IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

# Texture Synthesis by Non-parametric Sampling

Alexei A. Efros and Thomas K. Leung
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776, U.S.A.
{efros,leungt}@cs.berkeley.edu

## Abstract

*A non-parametric method for texture synthesis is proposed. The texture synthesis process grows a new image outward from an initial seed, one pixel at a time. A Markov random field model is assumed, and the conditional distribution of a pixel given all its neighbors synthesized so far is estimated by querying the sample image and finding all similar neighborhoods. The degree of randomness is controlled by a single perceptually intuitive parameter. The method aims at preserving as much local structure as possible and produces good results for a wide variety of synthetic and real-world textures.*

## 1. Introduction

Texture synthesis has been an active research topic in computer vision both as a way to verify texture analysis methods, as well as in its own right. Potential applications of a successful texture synthesis algorithm are broad, including occlusion fill-in, lossy image and video compression, foreground removal, etc.

The problem of texture synthesis can be formulated as follows: let us define texture as some visual pattern on an infinite 2-D plane which, at some scale, has a stationary distribution. Given a finite sample from some texture (an image), the goal is to synthesize other samples from the same texture. Without additional assumptions this problem is clearly ill-posed since a given texture sample could have been drawn from an infinite number of different textures. The usual assumption is that the sample is large enough that it somehow captures the stationarity of the texture and that the (approximate) scale of the texture elements (*texels*) is known.

Textures have been traditionally classified as either regular (consisting of repeated texels) or stochastic (without explicit texels). However, almost all real-world textures lie somewhere in between these two extremes and should be captured with a single model. In this paper we have chosen a statistical non-parametric model based on the assumption of spatial locality. The result is a very simple texture synthesis algorithm that works well on a wide range of textures and is especially well-suited for constrained synthesis problems (hole-filling).

### 1.1. Previous work

Most recent approaches have posed texture synthesis in a statistical setting as a problem of sampling from a probability distribution. Zhu et. al. [12] model texture as a Markov Random Field and use Gibbs sampling for synthesis. Unfortunately, Gibbs sampling is notoriously slow and in fact it is not possible to assess when it has converged. Heeger and Bergen [6] try to coerce a random noise image into a texture sample by matching the filter response histograms at different spatial scales. While this technique works well on highly stochastic textures, the histograms are not powerful enough to represent more structured texture patterns such as bricks.

De Bonet [1] also uses a multi-resolution filter-based approach in which a texture patch at a finer scale is conditioned on its "parents" at the coarser scales. The algorithm works by taking the input texture sample and randomizing it in such a way as to preserve these inter-scale dependencies. This method can successfully synthesize a wide range of textures although the randomness parameter seems to exhibit perceptually correct behavior only on largely stochastic textures. Another drawback of this method is the way texture images larger than the input are generated. The input texture sample is simply replicated to fill the desired dimensions before the synthesis process, implicitly assuming that all textures are tilable which is clearly not correct.

The latest work in texture synthesis by Simoncelli and Portilla [9, 11] is based on first and second order properties of joint wavelet coefficients and provides impressive results. It can capture both stochastic and repeated textures quite well, but still fails to reproduce high frequency information on some highly structured patterns.

## 1.2. Our Approach

In his 1948 article, *A Mathematical Theory of Communication* [10], Claude Shannon mentioned an interesting way of producing English-sounding written text using $n$-grams. The idea is to model language as a generalized Markov chain: a set of $n$ consecutive letters (or words) make up an $n$-gram and completely determine the probability distribution of the next letter (or word). Using a large sample of the language (e.g., a book) one can build probability tables for each $n$-gram. One can then repeatedly sample from this Markov chain to produce English-sounding text. This is the basis for an early computer program called MARK V. SHANEY, popularized by an article in *Scientific American* [4], and famous for such pearls as: *"I spent an interesting evening recently with a grain of salt"*.

This paper relates to an earlier work by Popat and Picard [8] in trying to extend this idea to two dimensions. The three main challenges in this endeavor are: 1) how to define a unit of synthesis (a letter) and its context ($n$-gram) for texture, 2) how to construct a probability distribution, and 3) how to linearize the synthesis process in 2D.

Our algorithm "grows" texture, pixel by pixel, outwards from an initial seed. We chose a single pixel $p$ as our unit of synthesis so that our model could capture as much high frequency information as possible. All previously synthesized pixels in a square window around $p$ (weighted to emphasize local structure) are used as the context. To proceed with synthesis we need probability tables for the distribution of $p$, given all possible contexts. However, while for text these tables are (usually) of manageable size, in our texture setting constructing them explicitly is out of the question. An approximation can be obtained using various clustering techniques, but we choose not to construct a model at all. Instead, for each new context, the sample image is queried and the distribution of $p$ is constructed as a histogram of all possible values that occurred in the sample image as shown on Figure 1. The non-parametric sampling technique, although simple, is very powerful at capturing statistical processes for which a good model hasn't been found.

# 2. The Algorithm

In this work we model texture as a Markov Random Field (MRF). That is, we assume that the probability distribution of brightness values for a pixel given the brightness values of its spatial neighborhood is independent of the rest of the image. The neighborhood of a pixel is modeled as a square window around that pixel. The size of the window is a free parameter that specifies how stochastic the user believes this texture to be. More specifically, if the texture is presumed to be mainly regular at high spatial frequencies and mainly stochastic at low spatial frequencies, the size of the window should be on the scale of the biggest regular feature.

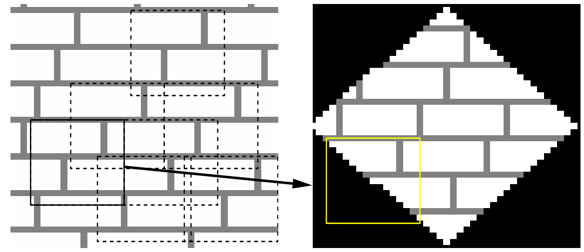


Figure 1. Algorithm Overview. Given a sample texture image (left), a new image is being synthesized one pixel at a time (right). To synthesize a pixel, the algorithm first finds all neighborhoods in the sample image (boxes on the left) that are similar to the pixel's neighborhood (box on the right) and then randomly chooses one neighborhood and takes its center to be the newly synthesized pixel.

## 2.1. Synthesizing one pixel

Let $I$ be an image that is being synthesized from a texture sample image $I_{smp} \subset I_{real}$ where $I_{real}$ is the real infinite texture. Let $p \in I$ be a pixel and let $\omega(p) \subset I$ be a square image patch of width $w$ centered at $p$. Let $d(\omega_1, \omega_2)$ denote some perceptual distance between two patches. Let us assume for the moment that all pixels in $I$ except for $p$ are known. To synthesize the value of $p$ we first construct an approximation to the conditional probability distribution $P(p|\omega(p))$ and then sample from it.

Based on our MRF model we assume that $p$ is independent of $I \setminus \omega(p)$ given $\omega(p)$. If we define a set

$$\Omega(p) = \{\omega' \subset I_{real} : d(\omega', \omega(p)) = 0\}$$

containing all occurrences of $\omega(p)$ in $I_{real}$, then the conditional pdf of $p$ can be estimated with a histogram of all center pixel values in $\Omega(p)$. [1] Unfortunately, we are only given $I_{smp}$, a finite sample from $I_{real}$, which means there might not be any matches for $\omega(p)$ in $I_{smp}$. Thus we must use a heuristic which will let us find a plausible $\Omega'(p) \approx \Omega(p)$ to sample from. In our implementation, a variation of the nearest neighbor technique is used: the closest match $\omega_{best} = argmin_\omega \, d(\omega(p), \omega) \subset I_{smp}$ is found, and all image patches $\omega$ with $d_(\omega(p), \omega) < (1+\epsilon) d(\omega(p), \omega_{best})$ are included in $\Omega'(p)$, where $\epsilon = 0.1$ for us. The center pixel values of patches in $\Omega'(p)$ give us a histogram for $p$, which can then be sampled, either uniformly or weighted by $d$.

Now it only remains to find a suitable distance $d$. One choice is a normalized sum of squared differences metric $d_{SSD}$. However, this metric gives the same weight to any mismatched pixel, whether near the center or at the edge of the window. Since we would like to preserve the local structure of the texture as much as possible, the error for

[1] This is somewhat misleading, since if all pixels in $\Omega(p)$ except $p$ are known, the pdf for $p$ will simply be a delta function for all but highly stochastic textures, since a single pixel can rarely be a feature by itself.

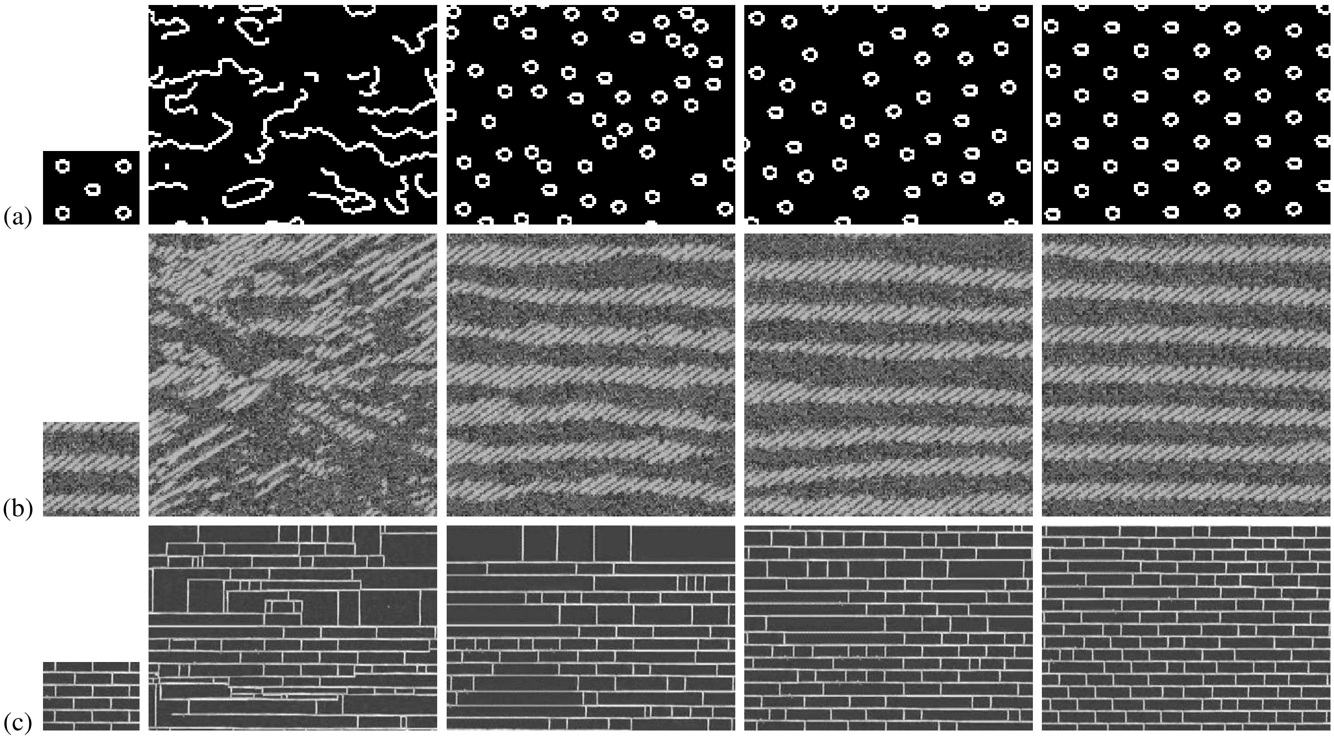



Figure 2. Results: given a sample image (left), the algorithm synthesized four new images with neighborhood windows of width 5, 11, 15, and 23 pixels respectively. Notice how perceptually intuitively the window size corresponds to the degree of randomness in the resulting textures. Input images are: (a) synthetic rings, (b) Brodatz texture D11, (c) brick wall.

nearby pixels should be greater than for pixels far away. To achieve this effect we set $d = d_{SSD} * G$ where $G$ is a two-dimensional Gaussian kernel.

## 2.2. Synthesizing texture

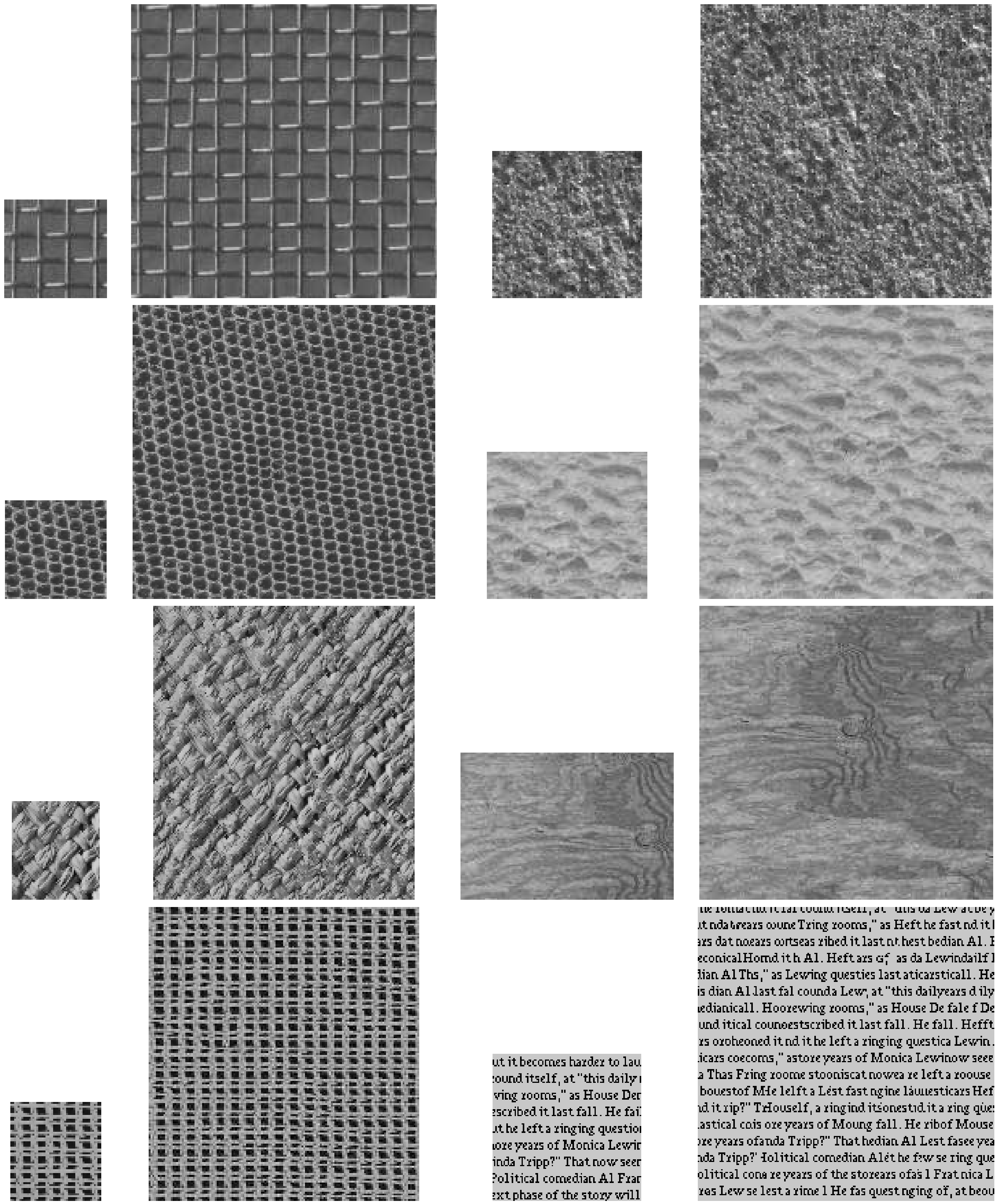
In the previous section we have discussed a method of synthesizing a pixel when its neighborhood pixels are already known. Unfortunately, this method cannot be used for synthesizing the entire texture or even for hole-filling (unless the hole is just one pixel) since for any pixel the values of only some of its neighborhood pixels will be known. The correct solution would be to consider the joint probability of all pixels together but this is intractable for images of realistic size.

Instead, a Shannon-inspired heuristic is proposed, where the texture is grown in layers outward from a 3-by-3 seed taken randomly from the sample image (in case of hole filling, the synthesis proceeds from the edges of the hole). Now for any point $p$ to be synthesized only *some* of the pixel values in $\omega(p)$ are known (i.e. have already been synthesized). Thus the pixel synthesis algorithm must be modified to handle unknown neighborhood pixel values. This can be easily done by only matching on the known values in $\omega(p)$ and normalizing the error by the total number of known pixels when computing the conditional pdf for $p$. This heuristic does not guarantee that the pdf for $p$ will stay valid as the rest of $\omega(p)$ is filled in. However, it appears to be a good approximation in practice. One can also treat this as an initialization step for an iterative approach such as Gibbs sampling. However, our trials have shown that Gibbs sampling produced very little improvement for most textures. This lack of improvement indicates that the heuristic indeed provides a good approximation to the desired conditional pdf.

# 3. Results

Our algorithm produces good results for a wide range of textures. The only parameter set by the user is the width $w$ of the context window. This parameter appears to intuitively correspond to the human perception of randomness for most textures. As an example, the image with rings on Figure 2a has been synthesized several times while increasing $w$. In the first synthesized image the context window is not big enough to capture the structure of the ring so only the notion of curved segments is preserved. In the next image, the context captures the whole ring, but knows nothing of inter-ring distances producing a Poisson process pattern. In the third image we see rings getting away from each other (so called Poisson process with repulsion), and finally in the last image the inter-ring structure is within the reach of the window as the pattern becomes almost purely structured.

Figure 3 shows synthesis examples done on real-world textures. Examples of constrained synthesis are shown on




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

Figure 3. Texture synthesis on real-world textures: (a) and (c) are original images, (b) and (d) are synthesized. (a) images D1, D3, D18, and D20 from Brodatz collection [2], (c) granite, bread, wood, and text (a homage to Shannon) images.

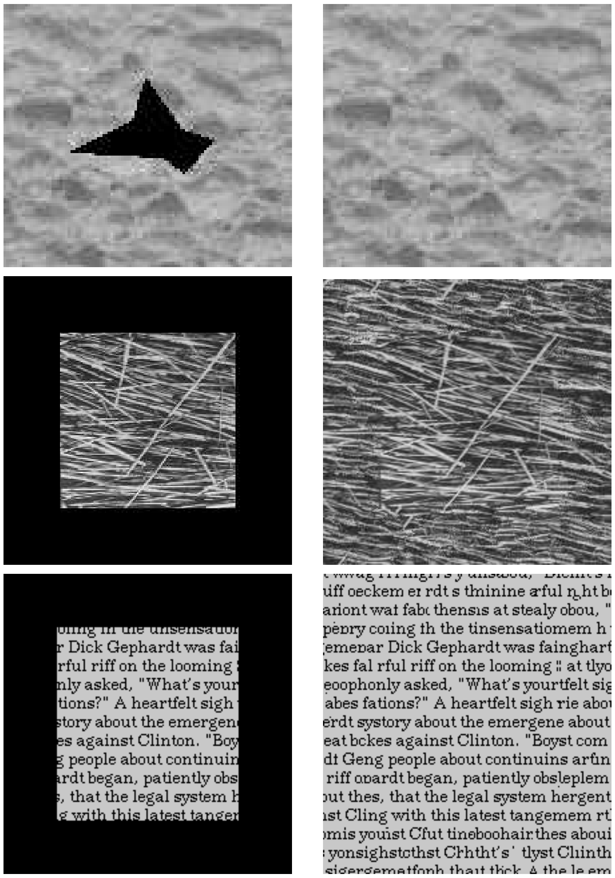


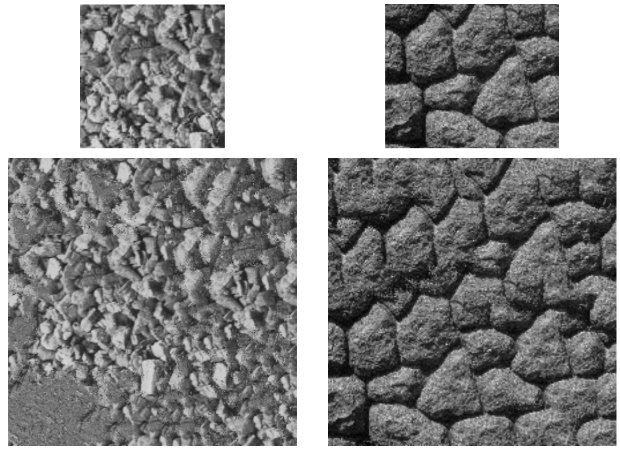
Figure 4. Examples of constrained texture synthesis. The synthesis process fills in the black regions.

Figure 4. The black regions in each image are filled in by sampling from that same image. A comparison with De Bonet [1] at varying randomness settings is shown on Figure 7 using texture 161 from his web site.

## 4. Limitations and Future Work

As with most texture synthesis procedures, only frontal-parallel textures are handled. However, it is possible to use Shape-from-Texture techniques [5, 7] to pre-warp an image into frontal-parallel position before synthesis and post-warp afterwards.

One problem of our algorithm is its tendency for some textures to occasionally "slip" into a wrong part of the search space and start growing garbage (Figure 5a) or get locked onto one place in the sample image and produce verbatim copies of the original (Figure 5b). These problems occur when the texture sample contains too many different types of texels (or the same texels but differently illuminated) making it hard to find close matches for the neighborhood context window. These problems can usually be eliminated by providing a bigger sample image. We have also used growing with limited backtracking as a solution.



(a) (b)

Figure 5. Failure examples. Sometimes the growing algorithm "slips" into a wrong part of the search space and starts growing garbage (a), or gets stuck at a particular place in the sample image and starts verbatim copying (b).

In the future we plan to study automatic window-size selection, including non-square windows for elongated textures. We are also currently investigating the use of texels as opposed to pixels as the basic unit of synthesis (similar to moving from letters to words in Shannon's setting). This is akin to putting together a jigsaw puzzle where each piece has a different shape and only a few can fit together. Currently, the algorithm is quite slow but we are working on ways to make it more efficient.

## 5. Applications

Apart from letting us gain a better understanding of texture models, texture synthesis can also be used as a tool for solving several practical problems in computer vision, graphics, and image processing. Our method is particularly versatile because it does not place any constraints on the shape of the synthesis region or the sampling region, making it ideal for constrained texture synthesis such as hole-filling. Moreover, our method is designed to preserve local image structure, such as continuing straight lines, so there are no visual discontinuities between the original hole outline and the newly synthesized patch.

For example, capturing a 3D scene from several camera views will likely result in some regions being occluded from all cameras [3]. Instead of letting them appear as black holes in a reconstruction, a localized constrained texture synthesis can be performed to fill in the missing information from the surrounding region. As another example, consider the problem of boundary handling when performing a convolution on an image. Several methods exist, such as zero-fill, tiling and reflection, however all of them may introduce discontinuities not present in the original image. In many cases, texture synthesis can be used to extrapolate the

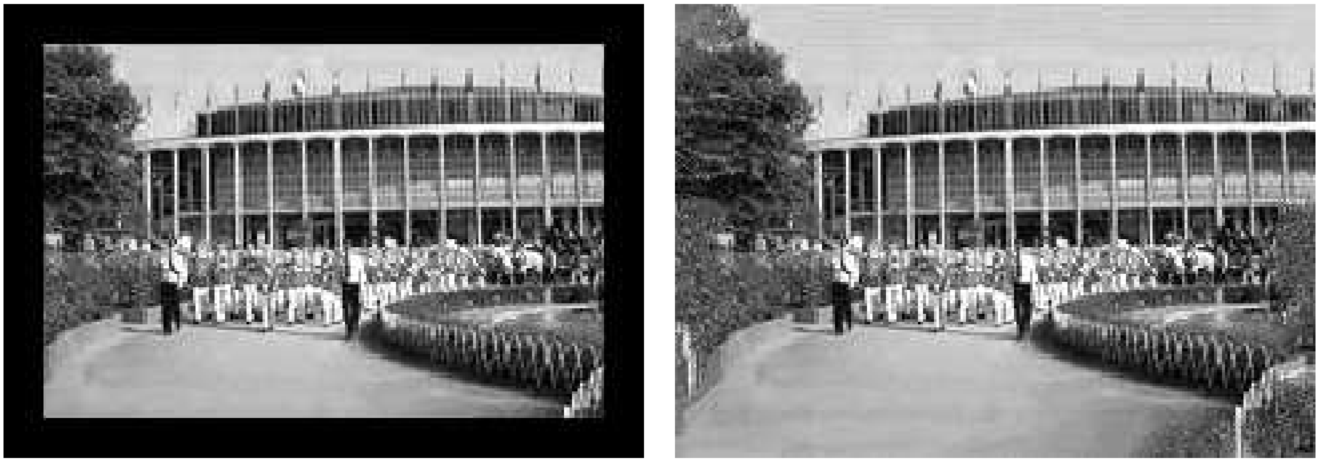
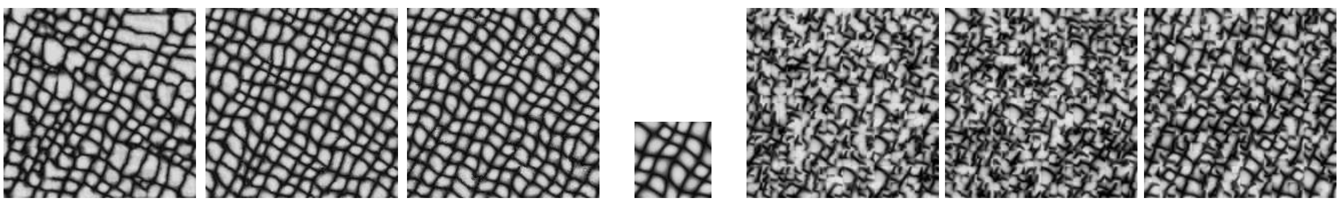


Figure 6. The texture synthesis algorithm is applied to a real image (left) extrapolating it using itself as a model, to result in a larger image (right) that, for this particular image, looks quite plausible. This technique can be used in convolutions to extend filter support at image boundaries.



Our method sample image De Bonet's method

Figure 7. Texture synthesized from sample image with our method compared to [1] at decreasing degree of randomness.

image by sampling from itself as shown on Figure 6.

The constrained synthesis process can be further enhanced by using image segmentation to find the exact sampling region boundaries. A small patch of each region can then be stored together with region boundaries as a lossy compression technique, with texture synthesis being used to restore each region separately. If a figure/ground segmentation is possible and the background is texture-like, then foreground removal can be done by synthesizing the background into the foreground segment.

Our algorithm can also easily be applied to motion synthesis such as ocean waves, rolling clouds, or burning fire by a trivial extension to 3D.

**Acknowledgments:** We would like to thank Alex Berg, Elizaveta Levina, and Yair Weiss for many helpful discussions and comments. This work has been supported by NSF Graduate Fellowship to AE, Berkeley Fellowship to TL, ONR MURI grant FDN00014-96-1-1200, and the California MICRO grant 98-096.

## References

[1] J. S. D. Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH '97*, pages 361–368, 1997.

[2] P. Brodatz. *Textures*. Dover, New York, 1966.

[3] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96*, pages 11–20, August 1996.

[4] A. Dewdney. A potpourri of programmed prose and prosody. *Scientific American*, 122-TK, June 1989.

[5] J. Garding. Surface orientation and curvature from differential texture distortion. *ICCV*, pages 733–739, 1995.

[6] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95*, pages 229–238, 1995.

[7] J. Malik and R. Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 23(2):149–168, 1997.

[8] K. Popat and R. W. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression. In *Proc. SPIE Visual Comm. and Image Processing*, 1993.

[9] J. Portilla and E. P. Simoncelli. Texture representation and synthesis using correlation of complex wavelet coefficient magnitudes. TR 54, CSIC, Madrid, April 1999.

[10] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. Journal*, 27, 1948.

[11] E. P. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *Proc. 5th Int'l Conf. on Image Processing Chicago, IL*, 1998.

[12] S. C. 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.