International Journal For Technological Research In Engineering
Volume 3, Issue 9, May-2016 *ISSN (Online): 2347 - 4718*

DATA HIDING USING TEXTURE SYNTHESIS: A REVIEW

Bhajiyawala Denish R
Department of Electronics and Communication,
Hasmukh Goswami College of Engineering, Ahmedabad, India

Abstract: A texture synthesis process resamples a smaller texture image, which synthesizes a new texture image with a similar local appearance and an arbitrary size. The texture synthesis process can be use into steganography to conceal secret messages. In contrast to using an existing cover image to hide messages, some algorithms conceal the source texture image and embed secret messages through the process of texture synthesis. This allows extraction of the secret messages and source texture from a stego synthetic texture.
Keywords: texture synthesis, stegnography, data hiding.

I. INTRODUCTION

A. Texture Synthesis

Texture as a noun it is described in vocabulary as, something composed of closely interwoven elements; specifically a woven cloth. In other words, textures are usually referred to as visual or tactile surfaces composed of repeating patterns, such as a fabric. In the domain of visual information processing, the word texture has a wider meaning. However, since a majority of natural surfaces consist of repeating elements; this narrower definition of texture is still powerful enough to describe many surface properties. Since natural textures may contain interesting variations or imperfections, certain amount of randomness over the repeating patterns should be allowed. For example, a honeycomb texture is composed of hexagonal cells with slight variations of size and shape of each cell. The amount of randomness can vary for different textures, from stochastic (sand beach) to purely deterministic (a tiled floor). Textures can be obtained from a variety of sources such as hand-drawn pictures or scanned photographs. The goal of texture synthesis can be stated as follows: Given a texture sample, synthesize a new texture that, when perceived by a human observer, appears to be generated by the same underlying process. Thus synthesized output texture is perceptually similar to the input texture but also ensures that the result contains sufficient variation. This cannot be achieved by simply tiling the given input texture several times. The blockiness in the result is clearly perceivable. wireless communication. The base-station is has a dedicated power supply and more resources (CPU, memory, storage, etc.) than a node. The node that can recognize the user requested i.e., attributes of interest is called source node. The source nodes transmit data to the sink either directly or by relaying it via other sensor node in WSN. Other than large surface creations, there are many other areas where texture synthesis can be applied as a solution. Like hole-filling, Image-video compression, foreground removal etc.

II. TEXTURE SYNTHESIS METHODS

A. An Optimization-Based Method

This method relies on a global optimization framework to synthesize a new texture.It essentially minimizes an energy function that considers all the pixels together. This function measures the similarity with respect to the example texture and is locally defined for each pixel. The local energy contributions coming from pixels are merged together in a global metric that's minimized.

B. Procedural Methods

These methods synthesize textures as a function of pixel co-ordinates and a set of tuning parameters. Of these methods, the most common in computer graphics is Perlin noise.Perlin noise is a smooth gradient noise function that's invariant with respect to rotation and translation and is band-limited in frequency. You can use it to perturb mathematical functions to create pseudorandom patterns. It has received wide use in various application domains—for example, rendering water waves, fire, or realistic-looking marble or crystal. Another way to efficiently synthesize different classes of textures, such as organic texture patterns, is to simulate a natural process. The process is usually modelled as small interacting geometric elements distributed on the domain. 2D procedural methods generally are efficient and easily extendable to solid-texture synthesis.

C. Statistical Feature-Matching Methods

These methods mainly capture a set of statistical features or abstract characteristics from an exemplar image and transfer them to a synthesized image. An image pyramid can be used to capture statistical properties in the exemplar image at different resolution levels. They initialize the synthesized texture with random noise. Then, they repeatedly apply histogram matching to make each level of the synthesized pyramid converge to the appearance specified by the exemplar image pyramid. This method and its extension work well on stochastic textures, but their quality can degrade if the example texture is structured.

D. Neighbourhood-Matching Methods

These methods mainly enforce and deploy the relation between the pixel colour and its spatial neighbourhood. After an initial training phase that correlates each pixel of the example texture to its neighbourhood kernel, the target image is synthesized pixel by pixel. This step substitutes each pixel with the one in the example texture that has the most similar neighbourhood.

E. Patch-Based Methods

These methods divide the example texture into a set of patches, which they then rearrange in the output image. An overlap region between adjacent patches can be used to appropriately quilt them, making sure they all fit together. This method chooses patches that minimize an overlap error step-by-step randomly from a set of candidates, and iteratively places the patches over the synthesized image. Then, it quilts the overlap region appropriately to minimize the error.

III. WORK IN DATA CODING AND HIDING

Texture images of iterative patterns can be used as a type of image code. Iterative textures sometimes serve as background images in printed material, and you can embed information on some blank places unoccupied by foreground images. Another promising application is the use of a camouflage code on real material. By synthesizing a texture with a sample image scanned from a real material such as wood or cloth, you can embed data by inconspicuously affixing a seal of the printed texture on the material. This camouflage code can record product information for the material while avoiding unsightly coded images.

A. Quick response Code

Denso Wave developed the QR (Quick Response) Code (www.qrcode.com) as an efficient, robust 2D bar code. It has become a popular way to read URLs for navigating Web pages and services, without the troublesome operations of a small numerical keypad. Researchers have developed several methods to introduce an eye-pleasing aesthetic pattern. For example, Color Code(www.colorzip.com) adds colors, and design QR (http://d-qr.net) partially modifies the QR code's patterns to create small icons . These approaches, however, employ the matrix form and impose rigid constraints on the codes' visual design.

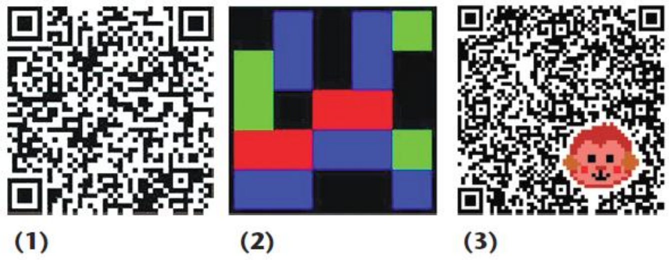


Figure 1 Example 2D bar codes: (1) QR (Quick Response) Code, (2) Color Code, and (3) design QR. These matrix-formed patterns lack flexibility in designing.

Recently, researchers have employed natural images as replacements for QR Code. Some methods intentionally modulate the image's specific color or frequency component according to embedded data. Most methods incorporate one of two numerical techniques: watermarking or steganography. Watermarking has been used to protect copyright, which requires robustness against intentional attack through alteration or destruction of images. Steganography increases the amount of hidden data while sacrificing robustness. These traditional approaches have only a limited amount of embedded data (or payload) for assuring robustness against noise caused by lights, lenses, and imaging devices. Also, increasing such robustness requires a large modulation of the original image's colour or frequency component, which often seriously compromises the resulting image's quality.

B. Expansion Embedding Techniques for Reversible Watermarking

This technique improves the distortion performance at low embedding capacities and mitigates the capacity control problem. Also a reversible data-embedding technique called prediction-error expansion. This new technique better exploits the correlation inherent in the neigh-borhood of a pixel than the difference-expansion scheme. Predic-tion-error expansion and histogram shifting combine to form an effective method for data embedding. The experimental results for many standard test images show that prediction-error expansiondoubles the maximum embedding capacity when compared to difference expansion. There is also a significant improvement in the quality of the watermarked image, especially at moderate embedding capacities. The other category of approaches some times referred to as type-II algorithms in the literature, involves methods to losslessly compress a set of selected features from an image and embed the payload in the space saved due to the compression. This approach results in higher embedding capacity than the type-I approach. They presenta novel reversible data-embedding technique by grouping the pixels and embedding data bits into the state of each group.Celiket al. proposed a generalized LSB (g-LSB) embedding algorithm extending the work in and, thus, were able to introduce several additional points along the capacity-distortion curve. The major drawback of Tian's scheme is the lack of capacity control, which results from having to embed the compressed location map along with the payload. The locations selected for expansion embedding determine the location map, so the compressibility of the location map depends on the set .Since it is impossible to predict the size of the compressed lo-cation map while selecting the locations to embed, it is difficult to determine the capacity before hand. Also, at low embedding rates (i.e., when only a few of the available expandable locationsare selected), the compressibility of the resulting location mapis low, resulting in a large fraction of the selected capacity to beallotted towards embedding the compressed map.

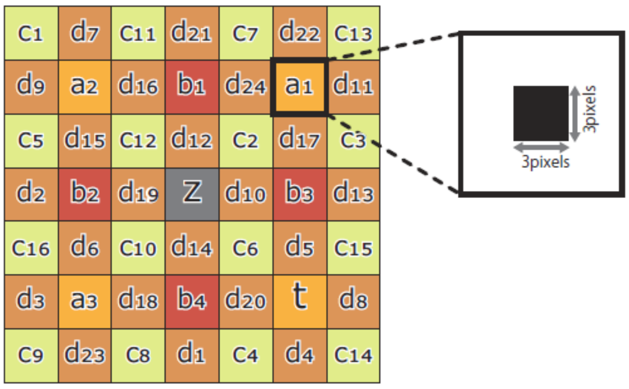
C. Data coding with dotted pattern

Data-coding strategy converts bit-patterns of information into colour, dotted patterns. First colours will be grouped in a sample image with a specific colour component, which is called a coded component. We usually set the component by brightness Y in a Y-Cb-Cr colour space because its variance is usually larger than other components, which is suited to ensure robustness in detecting their differences. We map all colours in a sample image into two groups corresponding to binary data representation. This mapping introduces two criterion colours: one colour, denoted by z, represents a zero point by which the relative displacements of the coded component are computed for all colours. The other criterion colour, denoted by t, provides the threshold τ of the components for giving a discriminator as follows:

$$B(y) = \begin{cases} 0 & \text{if } |y - z| < \tau \\ 1 & \text{else} \end{cases}, \quad \tau = \frac{|t - z|}{2}$$

Where y is the coded component of each colour, and the criterion colours z and t are regarded as parameters to be optimized.

The dotted pattern is then painted using the colours that belong to the corresponding group to embedded binary data, by using the following rule. We regularly subdivide the image region into the blocks of the same size, as shown in Figure, and paint a small square dot at the centre of each block, where the dot area usually occupies 3 × 3 pixels. Here we try to uniformly distribute the dots in random order to avoid undesirable bias in the pattern. Every block is sequentially ordered by its relative centrality, as shown by the number on each block. We first paint dots with two criterion colours; the block marked 'z' for the criterion colour is first chosen at the canter of the image region, and the blocks marked 't' and 'a[1-3]' are secondarily chosen at every center of the four subdivided regions where the subscript number of 'a' denotes the randomly selected sequential order. Thirdly, the blocks marked 'b[1-4]' locatedat the center of four inner boundaries are chosen and randomlyordered. This process is recursively iterated for theset of 'c[1-16]' and 'd[1-24]' by subdividing every region.



the order of dot arrangement.

Notice that the example in Figure 2 represents the arrangement for the dots of n = 7×7 resolution for simplicity, but we actually arrange them with n = 31 × 31. A shorter message is embedded by partially painting the sequence of dots with a terminal symbol.

Every pixel color inside the dots is determined by selecting one from those included in a sample image called an exemplar. The nearby colors to the threshold:

$\tau - T/2 < |y - z| < \tau + T/2$,

should be omitted to ensure robustness, where T represents the width of a barrier for isolating the coded components from the threshold τ, and an increase of T enhances robustness against the noisy variation caused in printing and photographing. However, too much T greatly narrows the range of usable colors, which damages the quality of the synthesized images. We have experimentally found that T = 30 ensures good balance for 8-bitcoded components.

D. Data Hiding in Encrypted color images by Reserving Room before Encryption with LSB Method

Reversible data hiding is the technique in which data in the cover image reversibly can retrieve after the extraction of hidden data in it. The technique provides the secrecy for a data, and also for its cover image. Ancestor methods of reversible data hiding were vacates room for data hiding after encryption, which leads to some errors at the time of data extraction and image recovery. Here describes a novel method of reversible data hiding in which, Reserving room before encryption in images, so that image extraction is subjected to free of errors. Here we are proposing an LSB plane method for the data hiding, which will result more space for embedded secret data. Moreover the usage of colour images as cover images will helps to store more data in different channels. This process is performing with the help of spatial correlation in decrypted image. In another method, at the decoder side by further exploiting the spatial correlation using a different estimation equation and side match technique, which provides much lower error rate. These two methods mentioned above rely on spatial correlation of original image to extract data. Implies that, decryption must be done in the encrypted before data extraction. All the above methods try to vacate room from the encrypted images directly. Because of the entropy of encrypted images has been maximized, these techniques can only obtain small payloads or generate marked image with poor quality for large payload and all of them are subject to some error rates on data extraction and/or image restoration. Some methods can use error correcting codes, also pure payload can consume. In this paper, we proposes a novel method for RDH in encrypted color images, for which we do not ―vacate room after encryption‖, but ―reserve room before encryption‖ We can first empty out spaces by embedding LSBs of some pixels into other pixels with a LSB plane method and then encrypt the image, so the place of these LSBs in the encrypted image can be used to hide data bits. This method separate data extraction from image decryption and data extraction and image recovery are free of any error. Here assigns the interpolation-error. Due to lesser modification of pixels, quality of image will be higher. Reversible Watermarking Algorithm Using Sorting and Prediction is another algorithm without using a location map is used for reversible watermarking. This algorithm employs prediction errors to hide data into an image. To record the prophesy errors based on magnitude of its local variances a sorting technique is used. Using sorted prophesies errors and a reduced size location map allows us to hide more data in the image with less distortion.

E. Steganography Using Reversible Texture Synthesis

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.

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.

This 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.

F. Message Embedding Procedure

shows the three processes of our message embedding procedure. We will detail each process in the following sections.

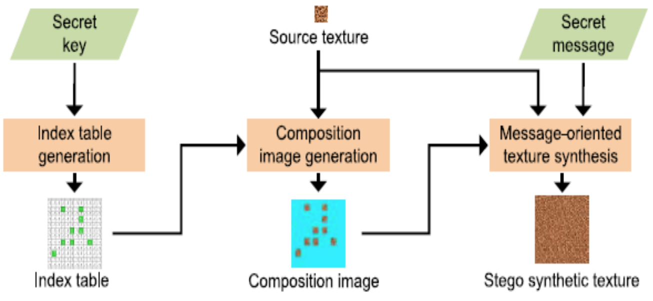


Figure **Error! No text of specified style in document.**-2The flowchart of the three-process message embedding procedure.

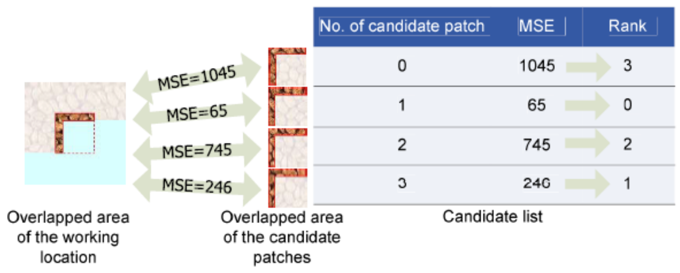
The first process is the index table generation where we produce an index table to record the location of the source patch set SP in the synthetic texture. The index table allows us to access the synthetic texture and retrieve the source texture completely.

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 to reduce the visual artifact on the overlapped area.

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 patchbased 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.


MSE=1045
MSE=65
MSE=745
MSE=246
Overlapped area of the working location
Overlapped area of the candidate patches
Candidate list

No. of candidate patch	MSE	Rank
0	1045	3
1	65	0
2	745	2
3	246	1

Figure **Error! No text of specified style in document.**-3 The MSEs and rank of patches in the candidate list for the working location.

F. 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.

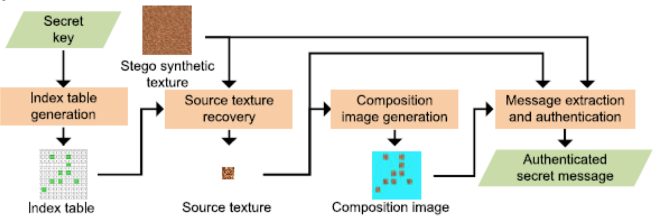


Figure **Error! No text of specified style in document.**-4The flowchart of the four-step message extracting procedure.

The extracting procedure contains four steps, as shown in Fig. 7. Given the secret key held in the receiver side, the same indextable as the embedding procedure can be generated. The next step is the source texture recovery. Each kernel region with the size of Kw × Khand its corresponding order with respect to the size of Sw×Sh source texture can be retrieved by referring to the index table with the dimensions Tpw×Tph. 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 work bench 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.

(4) Dispatching MA: The exploratory data may reach to the sink from many neighbours. The data contains the id of every node that took part in relaying the information from source nodes to the sink, their residual energy i.e., the remaining battery power and the message latency values. The message latency refers to the delay incurred in receiving the response on an interest from the neighbouring nodes from the time a node broadcasts the interest. Once the time mentioned in the interest query expires, the sink which has a dedicated power supply uses:

IV. CONCLUSION

In this paper texture synthesis approach for data hiding is discussed. Five methods of data hiding is discussed out of which only Qr code method is not related with some texture as an input and binary coded texture as an output, it generate a texture image containing a message by putting already assigned pattern for binary code. Stegnography using reversible texture synthesis is most secure as the message is coded and decoded using a secret key only. Authentication and encryption both provided in this algorithm.

REFERENCES

[1] Otori, H.; Kuriyama, S., "Texture Synthesis for Mobile Data Communications," in Computer Graphics and Applications, IEEE, vol.29, no.6, pp.74-81, Nov.-Dec. 2009.

[2] Otori, H.; Kuriyama, S., "Robust Data Hiding on Texture Images," in Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference , pp.563-566, 12-14 Sept. 2009.

[3] Pietroni, N.; Cignoni, P.; Otaduy, M.A.; Scopigno, R., "Solid-Texture Synthesis: A Survey," in Computer Graphics and Applications, IEEE , vol.30, no.4, pp.74-89, July-Aug. 2010.

[4] Sreedevi, P.; Wen-Liang Hwang; Shawmin Lei, "An Examplar-Based Approach for Texture Compaction Synthesis and Retrieval," in Image Processing, IEEE Transactions on , vol.19, no.5, pp.1307-1318, May 2010.

[5] Efros, A.A.; Leung, T.K., "Texture synthesis by non-parametric sampling," in Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on , vol.2, no., pp.1033-1038 vol.2, 1999.

[6] O'Brien, J.T.; Wickramanayake, D.S.; Edirisinghe, E.A.; Bez, H.E., "Image quilting for texture synthesis: a revisit & a variation," in Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on , vol.2, pp.763-767, 15-18 Dec. 2003.

[7] Kuo-Chen Wu; Wang Chung-Ming, "Steganography Using Reversible Texture Synthesis," in Image Processing, IEEE Transactions on , vol.24, no.1, pp.130-139, Jan. 2015.