Patch-Based Texture Synthesis using Wavelets

Leandro Tonietto
ltonietto@unisinos.br

Marcelo Walter
marcelow@unisinos.br

Cláudio Rosito Jung
crjung@unisinos.br

Abstract

Patch-based texture synthesis builds a texture by joining together blocks of pixels – patches – of the original sample. Usually the best patches are selected among all possible using a L2 norm on the RGB or grayscale pixel values of boundary zones. The L2 metric provides the raw pixel-to-pixel difference, disregarding relevant image structures – such as edges – that are relevant in the human visual system and therefore on synthesis of new textures. We present a wavelet-based approach for selecting patches for patch-based texture synthesis. For each possible patch we compute the wavelet coefficients for the boundary region and pick the patch with the smallest error computed from the wavelet coefficients. We show that the use of wavelets as metric for selection of the best patches improves texture synthesis for samples which previous work fails, mainly textures with prominent aligned features.

1. Introduction

Texture mapping [4] is a powerful concept in Computer Graphics, where visual details are added to objects without the need to explicitly model them. The visual information is captured on an image called the texture map and transferred to the 3D object's surface. For many tasks real textures, as opposed to procedurally generated ones [8], are the best option for achieving high realism. One of the main problems with real images is their usually small resolution. Texture synthesis from samples is an excellent solution for building textures which are not only visually similar to the given sample but also can be built at user-defined resolutions. The advances in this area grew from pixel-based techniques [10, 28], to more recent patch-based techniques [17, 9], where the final texture is formed by joining together pieces or blocks of the original sample, with a RGB metric for selecting the best matches.

We propose here the selection of the best patches using a wavelet-based metric. Due to the wavelet properties we show that we can satisfactorily synthesize arbitrarily large textures where previous solutions fail, mainly maintaining spatial prominent features, as illustrated in Figure 1. In the image quilting result [11] (bottom right on the figure) there are discontinuities not present in our result.

2. Previous Work

The importance of texture synthesis has always been acknowledged in both Image Processing and Computer Graphics fields, with research going back as far as the late seventies [19]. There was a lot of research in late seventies and early eighties which have tried to generate textures either to validate texture models – mostly in Image Processing tasks – or simply to use the result in an application. Many approaches have used the idea of a sample as input information to create the result (see for instance [19, 22, 6, 13]). Until recently, despite progress, such techniques were either too slow to be of practical use or the results were not general enough to be useful [15, 7, 25]. The most recent techniques on this subject synthesize a new, arbitrarily-sized texture, by copying either pixels or blocks of pixels into the final texture. We divided this review section into three parts: one pixel at a time synthesis, patch-based synthesis and wavelets.

2.1 One pixel at a time synthesis

The work of Efros and Leung presented in 1999 [10] introduced a new simple way of looking at the texture synthesis problem by "growing" a texture one pixel at a time from an initial seed. The color of a given pixel is determined by scanning over square patches of the sample texture that are similar to the patch on the texture being generated. A random patch in the sample is selected among the few satisfying the similarity criterion. The similarity is measured with a $L2$ norm (sum of squared differences) on the RGB colors weighted by a Gaussian kernel. The original Efros and Leung's algorithm is slow and later extensions have im-

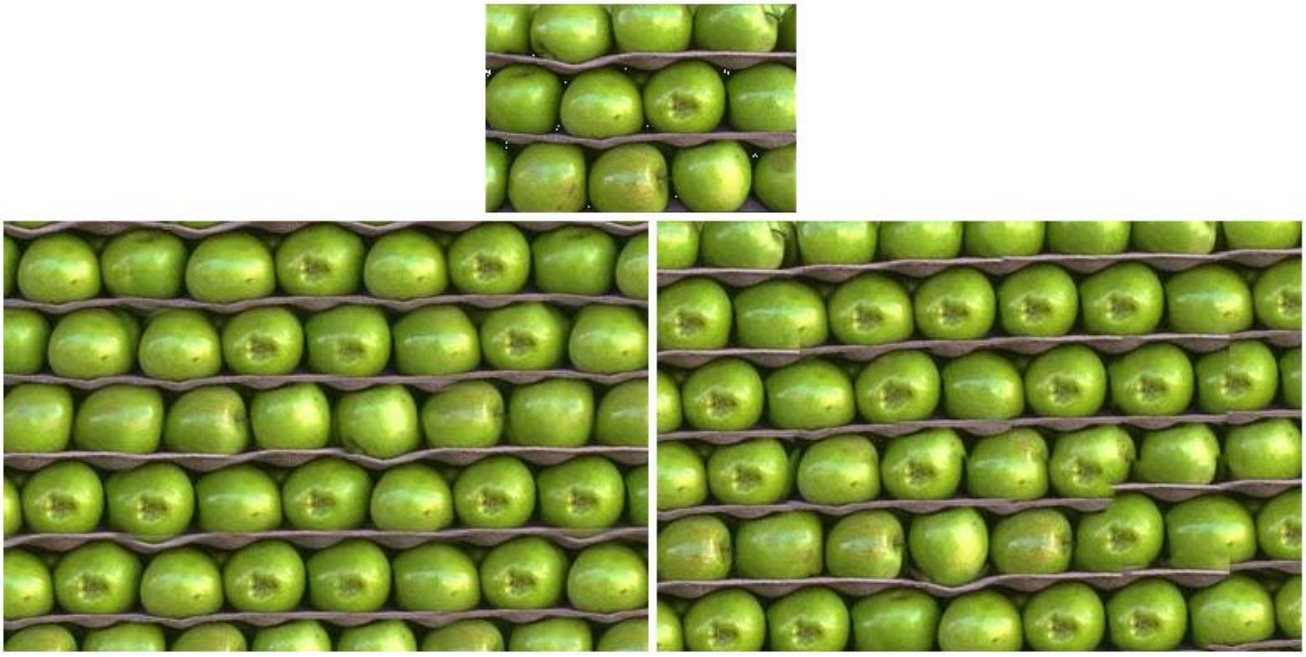

Figure 1. Example of Results – Sample (185x124), Ours (wavelet-based), result from [11] (both 384x256)

proved its performance, particularly the work of Wei and Levoy [29]. They have used a raster scan ordering to transform noise pixels into the result texture and have also improved the performance of the algorithm by using a multi-scale framework and tree-structured vector quantization. Their approach also minimizes the $L2$ norm in RGB space but without any weighting. Both approaches do not allow any kind of control of results. In 2001 Ashikhmin [2] improves on the Wey and Levoy technique by introducing two main contributions over prior work: first, his solution improves results on natural textures which mostly fail with the WL algorithm. Second, he introduces a second image, called the *target* image in the synthesis process. The final synthesis combines pixels from both the sample and the target, allowing nice effects such as writing words with the texture elements. A second approach towards local control for texture synthesis from samples was introduced by Tonietto and Walter [27] where the final texture is built from a collection of the same sample at different resolutions. This allowed synthesis of textures with local control over the size of texture elements.

More recently, Zhang and colleagues introduced an image-based texture synthesis method for rendering of *Progressively Variant Textures* (PVTs) [30] still on a pixel-at-a-time basis. Although not formally defined in the paper, the concept of PVT is an important one for texture synthesis, since it captures the class of textures where the texture elements vary in a progressive fashion, typical examples being mammalian fur patterns such as leopard skin. From an homogeneous texture sample they were able to synthesize a PVT. One main idea in their work was the notion of *texton masks*. A texton mask is a binary image marking prominent features or texture elements in the texture. In their synthesis algorithm, the texton masks prevent the disintegration of texture elements during synthesis.

2.2 Patch-based synthesis

Efros and Freeman introduced yet another way of synthesizing image-based textures by stitching together random blocks of the sample and modifying them in a consistent way [9]. They call the technique "image quilting". The idea improves dramatically on the one-pixel-at-a-time approach since it builds the texture at a much coarser scale while being able to keep high frequencies of the sample. The same idea of using patches from the sample to synthesize the result was explored by Liang *et al* [17]. In this work they were able to achieve real-time generation of large textures using special data structures and optimization techniques. Our work is an extension of Patch-Based Texture Synthesis.

Instead of using a RGB metric to pick the best patches, we propose a wavelet-based approach. A more detailed explanation of the Patch-Based Texture Synthesis is given in the next section.

2.3 Wavelets

Since wavelets were first introduced in the graphics community [14], they have come a long way and are an important tool in many graphics and image processing applications. The wavelets properties make them the tool of choice for dealing with either 1D or 2D signals with high discontinuities. The first work to use wavelets in a similar context to ours was a tool for querying image databases. The wavelets coefficients were used as a "signature" to find matches among a given sketch image and the ones stored in the database [16]. Another interesting use of wavelets for comparing images is the assessments of rendered images. Instead of judging results visually, the work of Gaddipatti, Machiraju, and Yagel [12] proposes a wavelet based metric to compare synthesized images.

There are several method based on wavelets for texture synthesis, analysis and classification [5, 23, 3, 24, 1].

Portilla and Simoncelli [23] proposed a statistical model for texture representation using an overcomplete complex wavelet transform. In their approach, a set of statistics computed on pairs of coefficients corresponding to basis functions at adjacent spatial locations, orientations and scales is used to represent a texture. They are also able to synthesize textures by randomly reproducing such statistics. This technique tends to fail for structured textures.

Bar-Joseph and colleagues [3] also worked with texture synthesis in the wavelet domain. In their approach, input textures are treated as sample signals generated by a stochastic process. A tree that represents the multiscale wavelet transform of the signal is computed, and new random trees generated by learning and sampling the conditional probabilities of the paths in the original tree are used for texture synthesis.

Wavelet representations have been also used in the context of texture classification. For example, Arivazhagan and Ganesan [1] described metrics to compare an input texture with several samples stored in a database, using wavelet statistical features and wavelet co-occurrence features.

Experimental results produced by these existing techniques indicate that wavelet analysis presents a large potential for texture synthesis and representation. However, approaches based on statistical simulation of wavelet coefficients typically have limitations for synthesizing more structured textures. Next, we describe the proposed algorithm, which explores the wavelet transform to synthesize both "random" and structured textures based on samples.

3. Our model

Our work is an extension of the basic algorithm presented for Patch-Based Texture Synthesis (PBTS) [17] and therefore we start this section by presenting an overview of this work.

3.1 Patch-Based Texture Synthesis

In Patch-Based Texture Synthesis, patches of the original sample are combined to form the final texture. The algorithm starts by randomly picking a patch B_0 to start the process. This patch is positioned at the bottom left corner of the output texture (Figure 2 (left)). The size of the patches w_B is user-defined and intuitively it should be the size of the main texture elements – or texels – present in the sample. For most textures using a patch of size between half and a quarter of the size of the original sample works well. For simplicity they are also restricted to square patches. The synthesis process follows by adding patches side-by-side and once a full row is completed the process continues for the row above and so on – Figure 2 (middle and right). There are three possible configurations for boundary zone matching, as illustrated in the same figure.

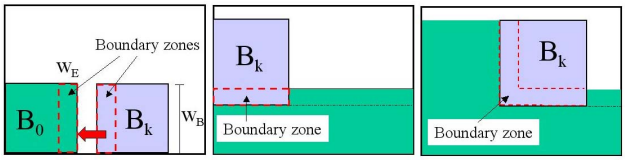

Figure 2. Illustration of patch-based texture synthesis adapted from [18]. Starting point (first configuration), second configuration for patch matching, and third configuration (L-shaped matching). Darker gray area represents texture already synthesized.

For each patch there is a boundary zone also with a user-defined width w_E. The optimal size of w_E depends on the texture being generated. It it is too small, it will not capture enough details. If it is too large it will negatively impact the algorithm's performance. As a balance they typically set w_E as $\frac{1}{6}$ of w_B.

The critical part of the algorithm is the selection of the next patch B_k to be pasted to the texture being constructed. As with many texture from sample techniques [10, 29], they use a RGB metric to compare patches and build a list of candidate patches which satisfy an error criterion at the border area. From this list a random patch is selected. To build this list, the input sample is searched for all possible patches. If there is no patch satisfying the condition, the algorithm picks the patch with the smallest distance.

More formally, given two texture patches I_1 and I_2 of the same size and shape, they match if

$$d(I_1, I_2) < \delta$$

where $d()$ represents the distance between the two patches and δ is a defined constant. This distance is computed only for the boundary zone E of patches as follows:

$$d(E_i, E_{i+1}) = \left[\frac{1}{A} \sum_{j=1}^{A} \left(p^j_{E_i} - p^j_{E_{i+1}} \right)^2 \right]^{1/2} \quad (1)$$

where A is the number of pixels in the boundary zone, and $p^j_{E_i}$ represent the values of the jth pixel in the E_i boundary zone. The pixel values can be either grayscale or RGB triplets, although in the paper there is no explicit reference to the metric used for their results. Once the patches are selected, there is a blending step to provide smooth transition among adjacent patches. This smoothing is performed with feathering as proposed by Szeliski and Shum [26].

The standard algorithm was optimized for real-time texture synthesis with an optimized kd-tree and a pyramid scheme, providing excellent timings.

3.2 Wavelet criterion

The wavelet transform (WT) is a mathematical tool that can be used to describe 1D or 2D signals (images) in multiple resolutions. A WT is obtained through a sequence of low-pass and high-pass filters, alternated with downsamplings [20]. The result of the WT is a downsampled smoothed signal, and several detail coefficients obtained at each downsampling, such that the resulting signal has the same size as the original one. In particular, detail coefficients are generated by signal transitions, and can be used to obtain a multiresolution representation of signal edges [21]. In other words, the WT produces a signal that encodes both information on the original signal values and its multiscale edges. Another interesting property of the WT is its *space-frequency locality*, that is, it has nice localization properties in both the spatial and the frequency domains.

We believe that using the $L2$ norm directly to compare pixels values, as proposed by some state-of-the-art techniques [17, 9], is not the best approach for patch-based texture synthesis. In fact, if we just compute raw pixel-to-pixel differences, we may disregard important image structures (such as edges) that are relevant in the human visual system. On the other hand, wavelet coefficients encode both information on the original pixels (a smoothed and downsampled version) and multiscale edge information (detail coefficients). Furthermore, the pyramidal algorithm proposed by Mallat [20] allows a very fast computation of the WT.

Our main contribution in this paper is to change the metric used to compute the distance between the patches by a wavelet-based one. We replace equation 1 by the following:

$$d(E_i, E_{i+1}) = \sum_{\Psi=R,G,B} \left[\sum_{j=1}^{A} \left(c^j_{E_i} - c^j_{E_{i+1}} \right)^2_{\Psi} \right] \quad (2)$$

where A is the number of pixels in the boundary zone, and $c^j_{E_i}$ represent the values of the jth wavelet coefficient in the E_i boundary zone.

We search among all possible patches for the one that minimizes the distance computed according to the equation above. For smoothing out the transition between adjacent patches we use a weighted interpolation among the two adjacent patches. For each pixel $c = 0 \ldots w_E - 1$ on a given row of two adjacent patches E_i and E_{i+1}, the final color p is given by $p = (1 - \alpha) \times p_i + \alpha \times p_{i+1}$ with $\alpha = \frac{c}{w_E - 1}$.

For the wavelet computation we implemented both 1D and 2D transforms for the Haar basis. For the 1D transform we compute the WT until the low pass component of the texture in the boundary zone is downsampled to a 1×4 signal and for the 2D transform we downsample to a 4×4 image.

We perform a raster scan ordering for pixel traversal for patches in the first and second configurations of composition. For the L-shaped configuration we implemented a special traversal illustrated in Figure 4. The motivation for this traversal is to maintain a balance between the vertical and horizontal details of the boundary zone when using 2D WT. For the 1D transform we perform a raster scan ordering on this final rectangular arrangement. We have not investigated yet the effect on the results of other schemes for traversal, but we expect that for anisotropic textures traversal ordering may play a role on the final results.

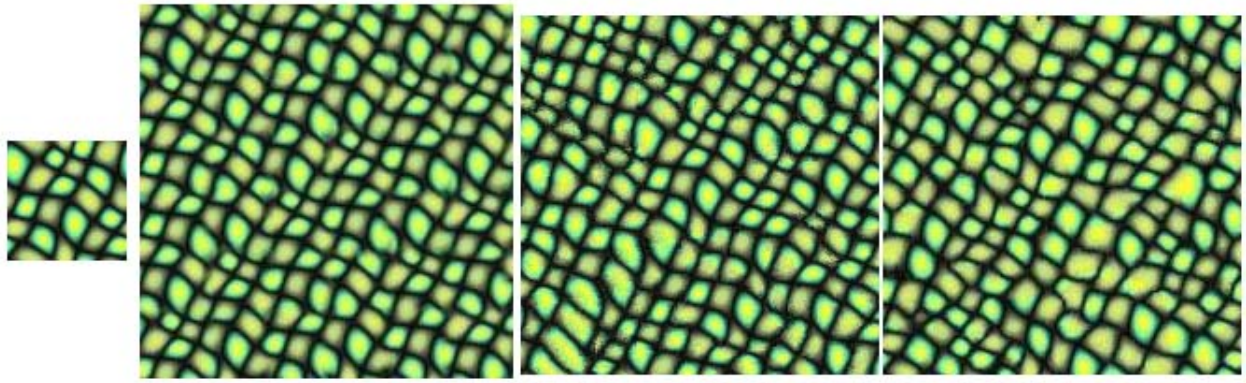

Figure 3. Example of Results. Sample, wavelet-based, result from [10],result from [29]

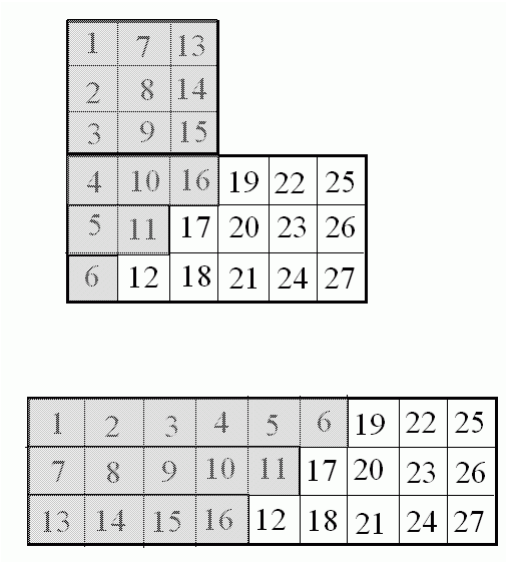

Figure 4. Traversal of pixels for L-shaped configuration. On the top the original L-shaped configuration. On the bottom the new arrangement of pixels suitable for traversal.

4. Results

In this section we present a few results of textures generated with our approach. First we present in Figure 3 our result for one standard texture that has been used as a test case for many texture from sample synthesis. We can see that the algorithm works well for non-structured textures such as this. All parameters used are given in Table 1.

In Figure 6 we can see that our solution correctly generates the cracker holes evenly spaced, whereas in the result by Liang *et al* [17] these are not preserved. We also show in this figure the effect of changing the size of the boundary zone w_E. We can see that the wavelet transform needs a boundary zone large enough to be able to capture the features of the texture. We have not investigated an automatic way of determining the proper size for w_E.

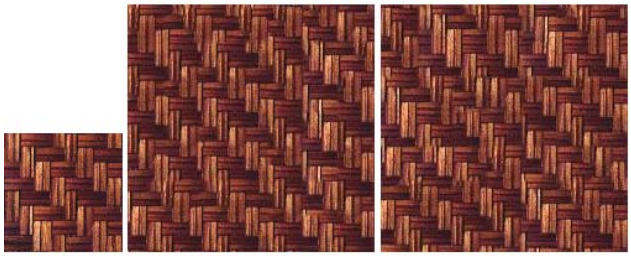

Figure 5. 1D versus 2D wavelets

The synthesis time for our results was very high for some cases (from a few seconds to even one hour for some results on a Pentium IV machine with 512MB of RAM). One reason is that the wavelet computation is more expensive than the $L2$ norm, but the main reason is that we have not yet implemented any optimizations as presented in the original patch-based work. This was left for future work. One interesting conclusion of our work is that 1D wavelet computations were good enough to generate our results. We illustrate in Figure 5 that there is no noticeable difference between the two results when using 1D and 2D wavelet computation.

5. Conclusions

We have presented a wavelet based approach for selecting the best patches in patch-based texture synthesis. The final output texture is constructed from blocks

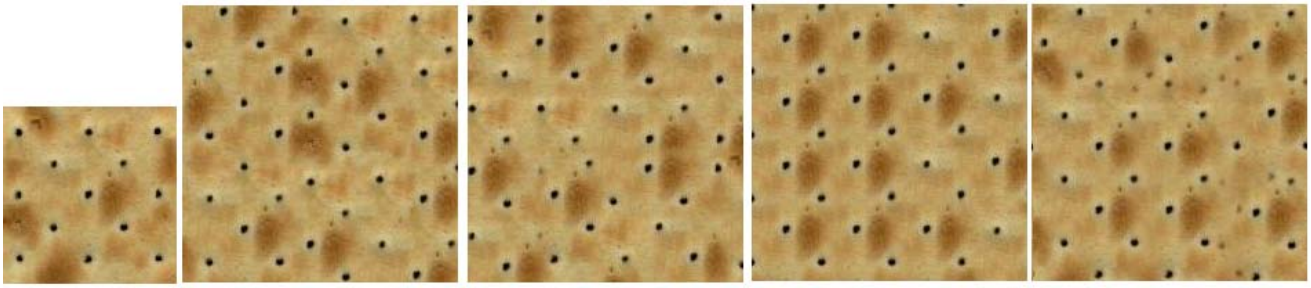

Figure 6. Changing the size of the boundary zone w_E. Sample, wavelet-based increasing w_E from left to right. Last is a result from [17]. We can see that in the last wavelet-based result the holes are correctly aligned.

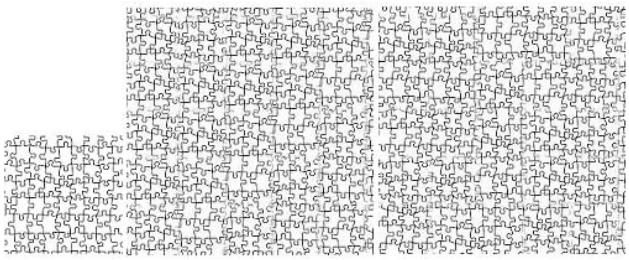

Figure 7. Gray versus RGB. Sample, result computed with gray values and result computed with the RGB values.

Figure	w_B	w_E
1	48	16
3	24	8
5	24	4
6(a)	32	8
6(b)	32	12
6(c)	32	16
7	32	6

Table 1. Values for parameters of results presented.

or patches of the original sample which have the smaller difference in the wavelet coefficients computed for the boundary zone. Our results show that this criteria improves on previous results for textures which have regular noticeable features. There are many avenues left for further research on this topic. Although we have implemented the Haar wavelet basis, we believe that further investigations could explore other wavelet basis. Also, for the wavelet computation we could explore alternative ways of visiting the pixels in the boundary zones. Finally, there are some textures for which the distance metric could be computed using the grayscale values of the sample. One such example is given in Figure 7 In this case the RGB information will not improve the results. We believe that this possibility might be true also for other textures which are not monochromatic such as this example.

Acknowledgments

We acknowledge the partial financial support from CAPES. We also thank Baining Guo and Ying-Qing Xu for kindly running their patch-based sampling algorithm on the cracker texture (Figure 6(last)).

References

[1] S. Arivazhagan and L. Ganesan. Texture classification using wavelet transform. *Pattern Recognition Letters*, 24(9-10):1513–1521, June 2003.

[2] M. Ashikhmin. Synthesizing natural textures. In *Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[3] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135, 2001.

[4] E. E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Ph.D. thesis, University of Utah, December 1974.

[5] T. Chang and C. Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Transactions on Image Processing*, 2(4):429–441, October 1993.

[6] G. Cross and A. K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):25–39, Jan. 1983.

[7] J. S. de Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 361–368. ACM SIGGRAPH, Addison Wesley, Aug. 1997. ISBN 0-89791-896-7.

[8] D. S. Ebert, F., K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*, volume 1. Morgan Kaufmann Publishers, third edition, December 2002.

[9] A. Efros and W. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, pages 341–346, August 2001. ISBN 1-58113-292-1.

[10] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, volume 2, pages 1033–1038, 1999.

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

[12] A. Gaddipatti, R. Machiraju, and R. Yagel. Steering image generation with wavelet based perceptual metric. *Computer Graphics Forum*, 16(3):241–252, Aug. 1997.

[13] A. Gagalowicz and S. Ma. Model driven synthesis of natural textures for 3-D scenes. In *Eurographics '85*, pages 91–108. 1985.

[14] S. J. Gortler, P. Schröder, M. F. Cohen, and P. Hanrahan. Wavelet radiosity. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 221–230, Aug. 1993.

[15] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In R. Cook, editor, *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 229–238, Aug. 1995.

[16] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 277–286, Aug. 1995.

[17] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, July 2001. ISSN 0730-0301.

[18] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (TOG)*, 20(3):127–150, 2001.

[19] S. Lu and K. Fu. A syntactic approach to texture analysis. *Computer Graphics and Image Processing*, 7(3):303–30, June 1978.

[20] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[21] S. G. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710–732, 1992.

[22] J. Monne, F. Schmitt, and D. Massaloux. Bidimensional texture synthesis by markov chains. *Computer Graphics and Image Processing*, 17(1):1–23, Sept. 1981.

[23] J. Portilla and E. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, October 2000.

[24] C. Pun and M. Lee. Log-polar wavelet energy signatures for rotation and scale invariant texture classification. *IEEE Transactions on Pattern Alalysis and Machine Intelligence*, 25(5):590–603, May 2003.

[25] E. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *Fifth IEEE International Conf on Image Processing*, volume I, pages 62–66, Chicago, Illinois, October 1998. IEEE Computer Society.

[26] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258. ACM Press/Addison-Wesley Publishing Co., 1997.

[27] L. Tonietto and M. Walter. Towards local control for image-based texture synthesis. In L. M. G. Gonçalves and S. R. Musse, editors, *Proceedings of SIBGRAPI 2002*, pages 252–258. SBC - Brazilian Computer Society, IEEE Computer Society, October 2002.

[28] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. *Proceedings of SIGGRAPH 2000*, pages 479–488, July 2000. ISBN 1-58113-208-5.

[29] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 479–488, July 2000.

[30] J. Zhang, K. Zhou, L. Velho, B. Guo, and H.-Y. Shum. Synthesis of progressively variant textures on arbitrary surfaces. *ACM Transactions on Graphics*, 22(3):295–302, July 2003.