IMAGE WATERMARKING USING REVERSIBLE TEXTURE SYNTHESIS

Mahalakshmi B[1], Shobana Bharathi A[2], Jayashankari J[3], Veeralakshmi P[4]

[1] *Student, IT, Prince Shri Venkateshwara Padmavathy Engineering College, TN, India*

[2] *Student, IT, Prince Shri Venkateshwara Padmavathy Engineering College, TN, India*

[3] *Assisstant Professor, IT, Prince Shri Venkateshwara Padmavathy Engineering College, TN, India*

[4] *Associate Professor, IT, Prince Shri Venkateshwara Padmavathy Engineering College, TN, India*

ABSTRACT

Steganography is a process of information hiding technique in host medium. The host medium can be of any host digital media. Thus, steganography provides communication between two parties whose existence is unknown to an attacker. Most image steganographic algorithms adopt an existing image as a cover medium to hide the message and the size of the cover image is fixed, so the more secret messages allow for more image distortion. In this paper, we propose a novel approach for steganography using a reversible texture synthesis. Texture synthesis is the process of sampling the original texture image. A patch represents a piece of source texture in which the secret message can be embedded. Then the patches along with the message are reformed into a single image. Thus, the message is passed in a more secure way and this process is highly efficient and robust against steganalytical attacks.

Keywords :- *Data embedding, reversible steganography, texture synthesis.*

1. 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 is 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. 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. In general, the host medium used in steganography includes meaningful digital media such as digital image, text, audio, video, 3D model, etc. A large number of image steganographic algorithms have been investigated with the increasing popularity and use of digital images. 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 secret messages for steganography if needed.

2. RELATED WORK

Texture synthesis has received a lot of attention recently in computer vision and computer graphics. 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 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 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, 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.* introduced the patch-based sampling strategy and used the feathering approach for the overlapped areas of adjacent patches. Efros and Freeman 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.

Ni *et al.* 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.*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.

3. PROPOSED SYSTEM

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. We can denote the size of a patch by its width (*Pw)* and height (*Ph)*.Given a source texture with the size of *Sw* × *Sh* we can subdivide the source texture into a number of patches. As an example, given a source texture with the size of *Sw*×*Sh* = 128×128, if we set the size *Pw* × *Ph as* 32 × 32, then we can generate *KB* = 16 kernel blocks (patches). Each element in *KB* can be identified as {*kb*0, *kb*1. . . *kb*15}.

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 *Pw*×*Ph* and then travel the source texture (*Sw*×*Sh)* by shifting a pixel each time following the scan-line order. Let *CP* = {*cpi* | *i* = 0, 1, . . . , *CPn* −1} represent the set of the candidate patches where *CPn* = *_CP_* denotes the number of elements in *CP*. 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.

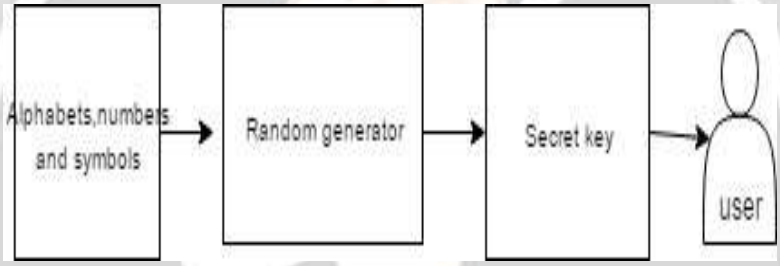

Fig -1 User's secret key generation.

3.1 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.

- **User's Secret Key Generation:**

  A unique secret key has been generated for each user. The generation of the secret key is shown in Fig. 1.It is generated using the random generator with characters, numbers and symbols as input.This secret key is used by the receiver for decrypting the extracted secret message.

- **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. The index table allows us to access the synthetic texture and retrieve the source texture completely. We first determine the dimensions of the index table *(Tpw* × *Tph)*. Given the parameters *Tw* and *Th*, 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.

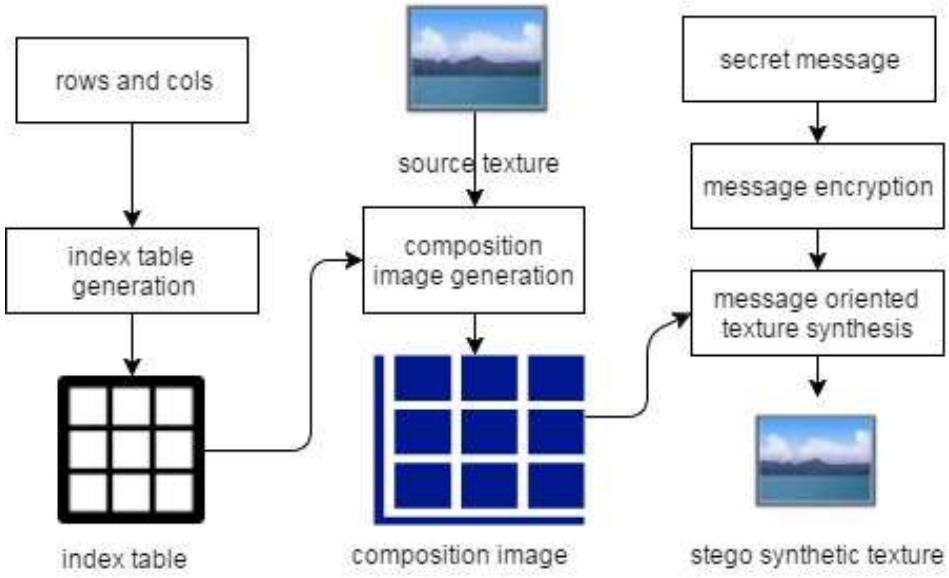

Fig -2 Embedding the secret message in image.

- **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.

- **Message encryption and 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. The entire process of embedding the message in the image is shown in Fig. 2.

Algorithm 1:

Begin
Input: Image, Secret_Message;
Convert Secret_Message to Binary_Codes;
Set BitsPerUnit to Zero;
Encode Message to Binary_Codes;
Add by 2 unit for bitsPerUnit;
Output: Stego_Image;
End

Fig -3 Algorithm for embedding data inside image.

The algorithm for embedding the message in an image is shown in Fig. 3. The secret message is encrypted using the encryption algorithm, before it is being embedded in the image.

3.2 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 (*EPn).* Each patch can conceal at least one bit of the secret message; thus, the lower bound of BPP will be 1, and the maximal capacity in bits that can be concealed at each patch is the upper bound of BPP, as denoted by *BPPmax* . In contrast, if we can select any rank from the candidate list, the upper bound of BPP will be log2 *(CPn).* The total capacity (*TC*) our algorithm can offer is the multiplciation of BPP and *EPn*. The number of the embeddable patches is the difference between the number of patches in the synthetic texture (*TPn)* and the number of source patches subdivided in the source texture (*SPn).*

3.3 Source Texture Recovery, Message Extraction And Decryption 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 which is shown in Fig. 5. The algorithm for extracting the message from the stego image is explained in Fig. 4.

Algorithm 2:

Begin
Input: Stego_Image;
Calculate BitsPerUnit;
Decode All_Binary_Codes;
Shift by 2 unit for bitsPerUnit;
Convert Binary_Codes to Secret_Message;
Output Secret_Message;
End

Fig -4 Algorithm for extracting data from stego image.

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. 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 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. In this way, we can authenticate and extract all of the secret messages that are concealed in the stego synthetic texture patch by patch. The extracted message is then decrypted using the decryption algorithm. But, the decryption process is carried out only if the receiver enters the correct secret key. Finally the secret message is obtained.

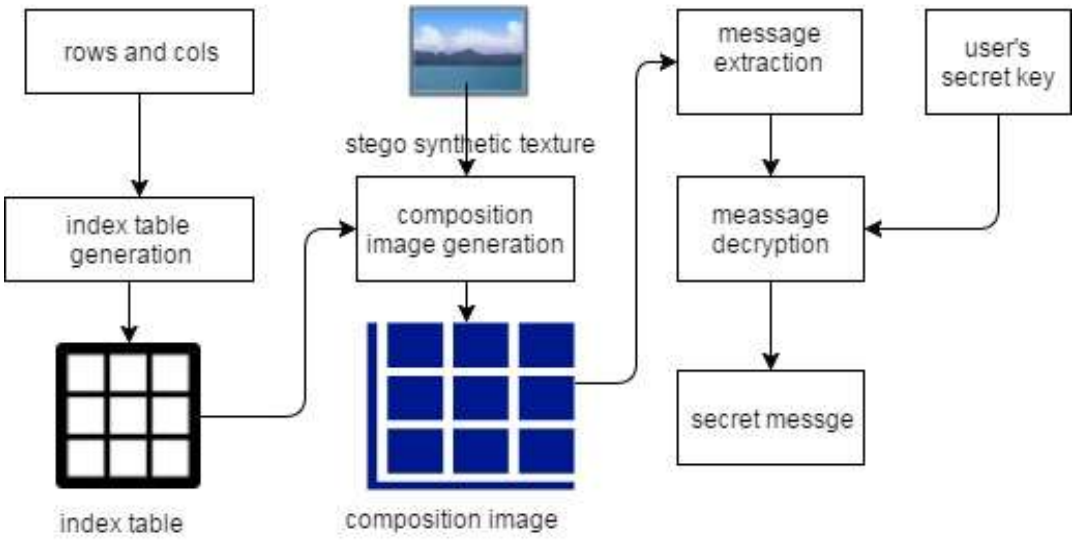

Fig -5 Message extraction process.

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1 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 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.

4.2 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 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 = [0110] and $-M$ = [0 − 1 − 10] is denoted as (RM, $R{-}M$), respectively. Similarly, the relative number of singular groups for masks M and $-M$ is denoted as (SM, $S{-}M$), respectively. If the magnitude of RM is equal to that of $R{-}M$, or the same equivalence happens to the singular group (SM, $S{-}M$), the embedded image will pass the RS steganalysis. Otherwise, it would reveal the presence of the secret message.

5. CONCLUSION

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.

6. 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," *IEEE Security Privacy*, vol. 1, no. 3, pp. 32–44, May/Jun. 2003.

[3] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062–1078, Jul. 1999.

[4] Y.-M. Cheng and C.-M. Wang, "A high-capacity steganographic approach for 3D polygonal meshes," *Vis. Comput.*, vol. 22, nos. 9–11, 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, Oct. 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, Apr. 2014.

[7] J. Fridrich, M. Goljan, and R. Du, "Detecting LSB steganography in color, and gray-scale images," *IEEE MultiMedia*, vol. 8, no. 4, pp. 22–28, Oct./Dec. 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, Oct. 2013.

[9] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn.*, 2000, pp. 479–488.

[10] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, Sep. 1999, pp. 1033–1038.

[11] C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun, "Multiscale texture synthesis," *ACM Trans. Graph.*, vol. 27, no. 3, 2008, Art. ID 51.

[12] H. Otori and S. Kuriyama, "Data-embeddable texture synthesis," in *Proc. 8th Int. Symp. Smart Graph.*, 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, Nov./Dec. 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 *et al.*, "Feature-aligned shape texturing," *ACM Trans. Graph.*, vol. 28, no. 5, 2009, Art. ID 108.

[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. 28th Annu. Conf. Comput. Graph. Interact. Techn.*, 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, Mar. 2006.

[19] X. Li, B. Li, B. Yang, and T. Zeng, "General framework to histogramshifting- based reversible data hiding," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2181–2191, Jun. 2013.

[20] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *Amer. Statist.*, 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, Apr. 2004.