Towards Local Control for Image-Based Texture Synthesis

LEANDRO TONIETTO, MARCELO WALTER

PIPCA - Mestrado em Computação Aplicada
UNISINOS - Centro de Ciências Exatas e Tecnológicas
tonietto/marcelow@exatas.unisinos.br

Abstract. New advances in image based texture synthesis techniques allow the generation of arbitrarily sized textures based on a small sample. The generated textures are perceived as very similar to the given sample. One main drawback of these techniques, however, is that the synthesized result cannot be locally controlled, that is, we are able to synthesize a larger version of the sample but without much variation. We present in this paper a technique which improves on current fast texture synthesis techniques by allowing local control over the result. By local control we mean a final texture that is still perceived as a whole but presents variations in size of the basic elements. Our solution generates the final texture from a small collection of the same sample at different resolutions, adequately interpolated. We illustrate our results with some examples, including natural textures such as animal coat patterns, which exhibit local variations that can be adequately captured by our algorithm.

1 Introduction

Recently, new techniques have been introduced which allow the synthesis of textures by constructing larger textures from a small sample [7, 19, 6]. The larger texture has the same visual appeal as the smaller texture, and can be generated relatively fast. This availability of tileable textures with arbitrary size has many applications in image processing and computer graphics tasks.

In the context of Texture Mapping [2], for instance, these image based texture synthesis techniques have addressed one of the main drawbacks of real-world textures used as texture maps, which is their usual low resolution. Another class of problems in the context of texture mapping arises in applications where the texture has many local variations. Real-world scanned images cannot capture the full richness of real textures due to limitations in the scanning process. To illustrate this problem, imagine trying to scan a full leopard skin and use it as a texture. The skin has local variations which are hard to capture on a single texture image.

In this context a natural question arises: can we use these recently introduced image based texture synthesis techniques to generate from a small sample textures with local variations, such as a full leopard skin? As introduced, the techniques do not have a mechanism for allowing *local control*. By local control we mean to obtain a final texture that is perceived as a whole but presents small variations in size, color, and shape of the basic elements. These type of variations occur naturally in many textures. In Figure 1 we illustrate the case with a leopard. Overall, we perceive the leopard skin as a whole but locally it has variations which add richness to the overall effect.

We introduce in this paper an extension in the texture synthesis algorithms that grow the texture one-pixel-at-a-time. Our extension provides a mechanism for adding local variation in the size of the basic elements of the generated texture. In general, the small sample will not contain all relevant information needed to synthesize a full texture with all local variations. Our approach modifies the small sample in a controlled way to allow for local variations in the final result. Our solution brings us closer to being able to generate the full leopard skin from a small sample, but does not completely solve it yet, as we explain in the conclusions section.

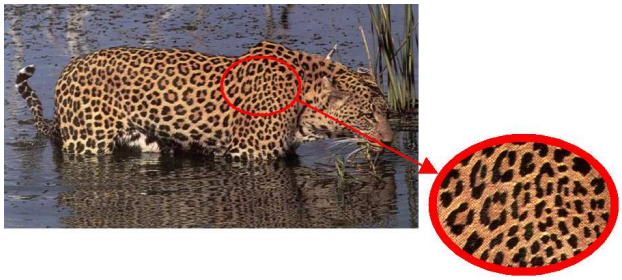


Figure 1: Local Variations on Natural Textures. Notice how the basic round elements decrease in size and eventually become individual dots.

In Figure 2 we illustrate the basic idea of our work. On *(a)* we have a small sample which is used to generate the result on *(b)*. The result has the same overall visual appeal as the sample on the left, but the basic round green element decreases in size towards the bottom of the image.

This possibility of textures with local variations has a distinct advantage in texture mapping tasks. Texture mapping is the main technique for adding rich visual details to

virtual objects without simulating them explicitly. The idea is simple: instead of modeling the visual details, they are captured in a image and this image is then mapped to the surface of the object. These visual details are referred to as the texture of the object and express the objects surface characteristics. In practice, one of the main problems of using texture mapping is that good texture maps are hard to come by. They are usually small real-world images scanned in. Our work allows us to generate textures which are hard to obtain from scanning in real-world materials.

(a) (b)

Figure 2: A generated texture with local variations. (a) Sample (b) Our result. The sample has 32x32 pixels and the result has 150x300 pixels. Notice the smooth decrease in size of the basic round green element towards the bottom of the image.

2 Related Work

Texture synthesis is an old research subject in both Image Processing and Computer Graphics fields, with research going back as far as the late seventies [12]. The many different models and approaches have always tried to generate textures either to validate texture models (mostly in Image Processing tasks) or simply to use the result in some application. The idea of using a sample as input information to create the result has always been present (see for instance [12, 13, 3, 8]). 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 [9, 4, 15].

The work of Efros and Leung presented in 1999 [7] introduced a new simple way of looking at this 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) weighted by a Gaussian kernel. The original Efros and Leung's algorithm is slow and recent extensions have improved its performance, particularly the work of Wei and Levoy [19]. 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 vector quantization. Their approach also minimizes the $L2$ norm in RGB space but without any weighting.

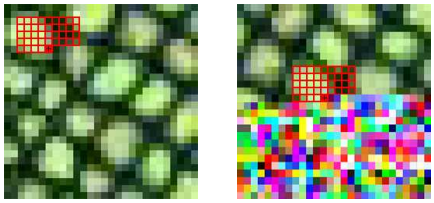
More recently, Efros and Freeman introduced yet another way of synthesizing image-based textures by stitching together random blocks of the sample and modifying them in a consistent way [6]. They call the technique "image quilting". The idea improves dramatically on the one-pixel-at-a-time approach since it builds the texture at a much coarser resolution 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* [11]. In this work they were able to achieve real-time generation of large textures using special data structures and optimization techniques.

Finally, there are a few alternative solutions to texture synthesis from samples which can be used. One of them is procedural texture synthesis, artificially generating the needed textures using a model. Of course, procedural models are mostly targeted to a particular effect. For instance, in the context of mammalian coat patterns biological inspiration is used to drive the procedure [16, 22, 18]. For these type of textures the control of desired results is always a potential drawback. A complete overview of the current state-of-the-art in procedural texture synthesis techniques is out of the scope for this paper. We direct the reader to the book by Ebert *et al* [5] for a good overview of procedural texture synthesis techniques.

All the above recent techniques are very good at synthesizing arbitrarily sized textures that are perceived as very similar to the given sample. However, they do not address the synthesis of textures with local variations as we are addressing here. They do have the potential to be explored in this way, and that is our main contribution in this work.

3 Algorithm

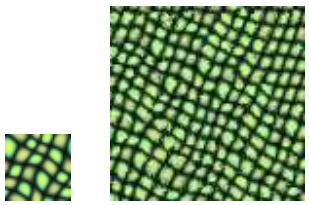
Our algorithm is an extension on the basic algorithm of Wey and Levoy [19] and therefore we start this section with a brief overview of their approach. Figure 3 illustrates the general idea.



(a) (b)

Figure 3: Illustration of Wey and Levoy's synthesis algorithm. (a) Sample being scanned for the best match (b) Noise pixels being transformed into the final texture. The images have been enlarged to illustrate the algorithm.

The algorithm starts with a small texture sample as input (Figure 3 (a)). This sample will be used to contruct a texture with the same overall visual appeal as the sample. Even though it is not strictly necessary, the synthesized texture is usually larger than the sample. In order to do this a noise texture is transformed in a raster scan ordering one pixel-at-a-time (Figure 3 (b)). Starting with the pixel at the upper leftmost corner, the algorithm searches on the sample for the best match measured with a $L2$ norm in RGB space. The search uses a user-defined neighborhood size. The sample is searched using continuous boundary conditions (toroidal). In Figure 4 we illustrate one result of Wey and Levoy's algorithm.



(a) (b)

Figure 4: Wey and Levoy's texture synthesis. (a) 32x32 sample. (b) 100x100 synthesized result.

The idea is simple and easy to implement. The original paper also introduced mechanisms to accelerate the computation. As we can see from the result, Wey and Levoy's algorithm is very good at synthesizing new textures that are visually indistinguishable from the sample. This is the good and the bad news at the same time. If one wishes a texture sligthly different, the algorithm does not provide any mechanism for that.

The basic idea of our algorithm is to build the result from a small collection of the same sample at lower resolutions. The idea of using the same sample at different resolutions dates back to 1983 with the introduction of MIP maps [21], although mip maps are used on a different context (filtering of texture maps) and uses linear interpolation between levels.

The final texture in our case will be a collection of texture "patches", each of which is synthesized using Wey and Levoy's algorithm but using as input the same sample at different resolutions. In order to have a smooth transition between patches we define a transition area.

(a) (b) (c) (d)

Figure 5: Input information for our algorithm. The multiresolution sample: (a) Original 32x32 sample (b) 70% (c) 50% (d) 40%

In Figure 5 we illustrate one possible input for our algorithm. We show a given sample at the original resolution and three versions of this sample at lower resolutions. The total amount of reduction is controlled by the user as a parameter of the algorithm (in this case 40%).
The main parameters of our algorithm are:

- Size of output texture image - I_s
  Size of the desired texture measured in pixels.
- Number of patches - L
  This parameter controls how many patches will be used to build the result. We number the patches L_0 (highest resolution), L_1 and so on.
- Size of the transition area between patches - t
  Between two patches, we will have a transition area specified by a given number of scan lines.
- Size of the neighborhood around a given pixel p - $N(p)$
  This is a parameter already present in Wei and Levoy's synthesis algorithm, and has the same meaning in our context, that is, it specifies how many pixels are used for finding the best match on the sample. The intuition behind this parameter is that each texture sample has an ideal neighborhood size which adequately captures the size of the texture elements present in the sample.
- The percentage of size reduction - r
  Specifies how much smaller than the original sample it will be the lowest resolution sample. For instance, with $L = 3$ and $r = 0.5$ we will have L_0 at the original size, L_1 with a resolution 75% smaller than L_0 and L_2 with a resolution 50% smaller than L_0.

Figure 6 illustrates schematically our approach. In this figure we have 3 patches L_0, L_1, and L_2. The coarser patch is L_0. For each patch we associate a sample (for instance samples (a), (b), and (c) of figure 5).

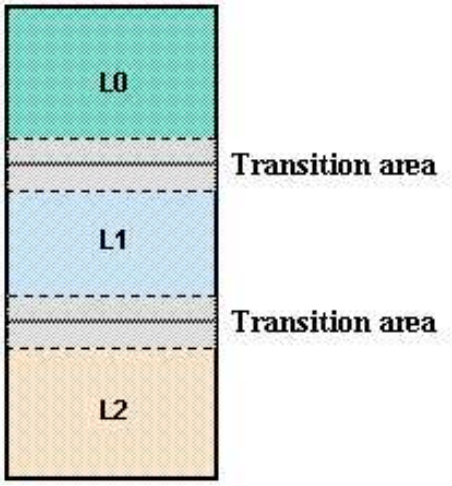


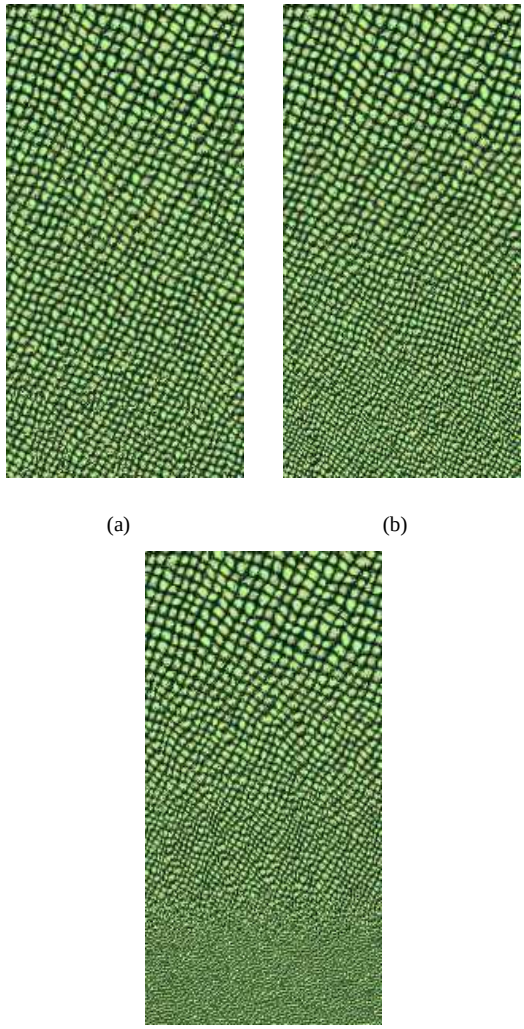
Figure 6: The different patches used for generating a texture, in this case three.

Our algorithm follows the single resolution synthesis procedure of Wei and Levoy by incrementally modifying noise pixels into the final texture on a raster scan ordering. The difference in our case is that depending on where we are generating the next pixel (that is, which scanline we are in), we will look for the best match in the corresponding sample.

Between patches we define a transition area. In this transition area we use a different approach to synthesize the pixel color. The goal is to have a visually-smooth transition such that the user does not perceive it. The pixel color will be choosen randomly from either one of the two adjacent samples, controlled by a probability. The probability of which sample we will look for the best match is determined by which scanline we are in. If we are computing the pixel color in a transition area between patches A and B, for instance, pixels in a scanline closer to patch A will have a higher probability of being chosen from the sample associated with A, but they can also be chosen from the sample associated with B. Pixels in scanlines middle way through the transition area will be chosen roughly half from the sample associated with patch A and half from the sample associated with patch B, and so on.

This random transition, controlled by a probability determined by which scanline of the transition area we are in, seems to work well in practice, as illustrated in the results. This was a key insight, since we tried other possibilities (such as linear interpolating the colors of the two samples) and the results were not good.

In the next two figures we show how the parameters change the final result. In Figure 7 we illustrate the effect of decreasing r (maximum reduction size) and in Figure 8 we illustrate the effect of modifying the size of the transition

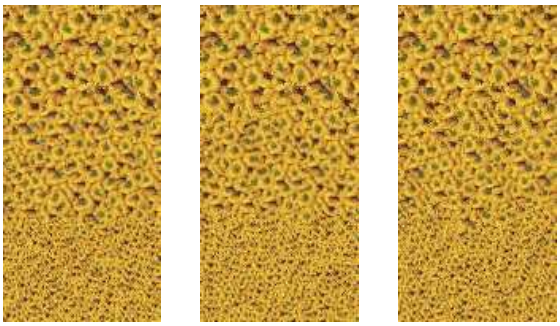


(a) (b)

(c)

Figure 7: Changing the maximum size reduction. (a) $r = 0.7$, (b) $r = 0.5$ and (c) $r = 0.3$. For (a),(b), and (c) $t = 40$ pixels, $N(p) = 9$, $I_s = 150x300$ pixels, $L = 4$.

area, from 0 to 20 to 40 scanlines. As expected, the increase in size of the transition area smoothes out the overall result. With $t = 0$ we can notice the border between individual patches.



(a) (b) (c)

Figure 8: Changing the size of the transition area. (a) $t = 0$ scanlines, (b) $t = 20$ scanlines and (c) $t = 40$ scanlines. For (a), (b), and (c) $r = 0.5$, $N(p) = 13$, $L = 3$, I_s = 100x200.

Another important implementation point is that, as we move from sample to sample, we are decreasing the parameter $N(p)$ (size of neighborhood) accordingly to the percentage of size reduction r.

4 Results

In this section we present a few results generated with our algorithm. Although the key idea is very simple, we were able to generate interesting results which we believe would be useful in the context of image processing and computer graphics tasks. In general, the synthesized textures presented a smooth transition between patches. Table 1 illustrates our main results. The many samples were taken from a few sites (for instance [10, 14, 1]). We have used both structured and non-structured texture samples. We considered all results "believable" in the sense that we do not perceive discontinuities, with the exception of the brick wall texture. Although the size of the bricks decreases gradually, the random transition did not work well in this case. Maybe alternative "mixing" of patches would give us better results.

Our system was implemented in Java and even though we are using the single resolution synthesis procedure of Wei and Levoy's, it is reasonably fast. The textures took from a few minutes to about 30 minutes to generate, depending basically on the value for $N(p)$ (size of the neighborhood) and the original size of the sample (on an AMD Athlon 1Ghz machine with $128Mb$ RAM).

5 Conclusions

We introduced in this paper a mechanism to allow the synthesis of arbitrarily sized textures from a small sample, and the generated texture exhibits local variations in size of the texture elements. Our algorithm is an extension on the work of Wei and Levoy [19] and generates the final texture from a small collection of the same sample at lower resolutions. Our results show that this simple idea introduces variations on the generated result which are useful in many applications.

In our current implementation the only possible variation is in the size of the basic elements which define the texture. We are currently investigating other possibilities for controlling the final result and would like to be able to control color as well. The results we have shown used patches arranged in a rectangular fashion with transition areas also rectangular, varying from top to bottom. Of course this is not the only possibility. We are currently investigating the possibilities of other arrangements as well. In the case of simulating a leopard skin, for instance, one would have to control the synthesis procedure over a possibly arbitrary domain. The two difficulties here are how does one define where each patch goes and how to adequately compute a transition between them. As presented, our technique is not yet able to synthesize a full leopard skin from samples. Our results do not exhibit any variation on the shape of the texture elements (as illustrated in the real leopard texture of Figure 1) but we could envisage a system where a given sample is modified in the synthesis procedure accordingly to some rules. We believe our system is a first step towards this more ambitious goal.

In the context of texturing objects our goal is to generate the texture directly on the surface of the object (as done in [20, 17]) but controlling it using the geometric attributes of the shape, such as curvature. For natural objects such as patterned animals, there is a correlation between local variations in the fur and the underlying body of the animal. Intelligent texture synthesis should use this correlation to ease texturing tasks.

References

[1] Absolute Background Textures Archive. http://www.grsites.com/textures/.

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

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

[4] J. S. de Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images.

In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 361–368. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[5] D. Ebert et al. *Texturing and Modeling: a Procedural Approach*. Academic Press, 1994.

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

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

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

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

[10] MIT Media Lab. Vision texture. http://www-white.media.mit.edu/vismod/imagery/-VisionTexture/vistex.html.

[11] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, July 2001. ISSN 0730-0301.

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

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

[14] J. Portilla and E. Simoncelli. Representation and synthesis of visual texture. http://www.cns.nyu.edu/ lcv/-texture/.

[15] 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.

[16] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH 91 Conference Proceedings)*, volume 25, pages 289–298. Addison-Wesley, July 1991. ISBN 0-201-56291-X.

[17] G. Turk. Texture synthesis on surfaces. *Proceedings of SIGGRAPH 2001*, pages 347–354, August 2001. ISBN 1-58113-292-1.

[18] Marcelo Walter, Alain Fournier, and Daniel Menevaux. Integrating shape and pattern in mammalian models. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 317–326. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[19] Li-Yi 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.

[20] Li-Yi Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. *Proceedings of SIGGRAPH 2001*, pages 355–360, August 2001. ISBN 1-58113-292-1.

[21] L. Williams. Pyramidal parametrics. In *Computer Graphics (Proceedings of SIGGRAPH 83)*, volume 17, pages 1–11, Detroit, Michigan, July 1983.

[22] A. Witkin and M. Kass. Reaction-diffusion textures. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH 91 Conference Proceedings)*, volume 25, pages 299–308. Addison-Wesley, July 1991. ISBN 0-201-56291-X.

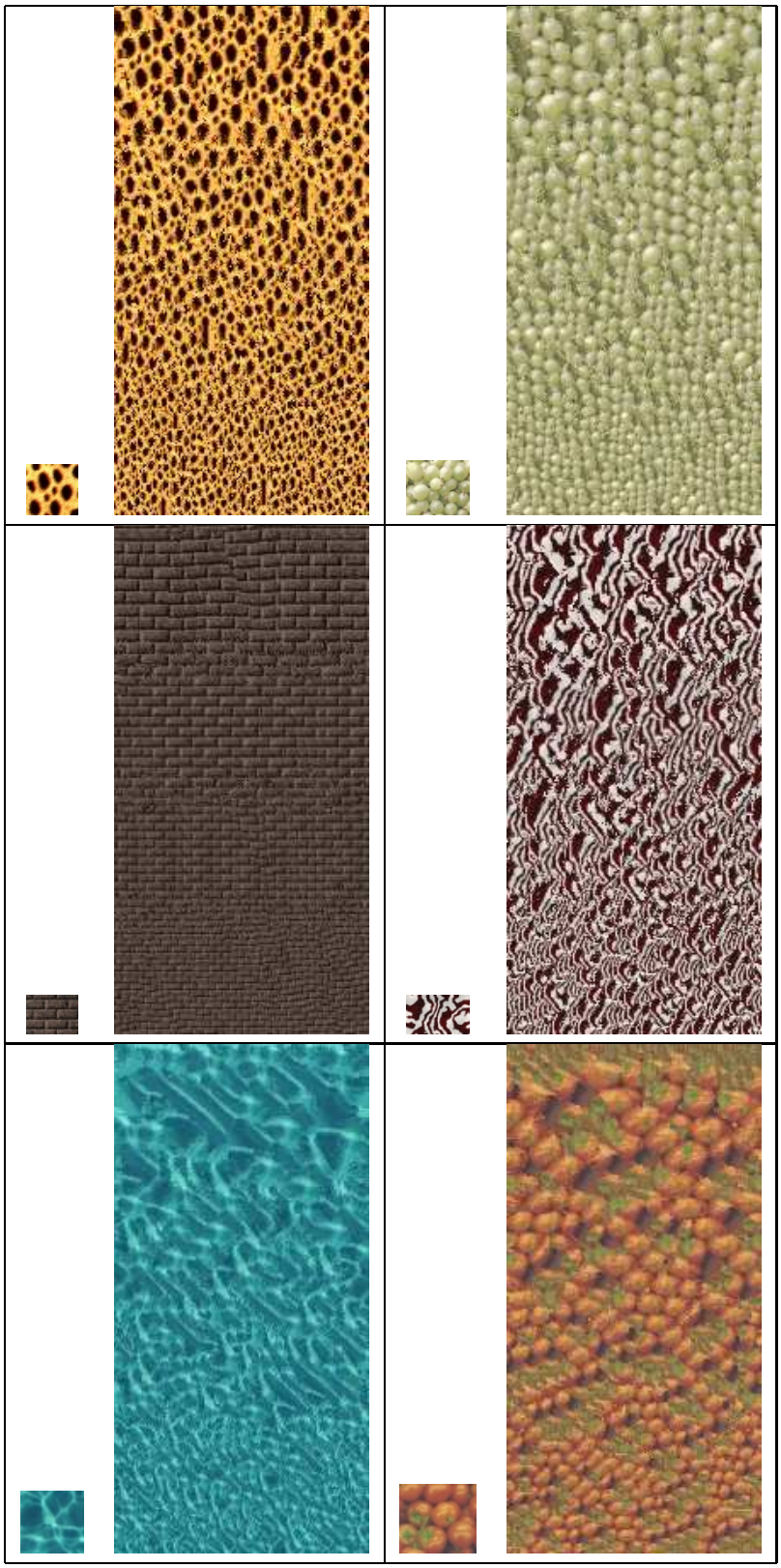


Table 1: Texture synthesis results. For all results I_s=150x300, $L = 4$, $r = 0.5$, and $t = 40$. First row: $N(p) = 13$; $N(p) = 15$; Second row: $N(p) = 17$; $N(p) = 19$; Third row: $N(p) = 13$; $N(p) = 17$;