# WAVELET BASED TEXTURE SYNTHESIS

Claire Gallagher and Anil Kokaram
Department of Electronic and Electrical Engineering
Trinity College
Dublin, Ireland.
email: `gallaghc@mee.tcd.ie`

**Abstract**

This paper presents a new algorithm for synthesising image texture. Texture synthesis is an important process in image post-production. The best previous approaches have used non-parametric methods for synthesising texture. Unfortunately, these methods generally suffer from high computational cost and difficulty in handling scale in the synthesis process. This paper introduces a new idea of using wavelet decomposition as a basis for non-parametric texture synthesis. The results show an order of magnitude improvement in computational speed and a better approximation of the dominant scale in the synthesised texture.

**Keywords**: *Texture Synthesis, Complex Wavelet Transform, Image Processing, Non-parametric Image Modeling.*
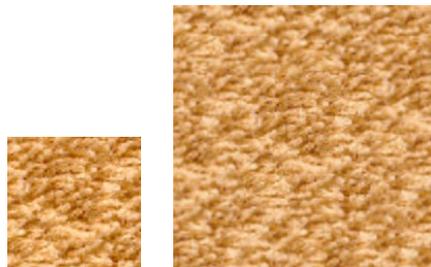
Figure 1: Texture synthesis: Given an example texture $I_e$ as an input (left), the algorithm aims to reproduce new texture $I_s$ (right).

## 1 Introduction

The problem of texture synthesis has been an active research topic in recent years [5, 4, 15, 10]. Given an example of texture as a small subimage, the idea is to create a much larger image by synthesising *more* texture. Figure 1 shows on the left a typical example image or "seed" of size $128 \times 128$ and on the right is the synthesised image of size $256 \times 256$ created by surrounding this "seed" with *new* texture. This kind of operation is often required in the post-production of digital images when a large area is to be covered with texture that *looks like* some smaller example. Picture editing often requires filling of missing information and texture synthesis processes like these can fill such holes with reasonable material.

The essential idea is to somehow estimate the p.d.f. of the image intensity $I(\mathbf{x})$, denoted by $P(I(\underline{\mathbf{x}}))$ at a pixel site $\underline{\mathbf{x}} = (i, j)$. The process of texture synthesis is then a matter of drawing a random sample from that distribution. What makes this difficult is estimating $P(I(\underline{\mathbf{x}}))$. Two different approaches have

emerged. Parametric techniques attempt to model $P(I(\underline{x}))$ with some definable process. Heeger and Berger [6] analyse texture using histograms of filter responses at multiple scales and orientations. Portilla and Simoncelli [11] improve on this idea by matching pairwise statistics across different scales and orientations. Kokaram [10] uses an autoregressive model when synthesising texture. All of these methods work well on simple textures but fail for more structured textures [11]. Non-parametric approaches rather, attempt simply to *measure* the p.d.f. from the image. The visual quality of the generated textures will be influenced primarily by the accuracy of the model, while the efficiency of the sampling procedure will be directly related to the computational expense [15]. Because of the wide variability in image behavior non-parametric approaches have achieved by far the more visibly pleasing results [5, 15, 1, 14, 2].

Most of the non-parametric methods rely on an idea introduced by Efros and Leung in 1999 [5]. Their approach was based on empirical measurement of the p.d.f. of a pixel using neighbourhood similarity. This method assumes texture can be modeled by a Markov Random Field (MRF), i.e. the intensity value for a pixel given the intensities of its spatial neighbourhood is independent of the rest of the image. The p.d.f. $P(I(\underline{x}))$ is then sampled and the newly assigned pixel is assigned to the synthesised image. This algorithm generates impressive results and works well on a large range of textures. However, computational cost is high because an entire search of the sample image is necessary for each of the pixels to be synthesised. In addition, the success of the algorithm is dependent on the correct choice of neighbourhood size. This user defined parameter controls the randomness of the texture to be generated.

Ashikhmin [1], Bornard [2] and Pei et al. [14] address the computational burden of the Efros algorithm by introducing coherent searching into the synthesis procedure. This speeds up the synthesis process by eliminating the need to search every possible neighbourhood in the sample image. Wei and Levoy [15] develop the algorithm further to include multi-resolution synthesis. They use Gaussian pyramids to represent the texture and transform a random noise sample to resemble the sample texture at different levels of the Guassian pyramid. This method works well on stochastic (random) textures but is not suitable for deterministic (structured) textures [4].

In order to explore the problems of scale and computational load associated with non-parametric methods, we have introduced the novel idea of using the complex wavelet transform as a basis for non-parametric texture synthesis. The introduction of the wavelet decomposition into the synthesis procedure has two advantages. Firstly, it facilitates the measurement of texture statistics at particular scales. Unlike previous methods, who use scale information as a control [15], we directly synthesise texture at these different scales. This allows us to exploit the dominant frequencies present in the texture image. The second advantage of our method is the reduction in computational load. By synthesising texture at coarser scales, the original information is represented by fewer pixels. Large features which were present at a fine scale, are now much smaller and can be represented by smaller neighbourhoods. Synthesising texture at these coarser scales is much more computationally efficient than synthesising texture at a fine scale. This is due to the reduction in size of the image to be synthesised (sub image of original image). In addition, because large features can be represented by smaller neighbourhoods, the neighbourhood size is reduced considerably thus improving computational cost further.

The following sections outline the single resolution non-parametric algorithm and illustrate how wavelet decomposition may be used as a basis for this non-parametric texture synthesis. A comparison is given between our proposed method and the best previous approaches. This comparison is based on computational load as well as visual texture results. Finally, advantages and limitations of our algorithm are presented.

## 2  Single Resolution Texture Synthesis

Let $\mathbf{X}_s$ represent the image grid of size $M \times N$ to be synthesised and $I_e$ be the sample input image of size $m \times n$ specified on the smaller grid $\mathbf{X}_e$. The algorithm assumes that $I_e$ is large enough to capture the statistics of the underlying infinite texture. Let $\mathbf{p} \in \mathbf{X}$ be a pixel to be synthesised and $w(\mathbf{p})$ be the spatial

neighbourhood of pixels surrounding $\mathbf{p}$ with width $w$. To synthesise a value for $\mathbf{p}$ an approximation to the conditional probability distribution $P(\mathbf{p}|w(\mathbf{p}))$ is constructed and then sampled. The approximation is built by directly identifying all patches in the sample image that are *perceptually similar* in some way to the existing neighbourhood around the pixel to be synthesised. The pixels at the centre of these similar patches then represent an empirical measurement of the p.d.f. required.

Let $d(w(\mathbf{p_1}), w(\mathbf{p_2}))$ denote the perceptual distance between two neighbourhoods or patches centred at locations $\mathbf{p_1}$ and $\mathbf{p_2}$. $d$ is defined to be the sum of squared intensity differences. The best matching patch $w_{best}$ in the sample image, is first found, $w_{best} = argmin_{\mathbf{x} \in \mathbf{X_e}} d(w(\mathbf{p}), \mathbf{x})$. All example image patches $w$ with $d(w(\mathbf{p}), w)) < (1 + \epsilon) d(w(\mathbf{p}), w_{best})$ are included in the set $\Omega(\mathbf{p})$. In this application $\epsilon = 0.1$. The centre pixel values of patches in $\Omega(\mathbf{p})$ gives a histogram for $\mathbf{p}$ which can then be used to obtain a sample numerically. To preserve the local structure of the texture, the error for pixels near the centre of the neighbourhood i.e. that corresponding to $\mathbf{p}$, is larger than that for pixels close to the edge of the neighbourhood. This is achieved by weighting the distance measure $d(\cdot, \cdot)$ with a two-dimensional Gaussian Kernel. A kernel with variance $w/6.4$ is used.

In practice it is sensible to visit pixels in the synthesised image in an order specified by the number of known spatial neighbours. The algorithm initially seeks out pixel $\mathbf{p} \in \mathbf{I_s}$ with the most known spatial neighbours. As some of the spatial neighbours of $\mathbf{p}$ are unknown, the distance measure is modified to match only the known values in $w(\mathbf{p})$. This error is then normalised by the total number of known pixels when computing the conditional p.d.f. for $\mathbf{p}$. Figure 2 illustrates an overview of this searching procedure.
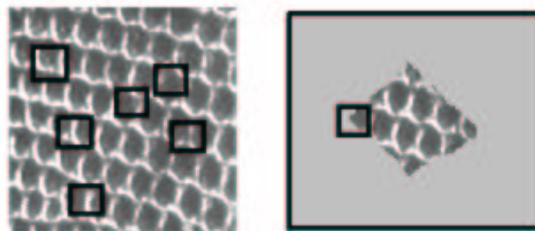


Figure 2: For each unknown pixel $\mathbf{p}$ in the synthesised image $I_s$ (right) the algorithm searches all possible neighbourhoods in the sample image $I_e$ (left) for a neighbourhood similar to that of the pixel $\mathbf{p}$. It then randomly chooses a matching neighbourhood and takes its centre to be the newly synthesised pixel.

Problems with boundary conditions are avoided by either treating the boundaries toroidally or padding with zeros. Here all boundaries were padded with zeros. The above algorithm generates impressive results on a wide variety of textures. However, searching the entire sample image for each pixel is computationally expensive and slows the algorithm considerably. A breakdown of the computational cost is given in section 3.2. In addition, the user defined neighbourhood width is critical to successful texture synthesis. To address these problems and also demonstrate the power of wavelets, the complex wavelet transform has been incorporated into the synthesis process.

# 3   Synthesising Texture using the Complex Wavelet Transform

The Dual Tree Complex Wavelet Transform (DT-CWT) originally proposed by Kingsbury has received much interest in image processing applications recently [8, 9, 13, 3, 7]. It builds upon the orthogonal Discrete Wavelet transform (DWT) and addresses some of its limitations such as, lack of shift invariance and poor directional selectivity [9]. The DT-CWT uses a dual tree of wavelet features that are assigned as real and imaginary components of complex wavelet coefficients. A full explanation of how the wavelet transform operates is beyond the scope of this paper but the interested reader is directed towards [9, 12]

for some supplementary material. As an outline however, the biorthogonal 2D DWT produces three band pass sub images at each level of the transform. These correspond to the lo-hi, hi-hi, hi-lo. The lo-lo sub image is passed onto the next level of the transformation. It is found that with real images, most of the significant information is contained within the first and second quadrants of the spectrum [13]. The 2D DT-CWT exploits this by producing three band pass sub images in each of the spectral quadrants 1 and 2. This gives a total of six band pass images with complex coefficients at each level. These images are strongly oriented at angles of $\pm 15^o, \pm 45^o, \pm 75^o$. Figure 3 shows the complex wavelet decomposition of an image containing a single bright circle. The sub band and lowpass images for the first level of decomposition are shown. The figure illustrates the directional sensitivity of the transform since different bands emphasise different parts of the circle contour. The DT-CWT gives a 4:1 redundancy for 2D images,. In a sense it is this redundancy that allows both shift invariance and good directional sensitivity.
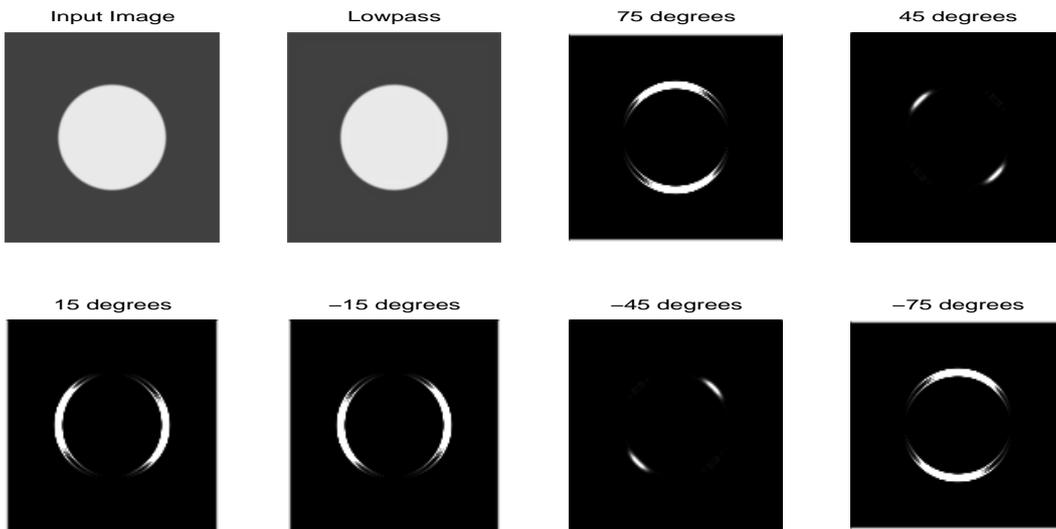


Figure 3: Illustration of sub image produced using DT-CWT. Figure shows input image (top left), lowpass image (top middle) and the bandpass images.

## 3.1 Algorithm

Given the initial sample image $I_e$ of size $n \times m$ and the required output size $N \times M$ of the image to be synthesised $I_s$, the algorithm proceeds as follows.

- The $n$ level complex wavelet transform is performed on the example image $I_e$. Using the initial dimensions of the image to be synthesised $I_s$, the dimensions of each of the sub band images and the final lowpass image are calculated. A sample of $I_e$ is placed at the centre of each of the sub images of $I_s$. The size of the sample used should be consistent among the levels, i.e. at level $n$ the seed should be half that used at level $n-1$. This is because of sub sampling in the wavelet transform. This sample is then surrounded by negative ones to indicate wavelet coefficients values to be synthesised.

- At the highest level, which is the coarsest level in terms of detail, the Efros searching algorithm given in section 2 is used to synthesise unknown wavelet coefficients in each of the six sub band images. In order to account for the correlation among sub band images and to maintain the efficiency of the algorithm, each of these images are searched coherently. That is, the same wavelet coefficient coordinate in each image is synthesised in parallel. Neighbourhoods from the six sub

band images and with the same centre coordinates are represented by a vector. The distance between two neighbourhoods is then given by the difference in magnitude between the two vectors representing them.

- Once the chosen wavelet coefficient has been selected from the sample image $I_e$, the six sub band images at the highest level are updated. Wavelet coefficients on the levels below follow the movement at the top level. This relationship is shown in Figure 3.1. That is, the wavelet coefficient at position $(i, j)$ at level $n$ corresponds to coefficients $(2i, 2j), (2i - 1, 2j), (2i, 2j - 1)$ and $(2i - 1, 2j - 1)$.

- This process is repeated for all unknown wavelet coefficients at the highest level. Once all of the wavelet coefficients have been generated, the synthesised image is inverse transformed to give an image that should resemble that of the sample texture. Note that, in order to avoid problems with boundary conditions, it is necessary to pad each sub image with zeros before performing the algorithm. This padding should be removed prior to inverse transform.

The above steps are based on generating grayscale images. To synthesise colour textures, first transform the image from the $rgb$ colour space to the $yuv$ colour space. Perform synthesis on the $y$ (luminance) component and then propagate relevant coordinates to $u$ and $v$ components.
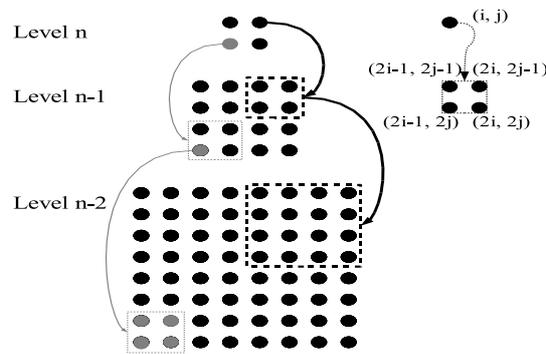


Figure 4: Simplified sub image of the DT-CWT showing the relationship between the wavelet coefficients across the different levels.

## 3.2  Computational Load

In order to demonstrate the advantages of our algorithm in terms of computational cost we have compared it against the original Efros algorithm [5]. Given a sample image $I_e$ of size $m \times n$ and the image to be synthesised $I_s$ of size $M \times N$. Let $\mathbf{p} \in I_s$ be a pixel to be synthesised and let $w$ be the width of the neighbourhood of the square spatial neighbourhood surrounding $\mathbf{p}$. For each pixel to be synthesised in $I_s$, the algorithm needs to search up to $nm$ locations. At each of those locations, $4w^2$ operations need to be performed to calculate the weighted sum squared difference. This is $4nmw^2$ operations in total for each searched site. Therefore to generate $I_s$ of size $M \times N$ the algorithm will have to perform $4NMnmw^2$ operations.

In comparison, the algorithm proposed in this paper synthesises texture at the third level of the complex wavelet transform. At this level the dimensions of the sample image are $nm/16$ and the image to be synthesised are $NM/16$. Since all the six sub images at this level must be searched, the total number of operations is given as $NM/16 \times nm/16 \times 4w_1^2 \times 6$. Here $w_1$ is the neighboiurhood size for this process and is typically smaller than that needed for the Efros algorithm. The load for the CWT is roughly $80NM$ and is negligible in comparison to the overall load. Therefore the overall computational load of the new CWT algorithm is given by $NMmnw_1^2/10$. This shows that the new algorithm is faster than the

original Efros algorithm by a factor $40w^2/w_1^2$. For the experiments shown in this paper $w = 11$, $w_1 = 5$ yielding an improvement of a factor of about 200.

Using a simple Matlab implementation for a grayscale image on a 2.4 GHz P4 PC, the CWT algorithm can generate a $256 \times 256$ image from a sample texture measuring $128 \times 128$ in approximately 60 seconds. For a colour image, this process takes just over 80 seconds.

## 4  Results

Synthesised images generated by the wavelet synthesis algorithm are shown in Figures 5 and 6. In order to demonstrate the effectiveness of the algorithm, it was tested on a wide range of different textures. A visual comparison of the results obtained using other approaches was also carried out. Some of these results are shown in Figure 6. In each case the sample image measured $128 \times 128$ pixels and the synthesised image measured $256 \times 256$ pixels. When using complex wavelets, it is optimal to use image sizes that are powers of 2. Texture synthesis was carried out at level 3 of the complex wavelet with a neighbourhood size of $5 \times 5$ pixels.

As can be seen from Figure 6, the wavelet texture synthesis algorithm compares well against results obtained using the Wei and Levoy [15] and Efros and Leung [5] methods. The Wei and Levoy algorithm is similar to the method proposed here in that it is based on multiresolution synthesis. In their case they use Gaussian pyramids to separate the image into various frequency bands. When synthesising a pixel they begin initially at the top level and work their way down the pyramid. The neighbourhood of each pixel incorporates those pixels situated a level above on the pyramid. This allows for correlation among the sub images. However, it implies that the neighbourhood size is large, thus slowing down the process. Their tree vectorisation overcomes this but synthesising the entire Gaussian pyramid one pixel at a time is still computationally expensive.

The synthesised text in Figure 5 shows the impact of scale in texture synthesis. Because the algorithm synthesises at level 3 of the complex wavelet transform, whole words are synthesised rather than letters. This clearly demonstrates the effect of scale. At high levels of the transform, large features (words) are represented by fewer pixels. By synthesising texture at this level, words rather than individual letters are generated. Because the Efros method synthesises on a fine scale it will grow letters rather than words. That is, it grows the texture rather than the individual text.

Visually the textures generated using our CWT method compare well against the sample texture. However, following close inspection, there is some blurring present in the synthesised texture. This is more perceivable in sharp textures than others, e.g. the text. This problem is due to using the coarse level synthesis to direct the synthesis of the other levels, thus the detail at the finer levels is not refined. In addition, if the original sample image is compressed then these compression artifacts will be propagated in the synthesised image thus leading to more visual errors. Resolving this problem is the direction of current work.

## 5  Final Comments

In this paper a new texture synthesis algorithm was introduced. Given an initial sample image, the algorithm generates new texture using a simple searching process and which incorporates the Dual Tree Complex Wavelet Transform (DT-CWT). Results show that the algorithm works well on a wide variety of textures and has the advantage of reduced computational cost. By exploiting the properties of the DT-CWT, the algorithm also addresses some the problems of scale and correct neighbourhood size. Future work involves addressing some of the blurring problems associated with the output results. This will involve refining the pixel choice rather than just copying and adjusting the coordinates from those attained at the highest level.
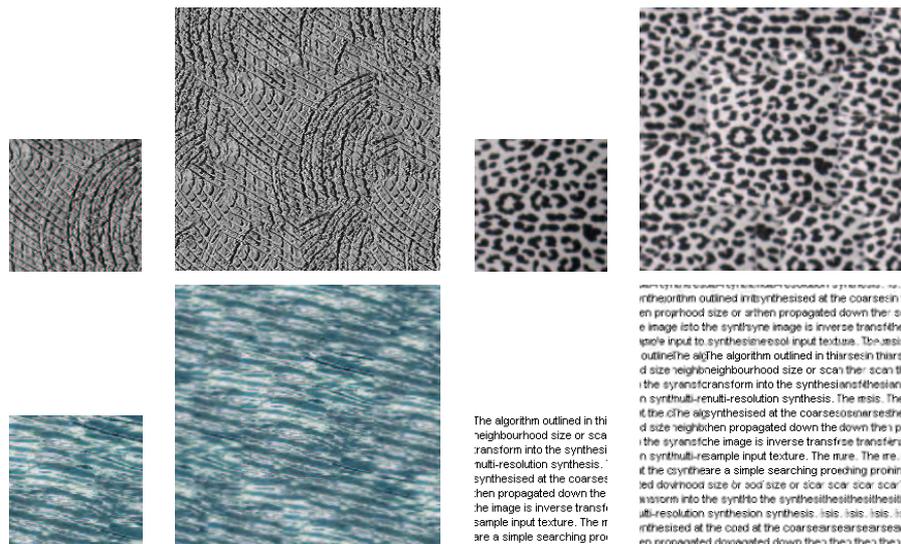
Figure 5: Results from the texture synthesis. The left hand image is the original image while the right is the synthesised image. Textures were synthesised on the third level of the transform.

## Acknowledgements

## References

[1] Michael Ashikhmin. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, pages 217–226, Research Triangle Park, North Carolina, USA, March 2001.

[2] Raphaël Bornard. *Probabilistic Approaches for the Digital Restoration of Television Archives*. PhD thesis, École Centrale Paris, 2002.

[3] Peter de Rivaz and Nick Kingsbury. Complex wavelet features for fast texture image retrieval. *Proceedings of IEEE Conference on Image Processing, Kobe Japan*, pages 25–28, October 1999.

[4] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, pages 341–346, August 2001.

[5] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999.

[6] D.J. Heeger and J.R. Bergen. Pyramid based texture analysis or synthesis. *SIGGRAPH 1995 Conference Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison Wesley*, pages 229–238, August 1995.

[7] Cian W. Shaffrey Nick G. Kingsbury and Ian H. Jermyn. Unsupervised image segmentation via markov trees and complex wavelets. *Proceedings of IEEE Conference on Image Processing, New York, U.S.A*, September 2002.

[8] Nick Kingsbury. The dual-tree complex wavelet transform: A new technique for shift invariance and directional filters. In *IEEE DSP Workshop paper no. 86*, Bryce Canyon UT, USA, 1998.

[9] Nick Kingsbury. Image processing with complex wavelets. In *Phil. Trans. Royal Society London A, September 1999, on a Discussion Meeting on "Wavelets: The Key to Intermittent Information?"*, February 1999.

[10] Anil Kokaram. Parametric texture synthesis for filling holes in pictures. In *IEEE International Conference on Image Processing*, pages 325–328, Rochester, New York, USA, September 2002.
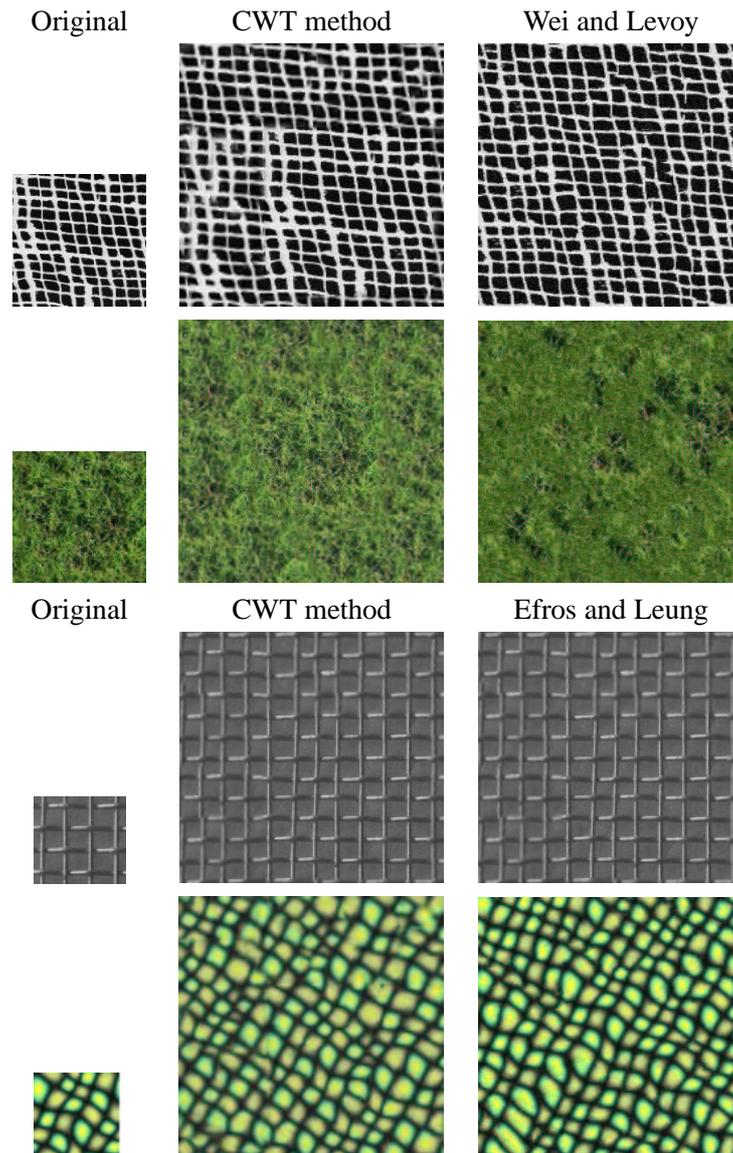
| Original | CWT method | Wei and Levoy |
| --- | --- | --- |

| Original | CWT method | Efros and Leung |
| --- | --- | --- |

Figure 6: Comparison of different texture synthesis methods.

[11] J. Portilla and E.P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int'l Journal of Computer Vision, vol.40(1)*, pages 49–71, December 2000.

[12] Olivier Rioul and Martin Vetterli. Wavelets and signal processing. *IEE Signal Processing Magazine vol 8, no 4*, October 1991.

[13] Sanjit K. Mitra Serkan Hatipoglu and Nick Kingsbury. Texture classification using dual-tree complex wavelet tranform. *Image Processing and its Applications, Conference Publication No. 465 IEE*, 1999.

[14] Yi-Chong Zeng Soo-Chang Pei and Ching-Hua Chang. Virtual restoration of ancient chinese paintings using color contrast enhancement and lacuna texture synthesis. *IEEE Transactions on Image Processing, volume 13, number 3*, March 2004.

[15] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. *Proceedings of SIGGRAPH 2000*, pages 479–488, 2000.