International Journal of Computer Engineering in Research Trends
Multidisciplinary, Open Access, Peer-Reviewed and fully refereed
Research Paper Volume-5, Issue-7, 2018 Regular Edition E-ISSN: 2349-7084

Secret Data Embedding using Texture Synthesis

Suphiya P. Inamdar, Suhas B. Bhagate

[1,2]*Dept. of Computer Science and Engineering, D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, India.*

e-mail: suphiyainamdar44@gmail.com, suhas.bhagate@gmail.com

Available online at: http://www.ijcert.org

Received: 11/July/2018, Revised: 12/July/2018, Accepted: 14/July/2018, Published: 24/July/2018

Abstract:- A steganography is an art of hiding confidential data into digital media such as image, audio, video etc. The proposed the system using steganography using reversible texture synthesis. Texture synthesis uses the concept of the patch which represents an image block of source texture where its size is user specified. A texture synthesis process resamples a smaller texture image and provides a new image with arbitrary size and shape. Instead of using an existing cover image to hide messages, the algorithm conceals the source texture image and embeds secret messages using the process of texture synthesis. This allows extracting the hidden messages and source texture from a stego synthetic texture. The approach offers some advantages. First, the scheme provides the embedding capacity that is proportional to the size of the stego texture image. Second, the reversible capability inherited from this scheme includes functionality, which allows recovery of the source texture. And third, there will be no image distortion since the size of the new texture image is user-specified..

Keywords: Steganography, Data embedding, Texture synthesis, Cover medium, Index table.

1. Introduction

Texture synthesis is the process of re-samples a small texture image drawn by an artist or captured in a photograph to synthesize a new texture image, which has a similar local appearance and arbitrary size. This project aims to combine the texture synthesis process into steganography [1] to conceal secret messages as well as the source texture. The hidden messages and the source texture can be extracted from a stego synthetic texture.

The texture is something which is composed of repeated patterns and it exactly looks like a consistent image. Texture synthesis is the process of creating repeated patterns of compositions from a smaller texture image known as the source texture by taking advantage of its organic content. Texture synthesis finds its application in a wide variety of areas like graphics, image enhancement techniques, animation etc. Nowadays stenographic data hiding [2] schemes are also being used together with texture synthesis methods. By the addition of coding techniques and cryptographic techniques along with the methods mentioned above, the data hiding capacity and security can be improved.

In texture synthesis process pixel-based algorithms generate the synthesised image pixel by pixel and use spatial neighbourhood comparisons to choose the most similar pixel in a sample texture as the output pixel. This output pixel is determined by the already synthesised pixels, any wrongly integrated pixels during the process influence the rest of the result causing propagation of errors. The steganography and reversible texture synthesis are based on patch-based algorithms which paste patches from a source texture instead of a pixel to synthesise textures. This method improves the image quality of pixel-based synthetic surfaces because texture structures inside the pieces are maintained. A patch represents an image block of a source texture where its size is user-specified. This algorithm conceals the source texture image and embeds secret messages through the process of texture synthesis. This is to extract the secret messages and source texture from a stego synthetic texture.

The remainder of this paper is organized as follows: in Section II, contain the related work of Texture Synthesis. 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.

 https://doi.org/10.22362/ijcert/2018/v5/i7/v5i703

2. Related Work

A simple and efficient pixel-based method is performed by Efros and Leung [3] in 1999. This method generates the synthesised image pixel by pixel and uses spatial neighbourhood comparisons to choose the most similar pixel in a sample texture as the output pixel. In this, a single pixel is generated at a time from an initial seed. A fixed size window with user-specified size is taken which is cantered on the currently synthesising pixel. The more matching pixel is searched and copied in the output target. This process is repeated until all pixels in the target image are generated. But any wrongly synthesized pixels during the process influence the rest of the result causing propagation of errors.

Wei and Levoy [4] presented a deterministic algorithm, and the output texture is generated in a scan line order. This method improves the speed of the synthesis procedure by using tree-structured vector quantization (TSVQ) to match the target pixel with the neighbourhood pixels. This fastest algorithm has the largest memory requirement. But it failed to synthesize the texture images of flowers, leaves, pebbles etc.
L. Liang, C. Liu [5] presented an algorithm for synthesizing textures from an input sample. This patch-based sampling algorithm is very fast and it creates high-quality texture image. This algorithm works well for wide variety textures like regular to stochastic textures. Can be sampling patches using a nonparametric estimation of the local conditional MRF density function.Also, avoid mismatching features across patch boundaries of an image.

The building blocks of the patch-based sampling algorithm are patches of the input sample texture to construct the synthesized texture. We can carefully select these patches of the input sample texture and paste it into the synthesised surface to avoid mismatching features across patch boundaries. Patch-based sampling algorithm combines the nonparametric sampling and piece pasting strengths.The texture patches in the sampling scheme provide implicit constraints to avoid garbage found in some textures.

A. A. Efros and W. T. Freeman [6] proposed a method for generates a new image by stitching together small patches of existing images. This process is known as image quilting. It is a quick and straightforward texture synthesis algorithm. By extending this algorithm to perform texture transfer operation.

In patch-based texture synthesis procedure, define the square block of user-specified size from the set of all such overlapping blocks in the input texture image. To synthesize a new texture image, let us tile the blocks taken randomly from the input texture image. Next step is to introduce some overlap in the placement of blocks onto the new image. Now, search source texture for such a block that agrees on some measure with its neighbours along the region of overlap. At last, let the blocks have ragged edges which will allow them to better approximate the features in the texture. Before placing the block into the composition can calculate the error in the overlap region between it and the other blocks. Then find a minimum cost path through that error surface and see the boundary of the new block.

Liang, Liu and et al.[7] introduced a fast and new algorithm to generate texture using the method of texture synthesis. The texture generation was based on a non-parametric estimation of Markov Random Field density function. This method assumes the texture as Markov Random model and the stochastic process is assumed to be stationary and local. Markov Random Field model is chosen because of its accuracy in modelling a variety of textures. The method is both applicable to texture synthesis which depends on some constraints as well as which is independent of constraints. Constraint-based texture synthesis is found in applications like texture synthesis for hole-filling and tileable texture synthesis. Here the constraint is a randomness parameter.

Z. Ni, Y.-Q. Shi [8] presented a reversible data hiding algorithm to recover the original image without any distortion from the marked image after the hidden data have been extracted. The zero or the minimum points of the histogram of an image is utilized by this algorithm and slightly modifies the pixel grayscale values for embedding the data into the image. By comparing the existing reversible data hiding algorithms [9], it can embed more data. The algorithm applies to a wide range of models such as commonly used images, medical images, texture images, aerial images and all of the 1096 models in CorelDraw database.

This method can embed a significant amount of data at the same time keeping a very high visual quality for all natural images. Explicitly, the PSNR of the marked image versus the original image is guaranteed to be higher than 48 dB. This technique applies to all types of images. This proposed lossless data hiding technique is used to still pictures and videos.

X. Li, B. Li, B. Yang, and T. Zeng [10], have used a Histogram shifting (HS) technique for reversible data hiding (RDH). Using HS-based RDH method, high capacity and low distortion can be achieved. This paper presents a general framework to construct HS-based RDH technique. Using this proposed framework can get an RDH algorithm by merely designing shifting and embedding functions.

In this method, first divides the host image into no overlapping blocks such that each block contains n pixels. Then, generates an n-dimensional histogram by counting the frequency of the pixel-value-array sized n of each divided block. At last, modifies the resulting n-dimensional histogram for implementing the data embedding scheme.

3. Methodology

The method of steganography using reversible texture synthesis is used primarily for hiding the secret messages. A new texture image is synthesized from several tiny texture images by using the texture synthesis process. The method consists of a combination of both texture synthesis process and steganography.

It contains mainly two procedures [11].

1: Message embedding procedure

2: Message extracting procedure

In the message embedding procedure, the first procedure is dividing the source texture image into different image block. This image block is called as patches. To record the corresponding source patch's location the index table is used. The workbench is a blank image whose size is the same as that of synthetic texture. With the help of source patch ID which is placed in the index table, the corresponding source patches are paste into the workbench to generate a composite image. After pasting the source patch the next step is to find the mean square error (MSE) of the overlapped region. This overlapped area is found in between the patch which we want to insert in the workbench and the synthesis area. The resultant patches are ranked as per the ascending order of mean square error (MSE). And finally the patches are selected from given list in such way that the rank of patches is equalled to a decimal value. The decimal value is nothing but the n-bit value of our secret message.

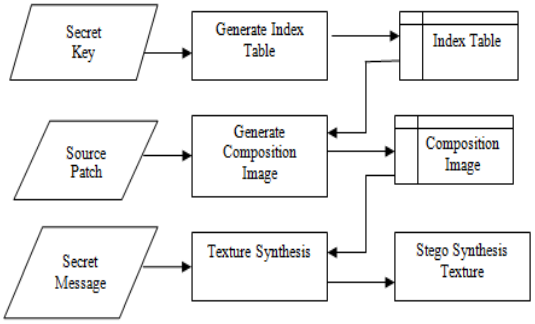

Fig.1 Message embedding procedure

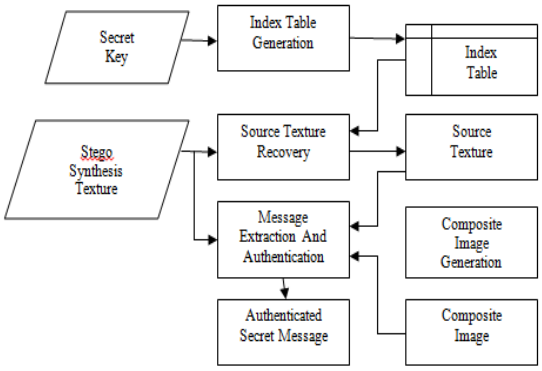

Fig. 2 Message extracting procedure

The proposed system will be designed and implemented in the following modules,

I. Source Patches Generation

II. Index table Creation

III. Message Oriented Texture Synthesis Generation.

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

The above modules work as follows-

In source patches generation module we call the input image as source texture image. This image may be captured in a photograph or drawn by an artist to create synthesized texture image which is having a similar appearance. First, the input image is divided into no. Of patches. Each patch has two areas.

1. Kernal boundary
2. Region boundary

The second process is the index table generation here will create an index table to reserve the location of the source patch set inside the synthetic texture. The index table will allow us to access the synthetic texture and extract the source texture wholly. The composition of any size

according to our wish can be generated using this index table. The next step that has to be used is to attach the source patches into a workbench to create a composition image. First here will set up an empty image as the workbench where the size of the workbench is proportional to the synthetic texture. By referring to the source patch IDs stored in the index table, we then attach the source patches into the workbench. During the attaching process, if no imbrications of the source patches are found, we can attach the source patches directly into the workbench. After the creation of the composition image we have to embed the secret message through the message-oriented texture synthesis to generate the final stego synthetic texture. Source Texture Recovery, Message Extraction, and Message Authentication Procedure the message extracted for the receiver side consist of creating the index table, attaining the source texture, performing the texture synthesis, and extracting and authenticating the secret message hidden inside the stego synthetic texture.

4. Experimental Results and Analysis

A. Results of the Embedding Capacity :

The embedding capacity is one concern of the data embedding scheme. Table I summarizes the equations we described to analyse 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 (CP n). The total capacity (TC) our algorithm can offer is shown in (1) which is the multiplication 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).

$$TC=BPP\times EPn = BPP\times(TPn - SPn) \quad (1)$$

Suppose we provide a source texture Sw×Sh =128×128, and we intend to generate a synthetic texture Tw × Th = 488×488. We specify the patch size Pw × Ph = 48× 48 and the boundary depth Pd = 8 pixels. This will lead to the range of the BPP between 1 and 12. We can produce SPn = 16 source patches and TPn = 144 patches on the stego synthetic texture. Thus, there are EPn =128embeddable patches. If we take the BPP = 12, then the total embedding capacity is TC = 1536 bits.

TABLE 1: THE EMBEDDINGCAPACITYPROVIDED 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.

We collect our experimental results on a personal computer with an i5-8th gen and 4GB memory. We adopt three source textures for the results of our collection. Table II presents the total embedding capacity our algorithm can provide when different solutions 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 Sw×Sh (96×96 vs. 192×192), the smaller the total embedding capacity (TC) our algorithm will offer (1870 bits vs. 1600 bits for 10 BPP). This is because the larger source texture will contain more source patches SPn (9 vs. 36) that we need to paste which cannot conceal any secret bits. This will reduce the number of embeddable patches (EPn) on the composition image (187 vs. 160), thus reducing the total embedding capacity. The maximal capacity provided by our algorithm is 1760 bits.

TABLE 2: TOTAL EMBEDDINGCAPACITY IN BITS OUR ALGORITHMCAN PROVIDE

Texture Synthesis Size: Tw*Th=488*488 Patch Size: Pw*Ph=36*36 Depth: Pd=2					
*Sw*Sh*	*Spn*	*Epn*	*TC(5BPP)*	*TC(10BPP)*	*TC(BPPMAX)*
96*96	9	187	935	1870	2057
128*128	16	180	900	1800	1980
192*192	36	160	800	1600	1760

5. Conclusion and Future Work

This paper proposes a reversible steganography algorithm using texture synthesis. Texture Synthesis method provides reversibility, and it can be used for the second round of texture synthesis if needed. Studies show that the previous techniques have many drawbacks; this can be solved using the new Steganography method using Reversible texture synthesis. Our approach can achieve reversibility, separate data extraction and image recovery.

Acknowledgement

I am grateful to my project guide Suhas B. Bhagat for his remarks, suggestions and for providing all the facilities like providing the Internet access and valuable books, which were essential. I am also thankful to all the staff members of the Department of Computer Science & Engineering of DKTE College of Engineering and Technology, Icchalkaranji.

6. References

[1] N. Provos and P. Honeyman, ―Hide and seek: An introduction to steganography,‖ IEEE Security Privacy, vol. 1, no. 3, pp. 32–44, May/Jun. 1999.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Renjie Chen, Ligang Liu, Guangchang Dong, ― Local resampling for patch-based texture synthesis in vector fields‖, Int. J. of Computer Applications in Technology ,2004.

[8] 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.

[9] H. Otori and S. Kuriyama,―Data-embeddable texture synthesis,‖ in Proc. 8th Int. Symp. Smart Graph., Kyoto, Japan, 2007, pp. 146–157.

[10] 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.

[11] Kuo- Chen Wu and Chung-Ming Wang, "Steganography Using Reversible Texture Synthesis", IEEE Transaction On Image Processing ,vol.24,no.1,pp.2015

