recovered. Then the kernel blocks are arranged based on their order, thus retrieving the recovered source texture which will be same as the source texture. The third step is the composition image generation. By referring to the index table, the source patches are pasted into the workbench to deliver a composition image. The composition image generated is identical to the one created in the embedding procedure.

The message extraction and authentication is the final step, which includes three sub steps.

3. By referring to the present working location, develop a candidate list based on the overlapped area. It’s similar to the embedding procedure, creating the same number of candidate lists and their corresponding ranks.

4. Match-authentication step, given the present working location on the workbench, refer to the relating stego synthetic texture to determine the stego kernel region.

5. Determine if there is a patch in the candidate list by searching it, where the kernel region of the patch is same as the stego kernel region.

If the patch is available, it is referred as the matched patch. The rank of the matched patch is located, and this rank represents the decimal value of the secret bits conveyed in the stego patch, when texture synthesis is operated in the message embedding procedure. However, if there is no matched patch in the candidate list where the kernel region is the same as the stego kernel region, it implies that the stego kernel region has been altered with, leading to a failure of the message authentication. In this way, all of the secret messages that are hidden in the stego synthetic texture are authenticated and extracted patch by patch.

V. CONCLUSION

Different approaches for image steganography in both spatial and frequency domain methods are described. Steganography using reversible texture synthesis is an effective method for data hiding. It utilizes a small texture as the input image. Given an input source texture, the scheme can produce a large stego synthetic texture depending on the size of the secret messages. Also the reversible capability allows recovery of the source texture, which can be used for the second round of texture synthesis if needed. The algorithm is secure and robust against an RS steganalysis attack.

REFERENCES


