



Reversible Data Hiding on Textures Combining Patch Based and Pixel Based Sampling

Sreedevi R T¹, Aswathy Devi T²

Student, ECE, LBSITW, Thiruvananthapuram, India¹

Assistant Professor, ECE, LBSITW, Thiruvananthapuram, India²

Abstract: This Texture steganography is a developing area of image steganography. Data hiding in images leads to distortion in image and hence revealing its presence. In such a case it is more suitable to use texture images a cover images as distortion have less effect on texture images. New texture synthesis methods are evolving where as it is triggering the development of new data hiding algorithms using texture images. Reversible algorithms enable the recovery of sample texture from the stego texture. This texture synthesis process re-samples a small texture image in order to synthesize a new texture image with a similar local appearance and arbitrary size. Data hiding is done by incorporating pixel based and patch based sampling of textures. The patch based samples are pasted on to a plane work bench and thus offers a reference platform with reference to which the message oriented texture synthesis using pixel based samples is done. It offers three advantages: first of all reversibility of source texture which is not usually a feature of pixel based texture synthesis, secondly improved data embedding capacity to texture synthesis using patch based sampling and finally improved continuity and local appearance of output images.

Keywords: Texture, texture synthesis, texture steganography, mask image.

I. INTRODUCTION

Steganography, as it implies in Greek as covert writing, is the process of hiding data without revealing its presence. Now-a-days, its wide spread in the field of digital information transfer. It extended its signature in almost all digital media for secret communication between two parties in the medium of exchange of data, such as digital text, images, video, audio, 3D model images etc. Among these, digital images are popular because they are simple to process and also due to the versatility of dimension of the parameters that can be modified for data hiding. In most cases, the data is being hidden by using an existing image as cover image. Such a data hiding leads to either in the distortion of image or limiting the embedding capacity of algorithm in expense of effective data embedding. Texture cover images are least concerned over these limitations, since the basic nature of input texture and output texture remains the same.

II. TEXTURE SYNTHESIS

Texture is simply a patterned image. It may be naturally occurring or the result of synthetic creativity. Naturally occurring textures are continuous and well defined such as flower field, fur, minerals, plants etc. Synthetic textures can be generated manually or by using computer algorithms. Hand drawn textures are easy to create but lack the photo-realistic finishing. Barely placing a pattern repeatedly won't generate a continuous texture. By employing appropriate texture synthesis algorithms visually plausible textures can be generated. Success of texture synthesis algorithms depends on the generation of

aesthetically sound textures. Automatic algorithms are preferred over algorithms requiring manual interactions. Effective sampling of reference textures and its modelling is the important part of those algorithms. Efficiency of sampling will determines the simplicity of remaining texture generation process. Texture synthesis is broadly classified into two: pixel based texture synthesis and patch based texture synthesis.

A. Pixel Based Texture Synthesis

Pixel based algorithms paints textures pixel by pixel to create large texture from the reference texture. It requires the spatial neighbourhood comparisons where the most matching pixel value from the reference texture is painted on the pixel position under consideration.

By physical simulation approach certain textures such as fur, scales, skin, mineral surface patterns, weathering phenomenon etc can be generated by modelling them directly using certain methods like reaction diffusion, cellular texturing, detailed simulation etc. Even though they can generate even minute details on 3D meshes so as to avoid distortion, these methods can only be used to a few type of textures. Some other algorithms uses probability sampling methods such as Markov Random Field, Gibbs sampling etc and they can provide good approximation and better results. The main disadvantage is that, they are computationally expensive taking considerable time for generating even simple textures. In some other algorithms the texture images are modelled with reference to their features and using them the texture



synthesis is done. The main drawback is that these are not universal in nature. These algorithms can successfully generate some textures while in some other cases, they terribly fails.

Fast texture synthesis^[2] algorithm generates new texture by comparing each local region with similar regions in the input texture. The input pixel with most similar neighbourhood pixel values is used to generate output pixels. For ensuring continuity and seamless tiling of the image, pixel values near the reference image boundaries are not considered. Another way is to pad the edges with its own reflected copies, but this will result in introducing discontinuities. So the first method is preferred over the second method. Only local and stationary phenomena can be represented using this process. But other visual effects such as 3D shape, depth, lightning reflection etc can't be modelled by this method. The texture synthesis by non parametric sampling^[3] aims on preserving the local structure of the reference texture throughout the texture synthesis. It synthesises a new image by growing outwards pixel by pixel from an initial seed. The algorithm finds out all the possible neighbourhoods of sample image, which are similar to pixel neighbourhood and any one is randomly selected. Based on weightage of 2D Gaussian kernels, the centre of the area under consideration is selected as newly synthesized pixel. Since this method doesn't assign any constrain for synthesis or sampling region, it is perfect for the constrained texture synthesis application like hole filling. Important limitation of this method is that random texture sampling results may cause the texture generation process to slip to the wrong part by selection of wrong pixels and there by growing garbage on to the texture or get stuck on to one place. Also due to different illuminations of pixels, its hard to choose close matches for the context synthesis window. This can be solved to a large extent by providing larger sample texture. Generally each of the output pixel generated by pixel based texture synthesis depends on previously generated pixels and if any erroneous placing of pixel happens, it will affect the whole texture synthesis process and results in propagation of error.

B. Patch Based Texture Synthesis

Patch based texture synthesis relies on sampling the texture in block level and there by preserving the local appearance in the sampling level itself. These patches are pasted instead of pixels to generate the texture and thus picture quality can be improved. An important drawback of automatic generation by barely pasting sampled patches is that, there is an issue of continuity at the boundary of pasting of the blocks. That is the individual blocks can be identified from the texture losing its naturalness and aesthetic appearance.

Patch stitching is used to place the image patches in such a way to ensure continuous transition from one patch to another. This can be done by choosing appropriate texture which has matching boundary with the neighboring

textures or to make the transition a continuous one by applying appropriate algorithms. Using Wang tiles for texture generation^[4] and patch based sampling method^[5] are utilized for the sampling method while constrained overlap method, feathering and image quilting^[6] falls under the second category. Wang tiles are square shaped tiles with colored edges. Tile pasting is done such that the colored edges match with the corresponding neighboring colored edges. The pasting starts from North West corner of the work bench with any tile selected randomly. Next tile selected such that its west side matches with east side of the first tile. In the next row the matching is done between north of selected tile with south of reference texture other than east west comparisons. Texture generation using Wang tiles^[4] samples the source image and four squared samples images are obtained. These four images are rotated to 45° to have diamond shaped tiles. These are arranged and pasted through four cutting paths through four cutting paths to generate four times large diamond shaped images. This is done with an optimization algorithm. Final diamond shaped sample texture is cut to eliminate corner positions to get possibly large square shaped texture and thus generating a Wang tile from the selected sample images. Similarly seven other Wang tiles are also generated by aligning the samples in other possible positions. These can be used for texture synthesis. Some artifacts may be present because of using a few sample patches and due to the algorithm used for patch stitching. It can be avoided by using more number of Wang tiles and by iteratively finding the best combination of samples to eliminate error. For each set, over all error is calculated by adding up the pixel colour errors along all cutting paths. If error is too high another set of sample images are taken from the reference texture and corresponding Wang tiles are generated. This continues till over all error is minimum or a particular number of iterations. In patch based sampling method^[5], the sampling is done by considering the known boundary zone. Comparing the known boundary with source image, all patches with same boundary zone are selected, to calculate the Markov Random Field density and form an empirical histogram. Since all value in the histogram satisfy the boundary condition, the method allows to select a patch in random, to paste on the vacant space constrained by the boundary. This process is used in constrained texture synthesis process such as tilable texture synthesis, hole filling etc.

In unconstrained texture synthesis after placing a randomly chosen texture patch Bk in lower left corner of the blank output image, the algorithm from a set of patches, that matches with the boundary zone of the first patch such that its corresponding boundary zone matches with it. Then randomly select a patch to continue this process of texture synthesis. As less as the number of matching patches, more similar will be the local structure of output texture. Advantage is that the sampling is fast and can generate visually different textures. Disadvantage is the verbatim copying is unavoidable. To ensure



continuity we can overlap images through boundaries to a particular depth either by overlaying images through transparency or by indexing objects to show according to their random preference. The drawback of overlapping is that sharp transition artifacts of boundaries will remain as such and hence affect the continuity of output image.

Texture stitching using feathering is one of the popular methods used for textures as well as for photographic images. It can avoid discontinuities in intensities and colour changes of the two images being combined. It simply weights the pixel values near to the boundaries based on the Euclidian distance from the edge or the nearest invisible pixel. Then the distinguished foreground object edges are blended to the background. The main disadvantage is that the blending can cause blurring of overlapped region, so that the sharp transition is spread over the thickness of overlapping.

Image quilting [6] involves taking the samples of a reference texture and stitching them in a consistent way. For every new patch to be generated, the algorithm searches for a appropriate patch and compares it with the present working location. Then find a minimum error path through the area of overlapping, such that the mean square error is the minimum. The patches are stitched through that minimum error path. It has high efficiency in generating structured textured compared to any texture synthesis algorithm with less chance of growing garbage values. But there are chances for identifiable discontinuities when the images to be quilted don't satisfy the threshold error limit especially when the images are randomly chosen.

III. TEXTURE STEGANOGRAPHY

It is the process of incorporating data hiding with texture synthesis so as to get a data embedded texture, which can be used for steganographic purposes. The texture steganography can be done by either pixel level generation or patch level generation.

Otori and Kuriyama [7] suggested a pixel level texture synthesis algorithm concealing arbitrary data by generating repeated texture patterns with reference to the features of a sample image. Unlike other techniques of image steganography, this method is not focused on changing the pixel colour component values of the image. Instead, the numerical data to be embedded is mapped to the values of colour domain and there by generating an initial coloured dotted pattern. This dotted pattern is then camouflaged by generating a texture pattern with reference to the sample texture, so as to get a uniform texture which appears to be generated from reference texture. However, there is improvement in number of error bits compared to the traditional texture synthesis methods, such that the embedded data can be recovered by applying error correction techniques, even though their perfect detection is not possible. Another limitation is that the border lines

of the image should be known for identifying the embedded bit positions. The reference texture cannot be recovered back from this type of data embedding.

Wu and Wang [8] suggested a reversible texture synthesis algorithm for steganography. It uses patch level sampling of reference texture for texture synthesis and data hiding. From source texture two sets of patches are generated called source patches and candidate patches. The source texture is divided into kernels of equal size and then extended to boundary region to a constant number of pixels to all sides to generate a new set of patches of size greater than the kernels to generate source patches. The next of patches are generated by using a window function of size same as that of source patch moving along the source texture in a scale-line sequence pixel by pixel. The candidate patch set is redefined by eliminating duplicate patches in order to avoid erroneous extraction of message that is embedded. The source patches are pasted initially on a plane image with reference to an index table and the candidate patches are used to fill the remaining space with reference to the message to be embedded. This is a reversible texture synthesis process and for that the position of pasting of source patches need to be predefined. With reference to a secret key, an index table is generated in which the patch index values corresponding to each patch, are scattered along a matrix and all other values are set to a negative constant. This will take the first priority and the second priority positions into consideration for placing the patch index values.

First priority positions L1 are those positions which neither comes on the boundary region nor share neighbourhood with other index values. Second priority positions L2 won't come in boundaries but share their corners with first priority positions. Index table values will satisfy these conditions.

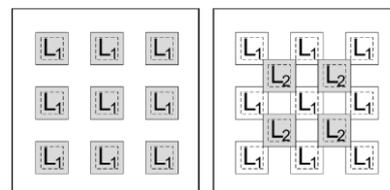


Fig. 6 Priority positions of a work bench

By placing source patches on a plane image with respect to the patch index in the index table, a composition image is generated. With reference to the patch composition image and by selecting n bits from message binary sequence, the message embedding is done by placing candidate patches on vacant position. Considering a working location, the candidate patches are ranked based on the mean square error, they would have made if placed at that location. The particular candidate patch is selected with the rank equal to the decimal value of bits to be embedded. This will continues by taking next n bits and embedding it till all the message bits are embedded, to get



the stego synthetic texture. To ensure continuity and ensure aesthetic appearance of the generated texture, the patches are stitched through a minimum error path using image quilting technique.^[6]

During message extraction, with reference to index table generated by using secret key, the source patches are extracted. By rearranging with respect to patch index, the source texture is retrieved. Using the recovered source texture, candidate patch set and source patch set are generated. Candidate patch set redefined to eliminate the duplicate patches using some criteria as the encoding section. Composition image is obtained referring to index table of source patch. By comparing working locations of stego synthetic texture and composition image, candidate patches are ranked. Rank corresponding to the matching patch with respect to working location is appended to a string in binary form to extract the message back. By comparing the results with reference to mean square error of overlapping area, it is revealed that for low number of bits embedded per patch and for low overlapping area, the mean square error found to be low.

But in every case of texture generation have comparable structural similarity index so that every stego texture is identical in appearance with negligible distortion. The main drawback of this method is that the embedding capacity depends on the number of candidate patches that can be included which is already being constrained by the output image size. Also the limitation of image quilting algorithm on random patch selection may affect the continuity and quality of output image.

IV. PROPOSED METHOD

The source image is divided into a number of kernels and these kernels are used as source patches. An index table is generated by randomizing the patch IDs among -1 values by using a secret key. The source patches are pasted on a plane work bench w:r:t their patch IDs in index table^[8], to get the composition image. Patch based sampling is used for the composition image generation and the remaining data hiding process is done by pixel based sampling.

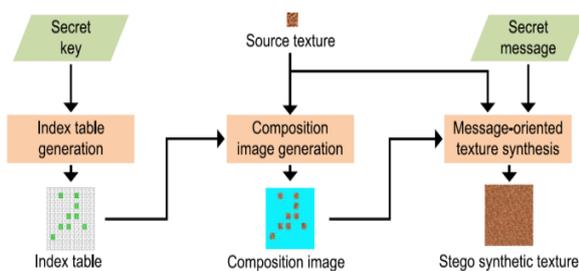


Fig. 7 Data embedding process

The secret message is converted to binary bits and then grouped based on the following criteria. If the number of bits is greater than the number of unfilled pixels i.e., the

embedding capacity, then bits are hid one by one. Otherwise, the bits are grouped in m numbered to form M bit groups $\{m_1, m_2, m_3, \dots, m_M\}$ such that M is less than the embedding capacity.

The composition image in grey scale is converted to binary to get a mask image where filled bit positions are represented in white and unfilled in black. The set of boundary pixels are found by dilating the mask image and then taking its difference from original mask image. These pixel positions are filled one by one and boundary filling is done. This process is repeated to embed data on unfilled region by selecting appropriate pixel value.

Consider a pixel position in unfilled region of the boundary under consideration. By applying a Gaussian mask of order 3 X 3 on that position find out the matching pixels with the same boundary from the source texture. After finding their squared sum of distance (SSD) redefine the set of pixels based on a maximum error threshold value. Now select the pixel with its index value equal to the data to be embedded. E.g. If decimal value of $m_1=1$, then select the pixel with index 1. This continues till the whole message is embedded. After that if any unfilled portions remains, then it is filled using random pattern.

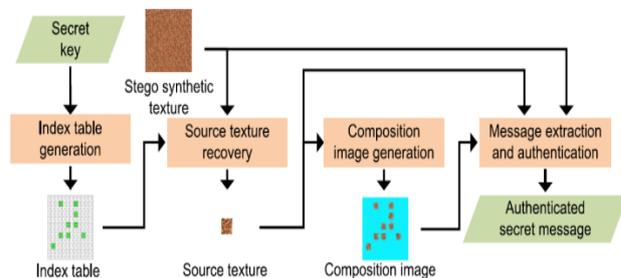


Fig. 8 Data extraction process

At receiver section, the secret key and the number of message bits embedded are known. Using the secret key the index table is generated w:r:t which the source patches are retrieved and rearranged to get the source texture. Composition image based on the retrieved source texture is generated and its mask image is used to find the boundary regions. Then bits embedded in each pixel are retrieved by repeating the same process as in embedding. The index values of the matching pixel are retrieved to get the embedded message.

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

-1	-1	-1	-1	-1	-1
-1	1	-1	3	-1	-1
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1
-1	2	-1	4	-1	-1
-1	-1	-1	-1	-1	-1

Fig. 9 Index table

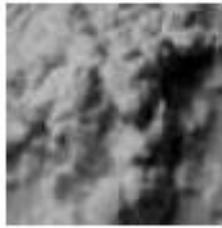


Fig. 10 Source image

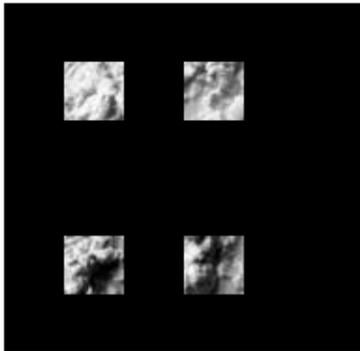


Fig. 11 Composition image

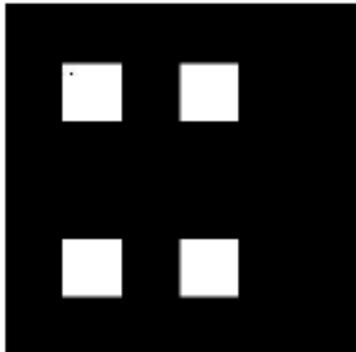


Fig. 12 Mask image of composition image

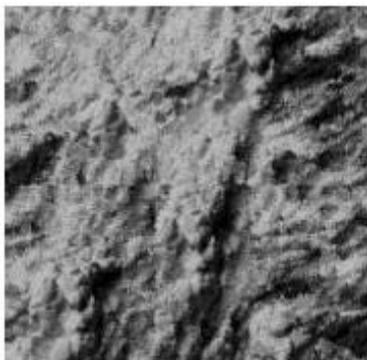


Fig. 13 output image

The main advantage of this method is its improved ratio of embedding capacity to the size of output image compared to patch based texture steganography [8]. Also the local appearance of the output image is very much improved compared to usual pixel based texture synthesis methods.

It can be clearly seen in the output image that the area in output texture, where the source patches are distributed heavily is showing more similarity to the source texture compared to the area where the source patches are weakly distributed. This shows that the placing of reference source patches during data hiding improved the visual quality of images. This reveals that by uniformly distributing the source patches, it is easy to achieve this trough out the image. Another important advantage is that the steganographic approach is reversible. i.e., the source texture can be retrieved back from the output image.

TABLE I VARIOUS PARAMETERS OF STEGO SYNTHETIC TEXTURES

Source image	Parameters of output stego synthetic texture		
	Mean square error	Computation time(s)	Bit error rate (%)
 Yogurt.gif	0.2305	21.017302	7.792
 Flower.jpg	0.3109	18.324033	9.091
 Marble.jpg	0.1479	18.232013	2.597
 Rice.bmp	0.2949	17.807175	6.493

The Table 1 shows the different parameters of the output stego synthetic textures generated from corresponding source texture inputs. Perfect texture images like marble patterns show minimum mean square error which indicates that the basic nature of image is preserved in the generated stego texture. The computation time is a measure of number of comparisons and calculations required for having the required texture synthesis process. As the number of pixels with comparable neighbourhood increases, the computations also increase and so as the computation time. The bit error rate is calculated by taking the ratio of the error bits to the total bits, and then converting the ratio to the per cent scale.

VI. CONCLUSION

In this paper we went through different methods of texture synthesis and steganography in general and texture steganography in particular. Data hiding using texture is becoming one of the reliable methods of image steganography. Embedding messages on texture will leaves less chance for identifying the presence of a hidden



message with bare eyes, due to the universal appearance of textures. The limitations of low embedding capacity of patch based texture steganography can be overcome by this method. It can be extended by applying different algorithms for enhancing the embedding capacity and by improving the sampling methods. The visual quality of the stego image is also increased compared to the usual pixel based texture synthesis algorithms. The level of security can be improved by randomizing the order of embedding of pixels in each boundary filling and the message sequence itself.

REFERENCES

- [1] S 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [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. 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.
- [7] H. Otori and S. Kuriyama, "Data-embeddable texture synthesis," in *Proc. 8th Int. Symp. Smart Graph*, Kyoto, Japan, 2007, pp. 146-157.
- [8] Kuo Chen Wu and Chung Ming Wang, "Steganography using reversible texture synthesis" *IEEE Trans. Image Process.*, vol.24, no.1, pp. 130-139, Jan. 2015.