Markov Information Propagation for Texture Synthesis

Guodong Guo
Charles R. Dyer

Technical Report #1446

October 2002

Markov Information Propagation for Texture Synthesis

Guodong Guo and Charles R. Dyer
Computer Sciences Department
University of Wisconsin-Madison

Abstract

A patch-based approach called Markov information propagation (MIP) is introduced for texture synthesis. The method is fast because it uses a simple horizontal and vertical patch filling process that preserves local structural similarity between the input texture and the synthesized texture. Results on both artificial and natural textures are presented and compared with some earlier methods.

Keywords: *Texture synthesis, Markov information propagation, patch-based approach.*

1 Introduction

Texture analysis has been an active research area in both psychology and computer vision for forty years, yet it is still difficult to accurately represent a wide variety of natural textures. Because of this, realistic texture synthesis using a synthesis-by-statistical-analysis approach has had limited success to date.

For example, Heeger and Bergen [5] analyzed texture in terms of histograms of filtered images at multiple scales and orientations, and texture synthesis was accomplished by matching the histograms of the to-be-synthesized texture with the input sample. This method works well for stochastic textures, but fails for more structured textures. Zhu *et al.* [11] defined a FRAME model, which incorporated the filters and marginal histograms with a Markov Random Field model [2], and learned the model parameters using an entropy criterion. For synthesis, a stochastic sampling method was used, equipped with the learned distribution parameters. De Bonet [1] used a multi-resolution filtering approach to analyze texture images, and then matched the joint probabilities for synthesis. Portilla and Simoncelli [8] also used a filtering approach to characterize texture images combined with a complicated optimization procedure.

It is generally difficult for synthesis-by-analysis methods to synthesize structured textures. So, instead, Efros and Leung [3] assumed a Markov Random Field model without any parameters, and synthesized a texture by comparing the local structural similarity between the input texture and the synthesized texture. Inspired by their work, many other direct synthesis methods have been proposed recently. For example, Wei and Levoy [9] used a tree-structured vector quantization approach to accelerate the synthesis process. These direct synthesis methods represent a structural approach to texture synthesis because they compare texture structures directly and explicitly for the purpose of synthesis, without any statistical modelling. Fig. 1 lists some representative examples of these two major approaches.

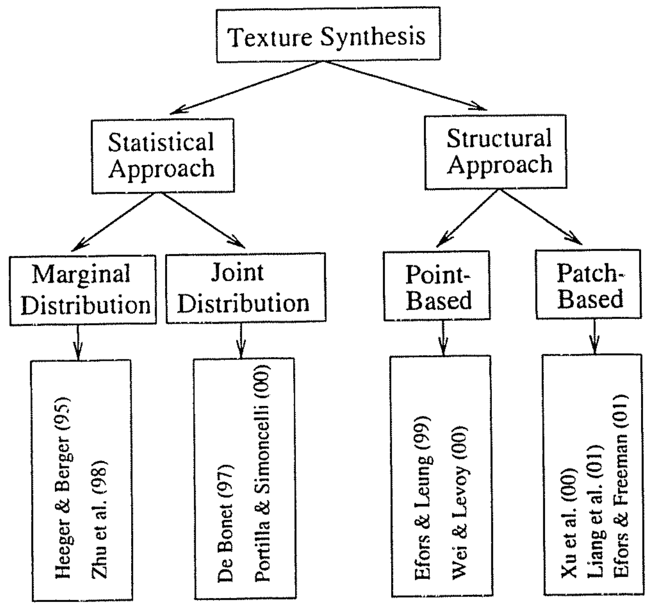

Figure 1. Categorization of texture synthesis methods. At the leaf nodes, some representative references are listed.

Using the structural approach, synthesizing one pixel at a time has drawbacks because a lot of searching is wasted on pixels that already know their fate [4]. Consequently, patch-based synthesis methods have been proposed recently [4] [10] [7]. Efros and Freeman [4] introduced a quilting method to stitch patches together. Patch-based methods

[10] [7] [4] share a common property, *i.e.*, to compute the degree of overlap between a new patch and existing ones to determine if this new patch can be added to the synthesized image.

We propose a novel patch-based, structural method for texture synthesis. Rather than finding a patch that has a large degree of overlap with an existing patch, we simply pick, if possible, the patch that is the direct neighbor of the current patch in the input texture. We explain the underlying mechanism and give a simple algorithm in next two sections. Experimental results are shown in Section 4.

2 Motivation

Assume there are two neighboring patches $\mathbf{p}$ and $\mathbf{q}$ in the synthesized texture image $\mathbf{T}$, and $\mathbf{p}'$ and $\mathbf{q}'$ (of the same size as $\mathbf{p}$ and $\mathbf{q}$) are two neighboring patches in the input texture image $\mathbf{I}$. The basic idea is that if $\mathbf{p}$ is structurally similar to $\mathbf{p}'$, then $\mathbf{q}$ should be similar to $\mathbf{q}'$. We illustrate this local similarity in Fig. 2. Consequently, if we already have a patch $\mathbf{p}$ in the synthesized image $\mathbf{T}$, we can simply duplicate patch $\mathbf{q}'$ in $\mathbf{I}$ as $\mathbf{q}$ in $\mathbf{T}$. We call this strategy Markov information propagation (MIP) because it locally propagates patch data to adjacent patches. In addition, since patches $\mathbf{p}'$ and $\mathbf{q}'$ are seamlessly connected in the input texture $\mathbf{I}$, $\mathbf{p}$ and $\mathbf{q}$ will be seamlessly connected too, if they are duplications of $\mathbf{p}'$ and $\mathbf{q}'$.

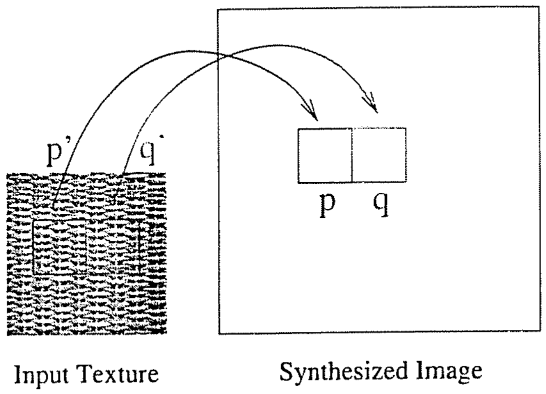

Figure 2. Illustration of the MIP strategy for texture synthesis. Patches $\mathbf{p}'$ and $\mathbf{q}'$ are neighbors in the input texture, while $\mathbf{p}$ and $\mathbf{q}$ are neighbors in the synthesized image.

3 Markov Information Propagation for Texture Synthesis

The Markov information propagation (MIP) strategy is different from overlap-focused methods such as [4] [10] [7] in that if a patch is selected from the input texture, we already know the information in its adjacent patches. We can use the data from adjacent patches directly, rather than search for a different patch in the whole input texture to find one with some degree of overlap to the current patch [10] [7] [4]. Furthermore, adjacent patches in the input texture are obviously connected seamlessly. When they are pasted together in the synthesized image, there is no need to worry about the boundary between them. Our MIP strategy tries to make direct use of the information from the input texture largely whenever possible, instead of always searching for a new patch each time when one is to be added to the synthesized image.

In texture synthesis, maybe the fastest and simplest method is to tile small patches (or even the input texture itself) together, if they are tileable. Our MIP strategy shares the simplicity of tiling, that is, adjacent patches in the input texture can be "tiled" seamlessly with the patch that is surrounded by them.

In the MIP strategy, there is one big issue to address, i.e., whenever patch $\mathbf{p}'$ is close to the boundary of image $\mathbf{I}$. For instance, if $\mathbf{p}'$ is close to the right side, we cannot obtain its right neighbor $\mathbf{q}'$ in the input texture $\mathbf{I}$. To deal with this problem, we need to find another patch $\mathbf{p}''$ in $\mathbf{I}$, which is perceptually and structurally similar to the patch $\mathbf{p}$. Therefore the question is how to define this kind of local and structural similarity. There are two criteria in general to determine this kind of local similarity. The first is to compute the difference between two patch images. Ideally, if the difference is zero, and hence two patches are the same, they are definitely structurally similar. But this measure is too strict in most cases. The second criterion is to measure only at the boundary regions where two adjacent patches meet, and the goal is only to keep the connection perceptually seamless. Currently, we simply use the summed square difference (SSD) to measure the local structural similarity between patches $\mathbf{p}$ and $\mathbf{p}''$, which satisfies the first but not second criterion.

The MIP algorithm works as follows:

- Select randomly a patch $\mathbf{p}'$ from the given texture image $\mathbf{I}$, and paste it to some position in the to-be-synthesized texture $\mathbf{T}$; mark the position as $\mathbf{p}$.
- Paste one neighbor $\mathbf{q}'$ of the patch $\mathbf{p}'$ in $\mathbf{I}$ to the position $\mathbf{q}$ in $\mathbf{T}$, where $\mathbf{q}$ is the corresponding neighbor of $\mathbf{p}$ in $\mathbf{T}$.
- If $\mathbf{p}'$ is close to the boundary of $\mathbf{I}$, find another patch $\mathbf{p}''$ in $\mathbf{I}$, which is structurally most similar to $\mathbf{p}$, and paste its appropriate neighbor $\mathbf{q}''$ to position $\mathbf{q}$ in $\mathbf{T}$.
- Add patches continuously along pre-defined directions until the whole image $\mathbf{T}$ is filled.

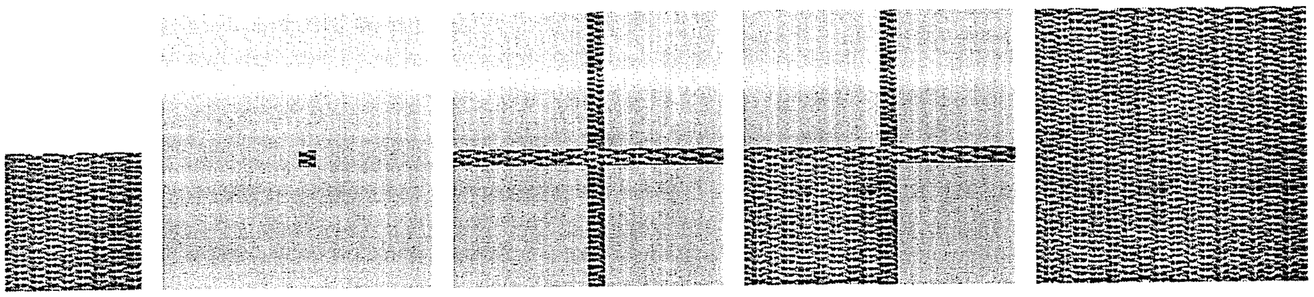

Figure 3. A simple approach for using Markov information propagation for texture synthesis. The left image is the input texture (128x128). A randomly selected patch from the input is pasted in the center of the to-be-synthesized image, and then patches are added successively along both the horizontal and vertical directions. Next, these patches are used as the starting patches for adding other horizontal and vertical patches until the output image is filled.

In the above algorithm, we do not address the issue of how to propagate local information. Currently, we use a simple method. We put the first randomly selected patch in the center of the synthesized image (it is empty in the beginning), and new patches are added along either the horizontal or vertical direction. The synthesis process is demonstrated in Fig. 3. In future work, we will investigate expect more complex pasting methods such as along the diagonal direction based on a pair of adjacent patches.

4 Texture Synthesis Results

We used the MIP algorithm to synthesize both artificial and natural textures. Some results are shown in Fig. 5. The size of all the input textures is 128x128, and the output is 256x256. The patch size is fixed at 16x16. In this figure, the first two rows show some artificial textures (from [8]). The last three rows show the synthesis of some natural textures. While the images in the last row are not as good, the reason is probably because of the simple filling method currently used. The synthesis process is fast, and can be further accelerated (in searching for patch $\mathbf{p}''$ in $\mathbf{I}$) by fast nearest neighbor search [9].

We compared our texture synthesis results with other methods, using the same images as in [4]. The comparison results are shown in Fig. 4. Our results on these three images are comparable to the image quilting results [4], and better than Wei and Levoy's results [9]. The synthesis results based on the methods of Portilla and Simoncelli [8] and Xu *et al.* [10] are not shown here, but are given in [4]. One can see from [4] that our results are better than those obtained by [8] and [10] on these three images.

5 Conclusions

We introduced a Markov information propagation (MIP) strategy for texture synthesis, and described a simple, fast algorithm. Some preliminary results for both artificial and natural textures were presented. Further research is needed to improve the similarity measure and pasting style used in the method. Finally, the MIP algorithm can also be used for constrained synthesis or hole filling, although we did not address that problem in this paper.

References

[1] J. S. De Bonet, Multiresolution sampling procedure for analysis and synthesis of texture images, In *SIGGRAPH 97*, p361-368, 1997.

[2] G. R. Cross and A. K. Jain, Markov Random Field Texture Models. *IEEE Trans. on PAMI*, 5:25-39, 1983.

[3] A. A. Efros and T. K. Leung, Texture synthesis by non-parametic sampling, *ICCV'99*, p1033-1038, 1999.

[4] A. A. Efros and W. T. Freeman, Image quilting for texture synthesis and transfer, in *SIGGRAPH 01*, p341-346, 2001.

[5] D. J. Heeger and J. R. Bergen, Pyramid-based texture analysis/synthesis, in *SIGGRAPH 95*, p229-238, 1995.

[6] B. Julesz, Visual pattern discrimination, *IRE Transactions on Information Theory*, 8(2): 84-92, 1962.

[7] L. Liang, C. Liu, Y. Xu, B. Guo, and H. Y. Shum, Real-time texture synthesis by patch-based sampling. MSR-TR-2001-40, Microsoft Research, March 2001.

[8] J. Portilla and E. P. Simoncelli, A parametric texture model based on joint statistics of complex wavelet coefficients, *International Journal of Computer Vision*, 40(1):49-71, 2000.

[9] L. Y. Wei and M. Levoy, Fast texture synthesis using tree-structured vector quantization, in *SIGGRAPH 00*, p479-488, 2000.

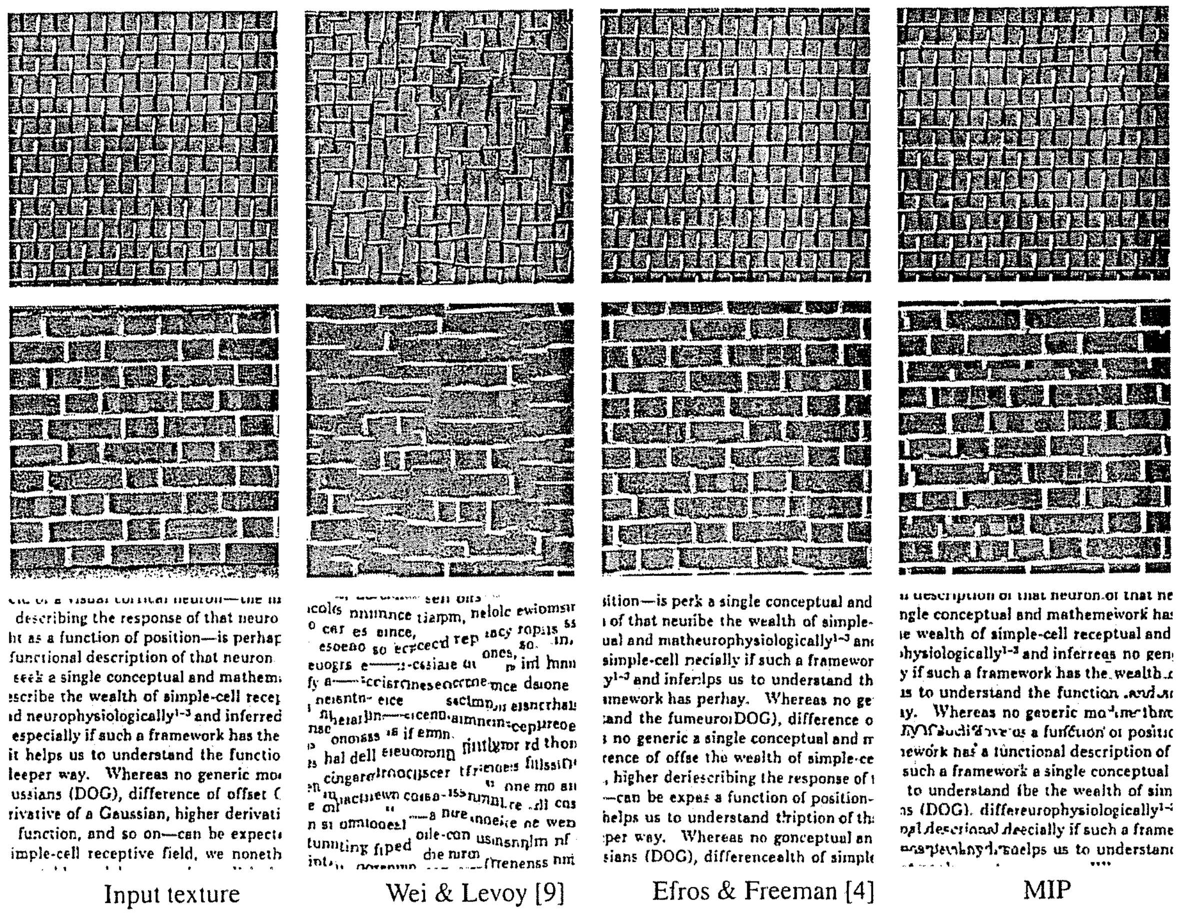

Figure 4. Comparison of different texture synthesis methods. The results of [8] and [10] are not shown here, but can be found in [4].

[10] Y. Xu, B. Guo, and H. Y. Shum, Chaos mosaic: Fast and memory efficient texture synthesis, MSR-TR-2000-32, Microsoft Research, April, 2000.

[11] S. Zhu, Y. Wu, and D. Mumford, Filters, random fields and maximum entropy (frame), *International Journal of Computer Vision*, 27(2):1-20, March/April, 1998.

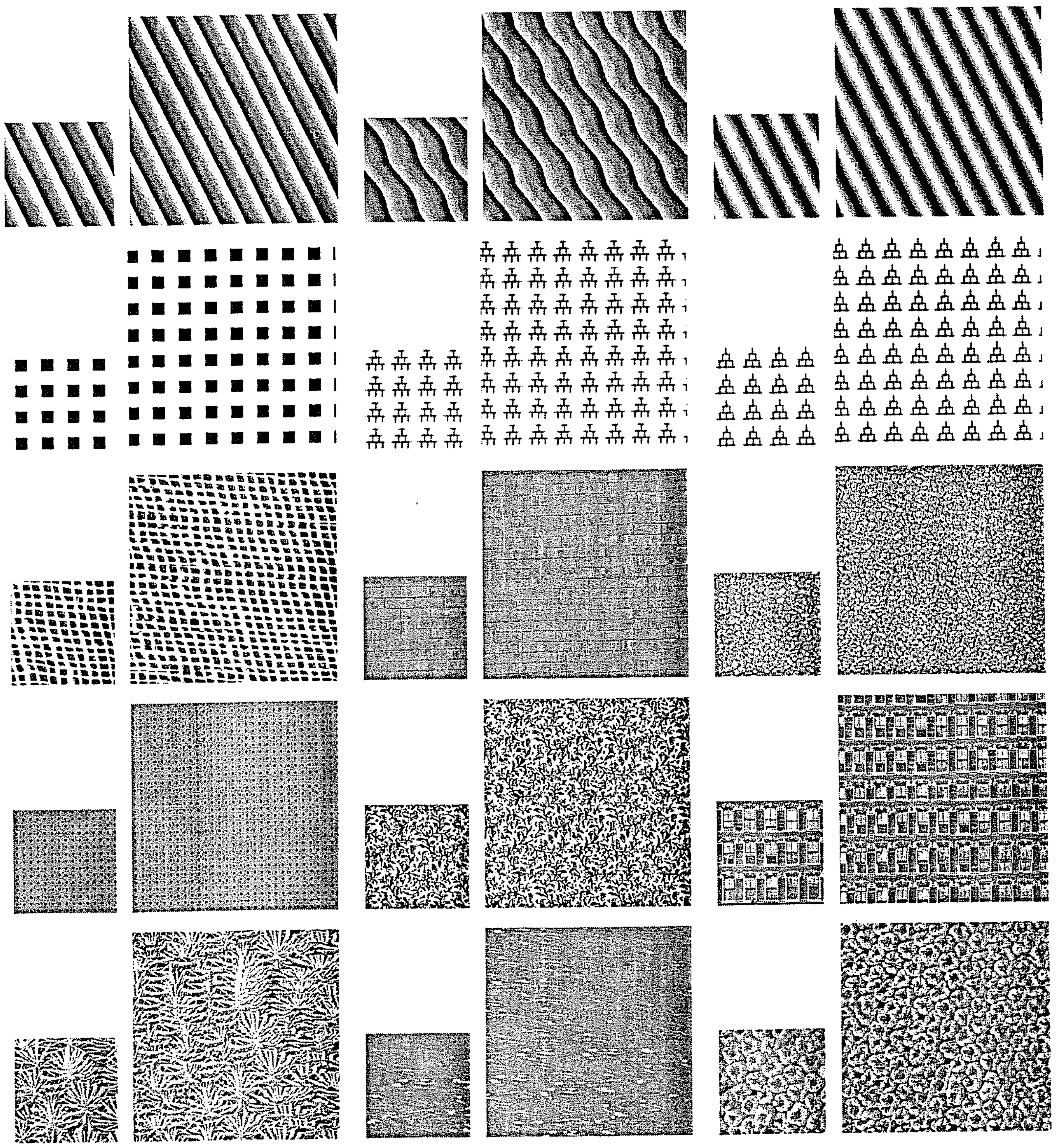

Figure 5. Texture synthesis examples. Each small image (128x128) is the input texture, and the big ones are synthesized textures (256x256).