# Texture Synthesis using Image Pyramids and Self-Organizing Maps

Patricio Parada, Javier Ruiz-del-Solar
*Dept. of Electrical Engineering,*
*Universidad de Chile, Santiago, CHILE*
*Email: {pparada, jruizd}@cec.uchile.cl*

**Abstract**

Markov Random Field is a very known model for the generation of textures. However, the estimation of its parameter results quite difficult in many cases. In this article a new algorithm for synthesis of textures is proposed, based on image pyramids and self-organizing maps. This procedure avoids the explicit computation of its parameters. Preliminary results support the appropriateness of this new approach.

## 1. Introduction

Texture is a common visual experience in our lives. Textures are homogeneous visual patterns that produce a uniformity sensation on the observer. A texture can be defined through two main features of it: local patterns and a placement rule. These elements, combined adequately, generate a huge variety of images.

Texture synthesis has been an increasingly active research field in computer graphics. Its final goal is to generate a new texture image from an example texture, whose visual perception is similar to the original one. Numerous approaches have been proposed for this task, and they can be divided in three main categories: based in physical simulations (reaction-diffusion or cellular texturing models, for example), based in structural models, or finally, based in statistical models. In this last group, Markov Random Fields [3] [4] and Autoregressive models [2] stand out for their large versatility and performance.

The texture synthesis method proposed in this article is based in the original idea of Wei and Levoy [9]. Their synthesis procedure, unlike most MRF based algorithms, is completely deterministic and avoids the explicit estimation of the probability distribution function. Using this idea, we have proposed a new texture synthesis method based on image pyramids and self-organizing maps, with very good results.

This paper is organized as follows. In section 2, is briefly reviewed some of the most representative approaches to texture synthesis. In section 3, is described the proposed algorithm. In section 4 are shown some of the preliminary results of this research. Finally, in section 5 are presented conclusions of the work.

## 2. Texture Synthesis

Textures are often been classified in two categories: deterministic and stochastic. A deterministic texture is characterized by a set of primitives or texels (texture elements), which corresponds to basic patterns, and a placement rule. On the other hand, a stochastic texture does not have easily identifiable primitives or regular placement rules (e.g. granite, bark, sand). Many natural textures (i.e. textures present in the nature) have some mixture of these two aspects.

Much of the research on texture synthesis can be classified according to what type of texture model is being used. Some of the successful texture models include physical simulations (reaction-diffusion [6]), frequency domain approaches [8], statistical/random field models [3][4] and autoregressive models [2]. In the last decade, a new approach, called feature matching, has appeared. Under this approach, a set of texture features is calculated, and new images are generated by matching the features of sample-textures [5].

It is desirable to develop an algorithm that combines the advantages of the previous approaches. In our method we use Markov Random Fields (MRF) for the texture model, since they have been proven to cover a huge variety of useful texture types. MRF model a texture as a realization of a local and stationary random process, i.e. each pixel of a texture image is characterized by a small set of neighboring pixels, and this characterization is the same for all pixels in the image [9]. Image pyramids and self-organizing maps are used for avoiding the explicit computation of the MRF parameters. In this sense, this approach is a kind of feature matching generation of textures.

## 3. Proposed System

### 3.1. Previous Work

Our texture synthesis model is based in the work developed by Wei and Levoy [9]. New textures are generated pixel by pixel, and each pixel is determined so that local similarity is preserved between the example texture and the result image. The synthesis process, unlike most MRF based algorithms, is completely deterministic and it does not need to calculate any probability distribution.

The algorithm starts with an input texture $T_I$, and an output texture, initiated as white random noise, $T_O$. The process will force $T_O$ to look like $T_I$, by a pixel-by-pixel transformation. To determinate the gray level of the pixel $p$ in $T_O$, its spatial neighborhood $N(p)$ is compared against all possible neighborhoods $N(i)$ from $T_I$. The gray level of the pixel $j$ is assigned to $p$, if $N(j)$ is the most similar neighborhood to $N(p)$. This similarity can be measured using the Euclidean Norm, or another distance function. With this approach, one pursues maintain as much local similarity between $T_I$ and $T_O$ as possible.

This deterministic synthesis procedure avoids the usual sampling of the MRF. However, the algorithm uses exhaustive searching, which makes it too slow. This can be faced by considering neighborhoods $N(p)$ as points in a multidimensional space, and viewing the neighborhood matching process as a *nearest-point search problem*.

The nearest-point search problem in a vector space can be stated as follows: given a set $A$ of $n$ points and a novel query point $q$, find a point $p$ in $A$ such that the distance from $q$ to $p$ is lesser than, or equal to, the distance from $q$ to any other point in the set. In the texture case, these points usually form clusters in the space, and therefore, the search problem might be solved using a vector quantization procedure.

The quality of the textures generated depends on the size and shape of the neighborhoods $N(i)$. If an adequate size is used, the algorithm will be able to capture the structures of the textures. Nevertheless, in many cases, the necessary neighborhood size is too large, which results in a big computation effort. This problem can be faced using a multi resolution approach. Particularly, image pyramids [1] provide an efficient solution representing large texture structures by a few pixels in a given level of lower resolution.

### 3.2. Innovations

The schema proposed by Wei and Levoy has two main parts. On the one hand, the problem of finding an adequate size of the neighborhood, which is solved defining a "spatial" neighborhood that extends the usual bi-dimensional one to several levels in a image pyramid. On the other hand, the neighborhood matching process is solved preprocessing all the possible $N(i)$ vectors in a data structure that allows perform a quick search. Their particular implementation uses TSVQ (Tree Structured Vector Quantization), a data structure that is constructed for performing fast searches but that uses too much memory resources.

These two issues are improved in a different way by this paper. In first place, the multiresolution structure can be used taking more advantages from the pyramid image representation. Second, the neighborhood matching process, considered as a nearest-point search problem, can be solved using SOM instead the TSVQ structure, because it allows not only perform vector quantization but also it maps the topologic structure of the original space into the network.

### 3.3. Self-Organizing Map in Texture Synthesis

The algorithm proposed employees the Self-Organizing Map (SOM) [7] for solving the nearest-point search problem. SOM corresponds to a very known neural model, which uses an unsupervised learning. SOM networks are usually used to solve clustering problems, association between patterns, and even classification tasks [7]. In the texture synthesis process, it is necessary to build a neighborhood $N(p)$ associated to each pixel $p$ in the original texture image, by copying the gray values of the neighboring pixels into an array or vector. Therefore, for a given image $I$ is possible to gather all the vectors in one single set. The cardinality of this set depends on the size of the texture-sample image and the neighborhood size considered. For instance, if we use a causal neighborhood, whose size is 3, the associated vector is the shown in Figure 1.

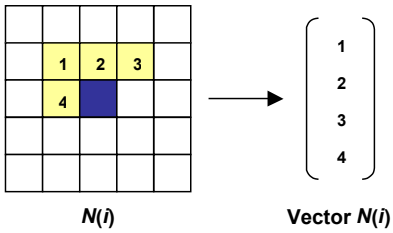


Figure 1. Example for the construction of the $N(i)$ vector.

The $N(i)$ are distributed forming clusters in the texture case [9]. Based on this assumption, Self-Organizing Maps can be used considering that their neurons are the *prototype vectors* of those clusters. Therefore, the set of all possible $N(i)$ may be used in the network training. Once the networks has been trained, is possible to make a mapping between the prototype vectors and a given gray level, dividing the training set into several subsets, where

each element of a given subset activates only one neuron of the SOM. Then, it can be calculated the mean gray value over each subset and associate it to the prototype vector under consideration. After the training process has been done, the SOM network is a data structure that allows perform quick and efficient searches. The gray value in the pixel $p$ at $T_O$ can be determined projecting $N(p)$ over the prototype vectors of the network, and then determining which one has the maximum projection. Finally, the gray level of this one will be associated to $T_O(p)$ (see Figure 2).

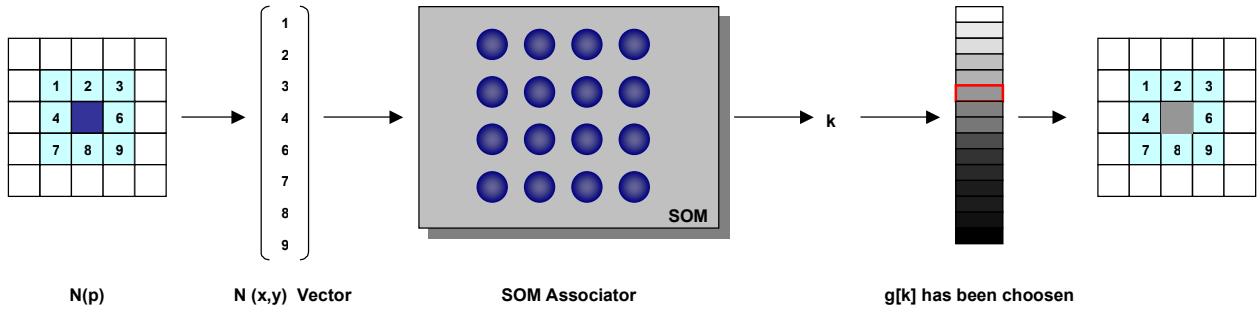


Figure 2. Block diagram for the calculation of $N(p)$.

### 3.4. Image Pyramid in Texture Synthesis

One of the critical factors that the original algorithm does not consider is the neighborhood size. If we choose the right one, the process might synthesize a good texture, but if that value does not allow capture the texture structure, the resulting image will not be valid. According to [9], neighborhood size has a strong relationship with texture quality. If we use a multi resolution representation, the computational cost for processing huge structures can be reduced.

Image Pyramids are one of the most efficient multi resolution representations available. Originally developed by Burt [1], image pyramid perform a subband decomposition of a given image. The defining feature of an image pyramid is that the basis/projection functions are dilated and translated copies one another (by a factor $2^j$ for a given integer $j$) [5].

Gaussian Pyramids are a particular kind of image pyramids, which consists on a low-pass set of images, where the size of the image decreases in a factor 2, from one level to the next, and a low-pass filter is convolved with level for avoiding aliasing problems. In our approach, images pyramid are used in different ways. A new texture is calculated level-by –level, from the lower to the highest, using the process described in3.3 .

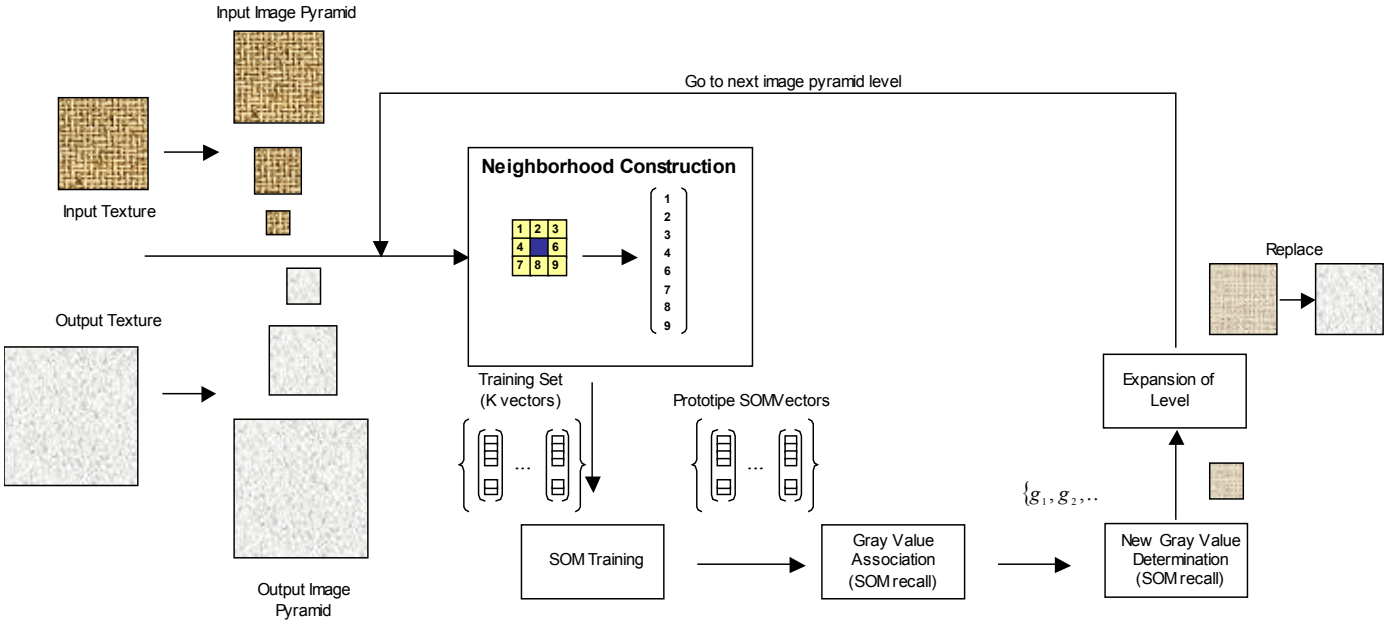


Figure 3. Block Diagram of the complete texture synthesis process.

In Figure 3 is present a block diagram representation of the algorithm. First, the Gaussian pyramid of the Texture Sample image and of a noise image (Texture Output), which will be forced to look like the texture sample, are calculated. Next, the data pre-processing stars with the training process of the SOM network. The *Neighborhood Construction* module builds a set of vectors that will be used in the SOM network training, and whose elements are the different neighborhood vectors (one for each pixel in the texture sample image). Then, each prototype vector of the SOM is associated to a gray level, using the process described in 3.3.

The second part of this sequence performs the texture synthesis process itself. For each pixel $p$ in a level of the image pyramid of the output image, its neighborhood $N(p)$ is projected onto the prototypes vectors of the SOM, and the unit with the largest response is selected. As every unit has a gray level associated (calculated in the *Gray Value Associator* module), the gray level of the triggered unit is assigned to $N(p)$.

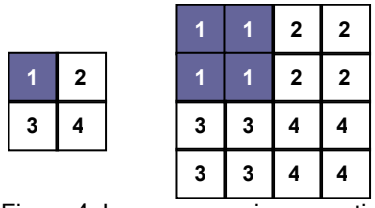

Figure 4. Image expansion operation.

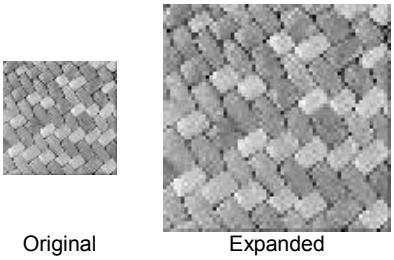

Figure 5. Natural texture expansion example.

After the processing of a level has been finished, the level image is expanded in the *Expansion Level* module, and the resulting image is used to initialize the next level in the $T_O$ Gaussian pyramid. The expansion operation makes grow the image size in a 2 factor. A visual representation of this operation is shown in Figure 4. The application of this operation over a natural texture image introduces some artifacts (see Figure 5), but it can be used as a good guess of the next image pyramid level.

The described method builds the new texture, level by level, and ends when the highest resolution level is reached.

### 3.5. Further Considerations

One of the goals of this algorithm is to use the multiresolution approach in the most useful way. Besides the use described in the previous section, the expansion operation can also be applied to expand the prototype vectors from one level to another. This action should be taken for improving the training process of the SOM network. When the training process begins, the prototype vectors in an $L$ level can be initialized with the expanded versions of the prototype vectors used in the $L+1$ level. In Figure 6 is shown an example of this procedure, that is somehow similar to the expansion process used in the image pyramid.

Another fact that is necessary to mention has relation with the shape and size of the neighborhood used in the algorithm. Considereting the selection of the shape, we have two main choices: use a causal neighborhood that considers only the pixel positions before the central one, or use a non-causal neighborhood, that considers the pixel positions before and after the central pixel. Our simulations were realized using both, and in the next section can be found a deeper development of this idea.

The use of image pyramids avoids an excessive increase of the neighborhood size for processing large elements in the image. However, it does not mean that if we use the same neighborhood size the resulting image will be a good texture. Instead, we have run our simulations increasing the neighborhood size from one level to the next, but in a controlled way. In Figure 6 the size of the neighborhood increase adding 2 to the previous value.

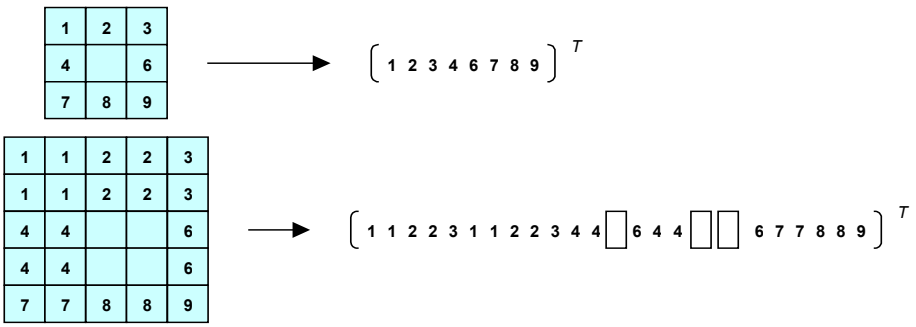

Figure 6. Expansion of the prototype-vectors in the SOM network.

A final aspect is that the number of units in the SOM network is not the same for all the image pyramid levels. Since the lower resolution levels have a small size, the cardinality of the training vector set is smaller also. Therefore, the number of network units (or prototype vectors) should be increase as the resolution level rises.

## 4. Preliminary Results

Till now, we have worked in the implementation of the algorithm. Although these simulations are not conclusive, because they did not cover a wide textures variety, they should be considered as good preliminary results.

In Figure 7 are presented three images pyramids. The first row corresponds to the original texture (Tex001), the second to the generated texture using a causal neighborhood and the third row to the generated textures using a non-causal one. Although in the lower resolution levels there is not much difference between the images, the neighborhood type makes a big difference at the higher levels.

It has to be noted that a new texture is constructed over a noise image and, because of that, local similarity with the original texture is good, but it can be improved.

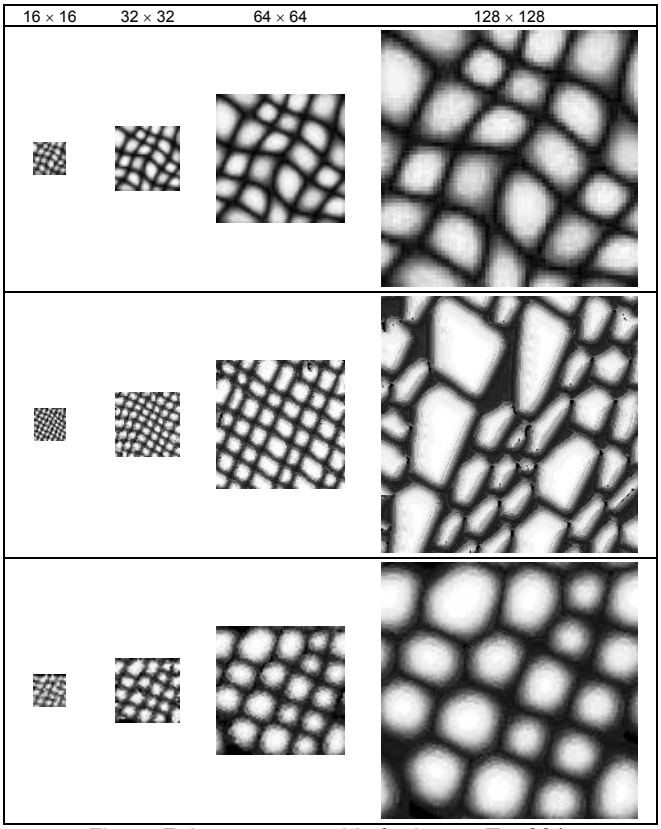


Figure 7. Images pyramids for image Tex001.

In order to accomplish the similarity requirement, it was decided to initialize the new texture not with white noise but with the same original texture, copied as many times as it is necessary (according to the desired image size). This action has an influence only in the lower level of the image pyramid of the output texture, due to the other level are computed applying the expansion operation from one level to the next.

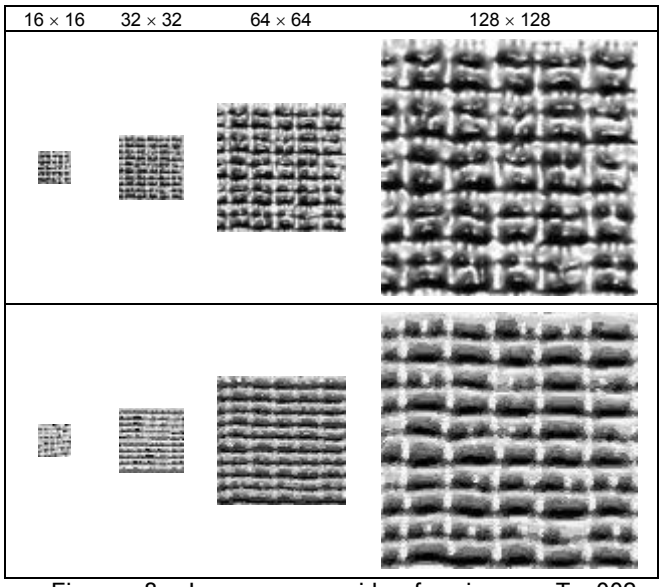


Figure 8. Images pyramids for image Tex002.

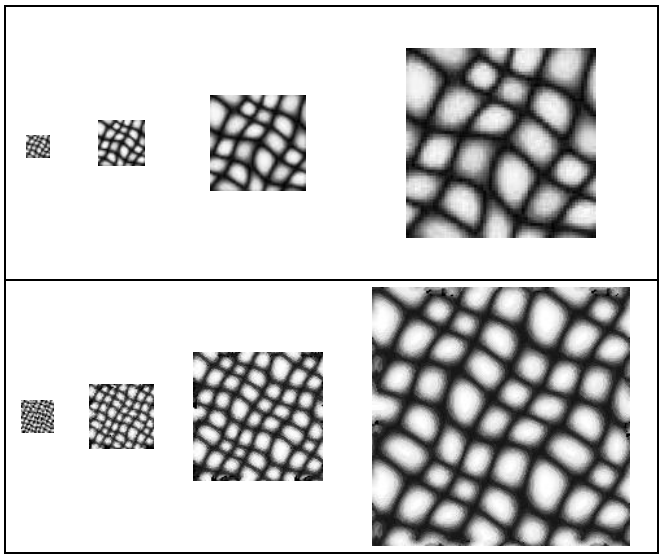

Figure 9. Images pyramids for image Tex001.
In this case the size of the new texture is greater than the original one.

In Figure 8 is presented the result of applying this modification over a quite regular texture (Tex002). The outcome is very impressive, because the model can emulate adequately the feature image, in a very good

manner. In Figure 9 it is shown the generation process of texture Tex001, but the size of the output texture is larger that the original one. Again, the algorithm not only reconstructs the part where both images have coincidence, but also it is capable to interpolate a new section of the image, not present in the original one.

## 5. Conclusions

In this article is proposed a new algorithm of texture synthesis, based in the use of image pyramids and self-organizing maps. This procedure was inspired by the proposed in [9], which uses a Markov Random Field as texture model, but avoids the explicit computation of its parameters.

The presented algorithm employees the image pyramid approach for processing patterns of different sizes in different resolution scales through image pyramid. The self-organizing map tool is used in the neighborhood match task, which can be thought as a nearest-point search problem.

The results that have been obtained are quite encouraging. The generated textures accomplish the similarity goal, and gives good results when the output image size is greater than the original texture one.

The proposed algorithm has several applications. It can be used in image replace applications, constrained texture synthesis, by name a few.At this time, the authors are working on the improvement of the computational aspects of the algorithm, for obtaining a processing time more suitable for on-line application.

## Acknowledgements

This research was supported by FONDECYT (Chile) under Project Number 1990595 and by the join "Program of Scientific Cooperation" of CONICYT (Chile) and BMBF (Germany).

## References

[1] P. Burt. "Fast Filter Transform for Image Processing". *Computer Graphics and Image Processing 16*, 1981, pp. 20 – 51.

[2] R. Chellappa, R. Kashyap, "Texture Synthesis Using 2D-Noncausal Autoregressive Models", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP - 33, No. 1, February 1985, pp. 194 - 203.

[3] G.R. Cross and A.K. Jain, "Markov Random Field Texture Models", *IEEE Transactions on Pattern and Machine Intelligence*, Vol. PAMI- 5, January 1983, pp. 25 – 39.

[4] M. Hassner and J. Sklansky, "The Use of Markov Random Field Models for Textures," *Compuing. Graphics Image Processing*, Vol. 12, April 1981, pp. 357 – 370.

[5] D.J. Heeger and J.R. Bergen. "Pyramid - Based Texture Analysis/Synthesis". *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, ACM SIGGRAPH, Addison Wesley, August 1995, pp. 229 – 238.

[6] F. Hering. "Modellbasierte Textureanalyse". Heidlberger Bildverarbeitungsforum: Textureanalyse, Germany, 1996, pp. 43 – 53.

[7] T. Kohonen. *Self Organizing Maps*. Second Edition, Springer-Verlag Berlin Heidelberg, 1997

[8] J.P. Lewis. "Texture Synthesis for Digital Painting". *Proceedings of SIGGRAPH '84* , Vol. 18, ACM SIGGRAPH, pp. 245 – 252, 1984.

[9] Wei, L. and Levoy, M. "Fast Texture Synthesis using Tree-structured Vector Quantization". of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, July 2000, pp. 479-488.