Asymptotically Admissible Texture Synthesis

Yingqing Xu[1], Song-Chun Zhu[2]
Baining Guo[1], Heung-Yeung Shum[1]

[1] Microsoft Research China
[2] The Ohio State University

May 14, 2001

Abstract

Recently there is a resurgent interest in example based texture analysis and synthesis in both computer vision and computer graphics. While study in computer vision is concerned with learning accurate texture models, research in graphics is aimed at effective algorithms for texture synthesis without necessarily obtaining explicit texture model. This paper makes three contributions to this recent excitement. First, we introduce a theoretical framework for designing and analyzing texture sampling algorithms. This framework, built upon the mathematical definition of textures, measures a texture sampling algorithm using admissibility, effectiveness, and sampling speed. Second, we compare and analyze texture sampling algorithms based on admissibility and effectiveness. In particular, we propose different design criteria for texture analysis algorithms in computer vision and texture synthesis algorithms in computer graphics. Finally, we develop a novel texture synthesis algorithm which samples from a subset of the Julesz ensemble by pasting texture patches from the sample texture. A key feature of our algorithm is that it can synthesize high-quality textures extremely fast. On a mid-level PC we can synthesize a 512×512 texture from a 64×64 sample in just 0.03 second. This algorithm has been tested through extensive experiments and we report sample results from our experiments.

1 Introduction

Texture analysis and texture synthesis have attracted much interest recently in computer vision and computer graphics. A key question in texture study is what human vision perceives from texture images. More precisely, what are the *sufficient and necessary (or minimum)* texture features and statistics so that a pair of texture images sharing such statistics can be regarded as the ”same class” of texture for human perception. This theme, studied by psycho-physicists who are interested in the early stage of visual perception, dates back to (Julesz, 1962)[6]. In late 1980’s and early 1990’s, the psychological study also pointed to a conjecture that human texture perception is governed by the empirical histograms of Gabor filtered images, inspired by the success of Gabor filters as a model for V1 cells and in image coding. Two texture images are regarded the same class if they share the same histograms for a band of Gabor filtered images. Unfortunately, a rigorous study of this problem was prohibited by the lack of mathematical tools for synthesizing texture pairs that share a given set of statistics.

Motivated by the psychology studies, Heeger and Bergen in 1995 proposed a pyramid based algorithm for texture synthesis that can approximately match marginal histograms of filter responses[5]. A mathematical model called FRAME is proposed in (Zhu, Wu and Mumford, 1997) [15]. The FRAME model integrates the filters and histograms into Markov random field models and adopts an accurate but expensive Markov chain Monte Carlo (MCMC) method for texture synthesis. Furthermore, the selection of feature statistics is well posed on a minimax entropy principle [15]. Other interesting algorithms explore texture synthesis with joint statistics of filter responses. For example, (De Bonet, 1997) synthesized textures by matching joint histogram of a long vector of filter response [2]. (Portilla and Simoncelli, 2000) studied an iterative projection method for matching the correlations of some filter responses[9]. These methods, among many other work in the literature, represent two distinct paths of texture synthesis. The first path learns analytical models of texture using Markov random fields, and synthesize texture by stochastic sampling from the model[1, 15]. The second path synthesize texture by matching statistics without deriving analytical texture model[2, 9, 14]. The two paths are unified by the equivalence between the Julesz ensemble and FRAME models [12].

More recently, several algorithms have also been proposed to synthesize textures by matching only local distributions. For example, a non-parametric estimation of MRF models is introduced by (Efros and Leung, 1999)[3], which can be further accelerated using a tree-structured VQ method (Wei and Levoy, 2000)[11].

The idea of estimating a non-parametric density can be traced back to Papat and Picard, 1993)[8], where they estimate an auto-regression model for ”growing” texture by sampling from a cluster-based model. These methods are good for synthesis (e.g., easy to be implemented and good-quality synthesis results), but inconvenient for analysis.

In this paper, we intend to contribute to the recent excitement of texture synthesis in both theoretical and practical aspects. Firstly, we introduce a theoretical framework for designing and analyzing texture sampling algorithms. While a texture sampling algorithm is ”acceptable” as long as it can synthesize satisfactory textures, it is critical to understand how to measure these algorithms in more rigorous terms. Our theoretical framework, built upon the mathematical definition of textures from [14, 12], measures a texture sampling algorithm using admissibility, effectiveness, and sampling speed. For a texture synthesis algorithm, admissibility measures the perceived difference in the texture characteristics of the sample texture and synthesized texture, whereas effectiveness measures the richness of the textures generated.

Secondly, we demonstrate how to analyze texture sampling algorithms based on admissibility and effectiveness. In particular, we show that there should be different design criteria for texture analysis algorithms in computer vision and texture synthesis algorithms in computer graphics. In texture analysis, a texture sampling algorithm must be optimally admissible, i.e., be consistent with the definition of the texture — the Julesz ensemble. For texture synthesis, on the other hand, an algorithm is acceptable provided that it samples from a subset of the Julesz ensemble. Realizing the difference in design criteria allows us to better balance the trade-off of admissibility, effectiveness, and sampling speed when designing texture synthesis algorithms.

Finally, we develop a novel texture synthesis algorithm which trades effectiveness for speed. The equivalence of the Julesz ensemble and FRAME models [12] states that texture synthesis can be done without necessarily learning Markov random field models provided that the texture is synthesized on a large image lattice[14]. Our algorithm adopts this strategy and samples from a subset of the Julesz ensemble by pasting texture patches from the sample texture. A key feature of our algorithm is that it can synthesize high-quality textures extremely fast. On a mid-level PC we can synthesize a 512×512 texture from a 64×64 sample in just 0.03 second. This algorithm has been tested through extensive experiments and we report sample results from our experiments.

The paper is organized as follows. In Section (2), we introduce the concepts of admissibility and effectiveness for texture sampling algorithms. In Section (3), we

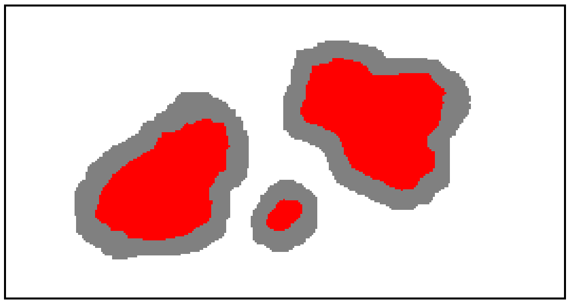


Figure 1: The lattice system and boundary conditions (shaded areas) for texture images.

first analyze various synthesis algorithms under this framework. Then we present our fast texture synthesis algorithm along with some experiments. We conclude the paper with a discussion in Section (4).

2 Admissibility and Effectiveness

In this section, we introduce the concepts of admissibility and effectiveness for texture sampling algorithms.

2.1 Texture Definition

We denote an image lattice by Λ. As Figure 1 shows, Λ can be of arbitrary shape and may not be connected. We denote by $\partial\Lambda$ the boundary of lattice Λ and show $\partial\Lambda$ as the shaded areas in Figure 1. A texture image is denoted by $\mathbf{I}_\Lambda$ with boundary condition $\mathbf{I}_{\partial\Lambda}$. In practice, we often ignore the effects of boundary conditions as the lattice Λ goes to infinity (or big enough) in the sense of Von Hove, which is defined as follows.

Definition 1 *A lattice is going to infinity in the sense of Von Hove if the ratio between the area of its boundary condition to the area of the lattice goes to zero, i.e.,* $\lim_{\Lambda\to\infty} \frac{|\partial\Lambda|}{|\Lambda|} = 0$.

In this paper we assume the Von Hove sense for any large lattice. For example, we say a lattice is large enough if $\frac{|\partial\Lambda|}{|\Lambda|} \leq 1/20$.

Now we are ready to give a rigorous definition for what we mean by ”a texture” following [14].

Definition 2 *As image lattice $\Lambda \to \infty$ in the Von Hove sense, the boundary effects and statistical fluctuation diminishes in the absence of phase transition,*[1] *a texture is defined as a set*

$$\Omega(\mathbf{h}) = \{\mathbf{I} \ : \mathbf{h}^*(\mathbf{I}) = \mathbf{h} \ \}. \tag{1}$$

where $\mathbf{h}^*(\mathbf{I})$ is the statistics extracted by human texture perception.

As discussed in [14, 12], the definition of a texture is up to human visual perception. If $\mathbf{h}^*(\mathbf{I}_\Lambda)$ is the features and statistics extracted by human visual cortex, then all texture images having the same statistics are perceived as the same texture.

As a spatial phenomenon, a texture cannot be defined on one point or even any finite image lattice Λ because the statistical fluctuation of $\mathbf{h}(\mathbf{I}_\Lambda)$. We must define texture on an infinite lattice, i.e. $\Lambda \to \infty$ in the Von Hove sense. On an infinite Λ, the statistical fluctuation will diminish and we obtain a deterministic definition of texture. Thus a vector $\mathbf{h}$ identifies a texture on infinite lattice Λ – just like a wavelength λ or (r, g, b) identifies a color! As Figure 2.a shows, different textures correspond to disjoint sets in the image space. Each texture set is associated with a distribution, denoted by $q(\mathbf{I}; \mathbf{h})$ which is uniformly distributed within $\Omega(\mathbf{h})$ and zero outside. We use the word *ensemble* from statistical physics to refer to a texture definition.

Definition 3 *An ensemble $\mathcal{E}$ is a set where elements may repeat. Thus it includes a base set Ω and a frequency f, $\mathcal{E} =< \Omega, f >$.*

On a large lattice, a texture $\Omega(\mathbf{h})$ is called a *Julesz ensemble* in (Zhu, Liu, and Wu, 2000). Because the uniform frequency in a Julesz ensemble, we don't distinguish the base set and the ensemble itself.

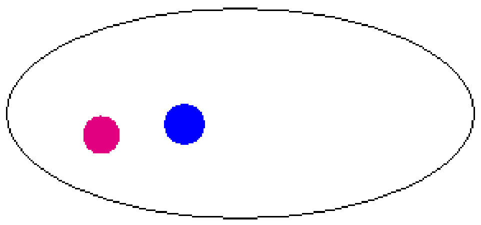
2.2 Admissibility

Given a texture example $\mathbf{I}_{\Lambda_o}$ on lattice Λ_o[2], let $\mathbf{h} = \mathbf{h}^*(\mathbf{I}_{\Lambda_o})$ be the observed statistics. Denote by $\Omega_\Lambda(\mathbf{h})$ the Julesz ensemble of the synthesized texture on lattice Λ. For a given texture synthesis algorithm A, there must exist a corresponding ensemble

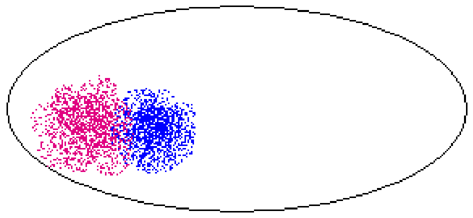
$$\mathcal{E}_A =< \Omega_A, f_A > .$$

[1] So far there is no confirmed example of phase transition in realistic texture, and thus we choose not to discuss this issue in this paper.

[2] As a lattice does not have to be connected, we consider the case of multiple observed texture images as different components of Λ_o.



a. Ensembles on infinite Λ



b. Ensembles on finite Λ

Figure 2: The ellipse represents the image space $\Omega_\Lambda = G^{|\Lambda|}$ where G is the set of grey levels for pixel intensity. Two ensembles overlap with each other on finite lattice and are disjoint on infinite lattice.

In Figure 3, the Julesz ensemble is represented by an ellipse in the image space Ω_Λ.

A basic property of a texture synthesis algorithm is whether the synthesized textures satisfy the texture definition according to the Julesz ensemble.

Definition 4 *For a texture synthesis algorithm A with ensemble $\mathcal{E}_A =< \Omega_A, f_A >$, A is said to be* admissible *if $\Omega_A \subset \Omega_\Lambda(\mathbf{h})$. A is said to be* optimally admissible *if $\Omega_A = \Omega_\Lambda(\mathbf{h})$ and f_A is the maximum entropy distribution.*

In Figure 3, the Ω_{A1} and Ω_{A2} areas represent the sampling sets of a non-admissible algorithm A_1 and an admissible algorithm A_2 respectively.

In practice, requiring an algorithm be admissible is often too restrictive. As we shall see, many successful texture synthesis algorithms are only asymptotically admissible, not admissible.

Definition 5 *If an algorithm is admissible when $\Lambda_o \rightarrow \infty$, then we say it is* asymptotically admissible.

For an algorithm A that is not admissible, we use *admissibility* to measure how far A is from being admissible. Suppose that for a texture sample $\mathbf{I}_{\Lambda_o}$ on

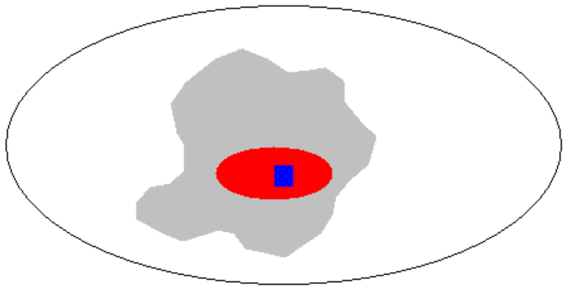


Figure 3: In the image space Ω_Λ, the ellipse represents the Julesz ensemble $\Omega_\Lambda(\mathbf{h})$. The sampling sets Ω_{A1} and Ω_{A2} are for a non-admissible algorithm $A1$ and an admissible algorithm $A2$ respectively.

a lattice Λ_o, the texture synthesis algorithm A estimates some statistics $\mathbf{h}_A$ in a k-dimensional feature space and synthesize a texture from the ensemble $\mathcal{E}_A =< \Omega_A, f_A >$ based on A. To make A admissible, one may choose a higher dimensional statistics $\mathbf{h}_A$ and hope to make Ω_A a subset of $\Omega(\mathbf{h})$. However, due to finite observations, the estimation error for the statistics $\mathbf{h}_A$ becomes large when the dimension increases. As a result, the actual texture ensemble Ω_A may grow outside the Julesz ensemble.

Inspired by the Fisher's information measure, we propose to measure admissibility by the inverse of the estimation variance

$$a_A = \frac{1}{|var[\mathbf{h}_A]|} = \frac{1}{|E[\mathbf{h}_A - E[(\mathbf{h}_A])^T(\mathbf{h}_A - E[\mathbf{h}_A])]|}$$

Why is admissibility an important measure for texture synthesis algorithms? Simply put, the admissibility measures the perceived difference in the texture characteristics in the sample texture $\mathbf{I}_{\Lambda_o}$ and synthesized texture $\mathbf{I}_\Lambda$. The more admissible an algorithm A is, the smaller is the perceived difference in the texture characteristics because the ensemble Ω_A defined by the estimated statistics $\mathbf{h}_A$ is more properly contained by the the Julesz ensemble $\Omega(\mathbf{h})$.

In practice, we cannot observe an infinity image although the Julesz ensemble is defined on infinite lattice. Thus we never know what the true Julesz ensemble is, never really know the true h. Because of ergodicity, we can take the ”single image statistics averaged over infinite lattice” as the ”expected statistics averaged over the whole ensemble of images”. Note that the concept of admissibility is related to but different from the ”practical ergodicity” in [9].

Therefore, a practical definition for admissibility becomes that “an algorithm is admissible with a confidence factor”, or specifically,

$$p(|\hat{h} - h| < \epsilon) > 1 - \delta.$$

for some small positive values of δ, ϵ.

Similar to the limit definition in calculus, given pre-set ϵ and δ, we need an observed image I with size $N = N(\epsilon, \delta)$ so that the above equation is satisfied. Each algorithm can be measured by the size of N_A it needs for a given ϵ, δ. The smaller the N_A, the ”more admissible” the algorithm. An algorithm is asymptotically admissible if for any ϵ there exists a finite lattice size N such that the estimation error is under ϵ.

2.3 Effectiveness

Another property of a texture synthesis algorithm is the richness of textures it generates.

Definition 6 *For any texture synthesis algorithm A, the* richness *of the algorithm is measured by the entropy of its ensemble frequency on the base set.*

$$r_A = \mathrm{entropy}(f_A) = -\int_{\Omega_A} f_A(\mathbf{I}_\Lambda) \log f_A(\mathbf{I}_\Lambda)\ d\mathbf{I}_\Lambda.$$

A more intuitive measure for the richness r_A is the volume of the base set of A, i.e., $|\Omega_A|$. It is trivial to prove the following proposition.

Proposition 1 *For any texture synthesis algorithm A with uniform ensemble frequency,*

$$r_A = \mathrm{entropy}(f_A) = \log |\Omega_A| + const.$$

The constant is determined by the discretization of the image space.

Now we define the *effectiveness* of an admissible algorithm as follows.

Definition 7 *For an* admissible *algorithm A, the* effectiveness *of the algorithm is defined as the difference,*

$$e_A = \mathrm{entropy}(f_A) - \mathrm{entropy}(q(\mathbf{I}; \mathbf{h})).$$

In case of uniform ensemble frequency,

$$e_A = \log \frac{|\Omega_A|}{|\Omega_\Lambda(\mathbf{h})|}$$

and the effectiveness of a texture synthesis algorithm is the logarithmic portion of the ideal Julesz ensemble it explores! It is trivial to prove the following proposition.

Proposition 2 *An optimally admissible algorithm A has maximum effectiveness $e_A = 0$.*

3 Design Texture Sampling Algorithms

Based on the theoretical framework introduced in the last section it is straightforward to discuss the design criteria for texture analysis in computer vision and texture synthesis in computer graphics.

3.1 Design Criteria

Admissibility, effectiveness and synthesis speed can be used to analyze and compare existing texture sampling algorithms, particularly those of texture synthesis in graphics.

- In computer vision, a texture model and texture analysis algorithm should be optimally admissible. For example, the FRAME model is preferred because it is consistent with the Julesz ensemble. Although a model that is not optimally admissible may still be successful in texture segmentation/classification in a small data set, it looses generality in building large vision systems.

- In computer graphics, a texture synthesis algorithm should preferably be admissible, but it does not have to be optimally admissible. For texture synthesis, the admissibility is more important than effectiveness.

- In practice, a designer often needs to trade off between the speed of algorithm and its effectiveness. As a result, many previous synthesis algorithms are not admissible, but only asymptotically admissible.

Next we examine some of the existing texture synthesis algorithms in graphics and see how they trade off admissibility, effectiveness and synthesis speed.

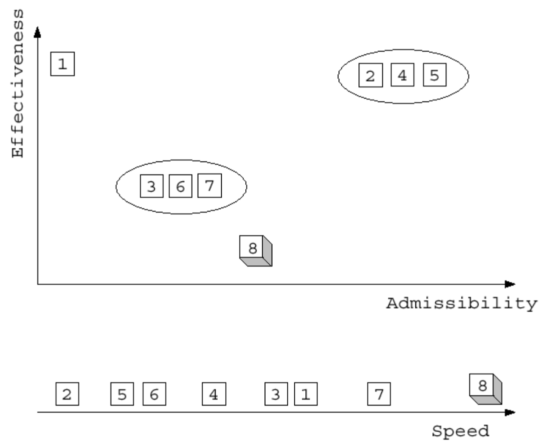


Figure 4: The trade-off of admissibility, effectiveness, and sampling speed of a number of existing texture synthesis algorithms. The algorithms are numbered as follows: (1) = (Heeger and Bergen, 1995), (2) = (Zhu, Wu, and Mumford, 1996), (3) = (DeBonet 1997), (4) = (Portilla and Simoncelli, 2000), (5) = (Zhu, Liu, and Wu, 2000), (6) = (Efros and Leung 1999), and (7) = (Wei and Levoy, 2000). The 3D box is for the patch pasting algorithm proposed in this paper.

3.2 Sampling algorithms for texture synthesis

There are mainly two sampling strategies used by current texture synthesis algorithms. The first strategy, called *ensemble sampling* for short, is to compute global statistics in feature space and thereby sample images from the texture ensemble directly on a large image lattice. The second strategy is to compute local conditional Gibbs distributions and thus synthesize pixels incrementally.

Algorithms using the first sampling strategy include (Heeger and Bergen, 1995), (Debonet, 1997), (Zhu, Liu, and Wu, 2000), and (Portilla and Simoncelli, 2000). When using this strategy one samples texture images from an ensemble, and usually one may sacrifice the effectiveness for fast speed of synthesis or for ease of sampling. For example, (Heeger and Bergen, 1995) has difficulties in synthesize textures with distinguishable features. This is not really because the marginal statistics they used are not enough for these textures (which was argued by some people in the literature), but the algorithm cannot match these statistics exactly. That is, their pyramid collapse method was sampling from an ensemble which is not a subset of $\Omega(\mathbf{h})$ as it supposed to be. Then (Debonet, 1997) used a similar algorithm on a pyramid. Since De Bonet used a much larger amount of statistics, i.e. the joint histogram of some > 50 filters (more than 50 dimensional histogram) that limits the ensemble of the algorithm within the Julesz ensemble for more textures.

Algorithms using the second sampling strategy include (Zhu, Wu, and Mumford, 1996, 1997), (Elfros and Leung, 1999) and (Wei and Levoy, 2000). The first method estimates the FRAME (Gibbs, or MRF) model using observed marginal histograms $\mathbf{h}(\mathbf{I}_{\Lambda_o})$ which are often easy to estimate accurately with Λ_o being reasonably big, say 128×128 pixels. The second and third methods estimate a one dimensional conditional MRF (FRAME or Gibbs) density $p(\mathbf{I}_v|\mathbf{I}_{\partial v})$ in a non-parametric method by an empirical histogram,

$$\hat{p}(\mathbf{I}(v)|\mathbf{I}_{\partial v}) = \sum_i \alpha_i \delta(\mathbf{I}(v) - \mathbf{I}_i(v)), \quad \sum_i \alpha_i = 1.$$

$v = (x, y)$ is a single pixel and $\mathbf{I}_i(v), \forall i$ are the intensities in the observed image under some similar boundary condition $\mathbf{I}_{\partial v}$. The weight α_i measures the similarity. In this work the statistics $\mathbf{h}^*$ is implicit in the neighborhood ∂v. Large neighborhood size means strong statistical constraints and small ensemble of the sampling algorithm. In comparison to the analytical FRAME model in (Zhu, Wu, and Mumford, 1997), this non-parametric method is faster to estimate although it is subject to much larger statistical fluctuations because in a small image $\mathbf{I}_{\Lambda_o}$ there may only be a few sites that have neighborhood which is similar to $\mathbf{I}_{\partial v}$.

Fig. 4 illustrates the trade-off between admissibility, effectiveness and synthesis speed for the above texture synthesis algorithms[3]. We qualitatively group these methods into three different groups with similar properties in terms of admissibility and effectiveness: group 1 with method 1, group 2 with method 3, 6 and 7 and group 3 with method 2, 4 and 5.

From our discussion of these algorithms it is easy to see that many of them are not admissible. For example, in (Elfros and Leung, 1999) the estimated MRF density $\hat{p}(\mathbf{I}_v|\mathbf{I}_{\partial v})$ is generally not the same as the conditional distribution of the Julesz ensemble $q(\mathbf{I}_v|\mathbf{I}_{\partial v}; \mathbf{h})$, although $p(\mathbf{I}_v|\mathbf{I}_{\partial v})$ does converge to $q(\mathbf{I}_v|\mathbf{I}_{\partial v}; \mathbf{h})$ as Λ_o goes to infinity and (Elfros and Leung, 1999) is asymptotically admissible. To measure the admissibility, we can first set an estimation error tolerance ϵ, then compute the size of the observed lattice Λ_0 needed to achieve this error tolerance as discussed before.

[3]Note that the comparison here is mostly qualitative. The quantitative comparison is hard in practice because of the variability of feature statistics used by different algorithms explicitly or implicitly.

Referring to Fig. 4, we have (Heeger and Bergen, 1995) at one end of the admissibility axis. Although their algorithm uses 1D marginal histograms which can be estimated quite accurately, the algorithm cannot guarantee close match of the statistics. (De Bonet 1997) is less admissible than the algorithm of (Elfros and Leung, 1999) and (Wei and Levoy, 2000) because the estimation of joint histogram in high dimension could generate very large variance. At the other end of the admissibility axis, (Zhu, Wu, and Mumford, 1996) loses some admissibility to (Zhu, Liu, and Wu, 1999) because the former introduces computational error in learning the Gibbs model.

The analysis of the effectiveness of the algorithms in Fig. 4 is similar to the admissibility analysis of these algorithms. Generally speaking, the more statistics an algorithm uses, the less flexible the the algorithm is and its effectiveness suffers as a result. For example, (Heeger and Bergen, 1995) has a higher effectiveness ranking than (De Bonet 1997) and (Elfros and Leung, 1999) because (Heeger and Bergen, 1995) uses only marginal histogram and it samples the histogram which has greater entropy than those used by (De Bonet 1997) and (Elfros and Leung, 1999).

Fig. 4 not only provides a qualitative way to compare the existing algorithms but also suggests how to design new algorithms. As mentioned earlier, the ensemble sampling strategy potentially allows us to achieve fast synthesis speed by compromising effectiveness. Unfortunately, the fastest algorithm using the ensemble sampling strategy, (Heeger and Bergen 1995), has low admissibility, which is a big drawback for graphics applications since the design criteria for texture synthesis in graphics dictate the importance of admissibility over effectiveness. Can we achieve fast speed without big sacrifice in admissibility ? We answer this question affirmatively by presenting an extremely fast algorithm that has average admissibility.

3.3 Texture Synthesis by Patch Pasting

We will now present our algorithm which can be used to synthesize high-quality textures extremely fast with some compromise in effectiveness.

This algorithm, first appeared in [13] as a technical report, has been successfully used in texture mapping 3D surfaces (Praun, Frinkelstein, and Hoppe, 2000)[10].

Suppose we partition the lattice of synthesis Λ into a set of n disjoint patches,

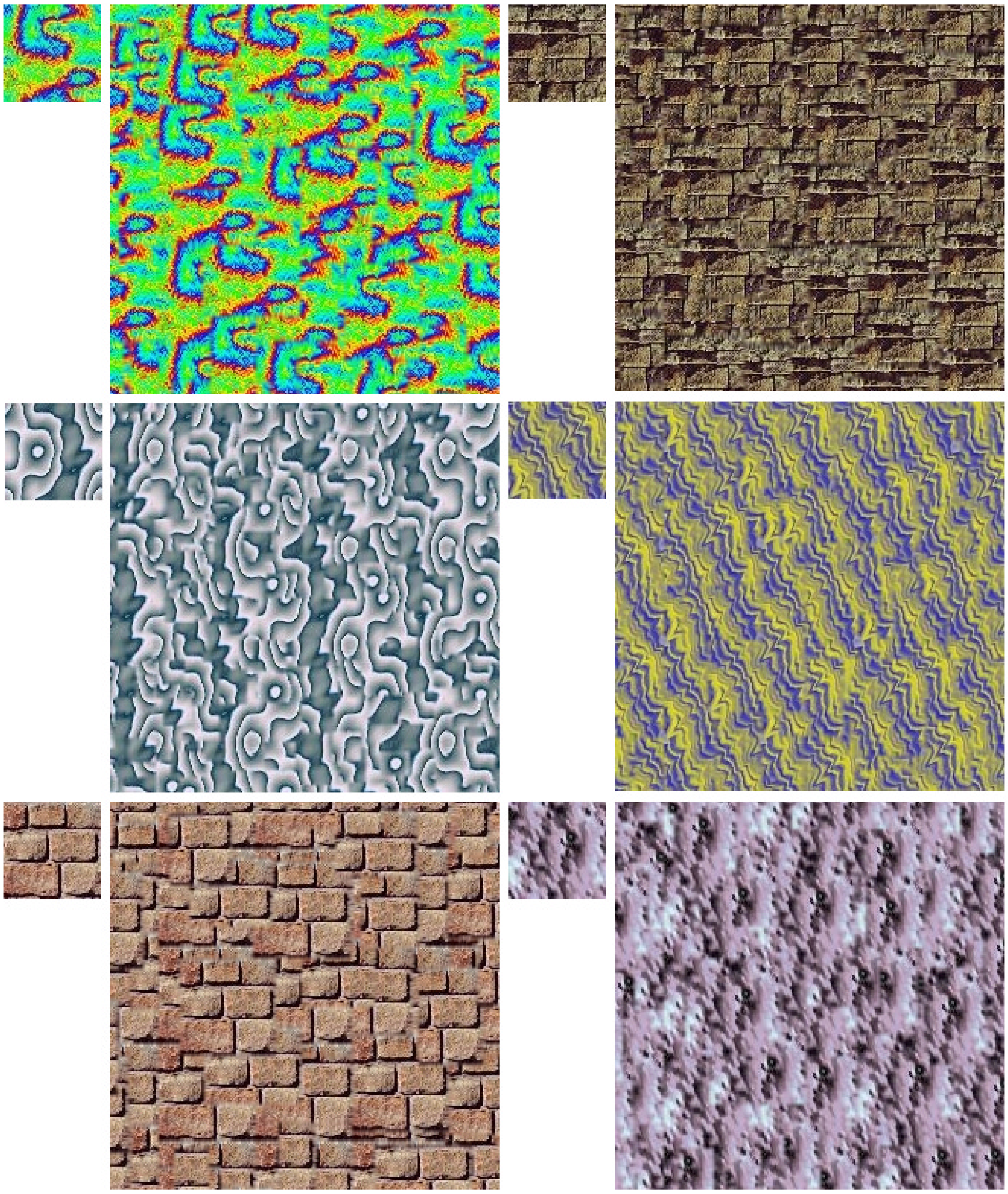


Figure 5: Texture synthesis results. The input samples are of size 64×64. The synthesized textures are of size 256×256.

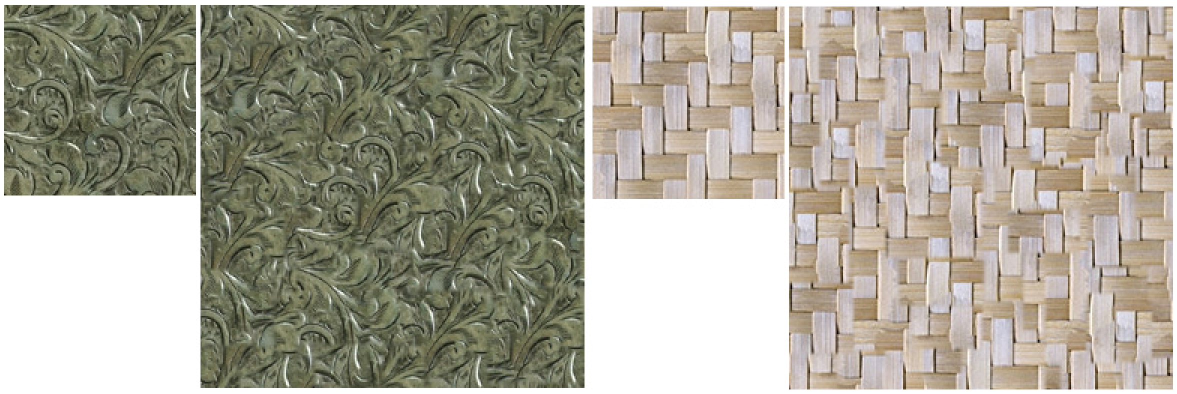


Figure 6: More texture synthesis results. The input samples are of size 128×128. The synthesized textures are of size 256×256.

and denote each partition as π.

$$\pi = \{R_i \ : i = 1, 2, ..., n\}, \quad \cup_{i=1}^{n} R_i = \Lambda, \text{ and } R_i \cap R_j = \emptyset.$$

Definition 8 *The* diameter $\phi(R)$ *of a patch* R *is the longest distance between two points in* R*. The diameter of a partition* $\phi(\pi)$ *is the largest diameter out of all the patches in* π*.*

$$\phi(\pi) = \max\{\phi(R) : \ R \in \pi\}.$$

The diameter and shape of these patches are constrained so that each of the region is smaller than the observed texture lattice Λ_o, i.e., $R_i \subset \Lambda_o$ after a translation. We denote by Ω_π the set of all possible partitions with various n.

$$\Omega_\pi = \{\pi \ : R_i \subset \Lambda_o; \forall R_i \in \pi\}.$$

Definition 9 *For each patch* $R \in \pi$*, we decompose it into* interior R^I *and boundary* ∂R^I*, so that for each pixel* $v \in R^I$*, its statistics features (filter responses) can be computed within* R*, while pixels in* ∂R^I *involves pixels outside* R^I*.*

Figure 7 illustrates a partition π with the interiors of the patches shown by blue color and the boundary conditions by the red color.

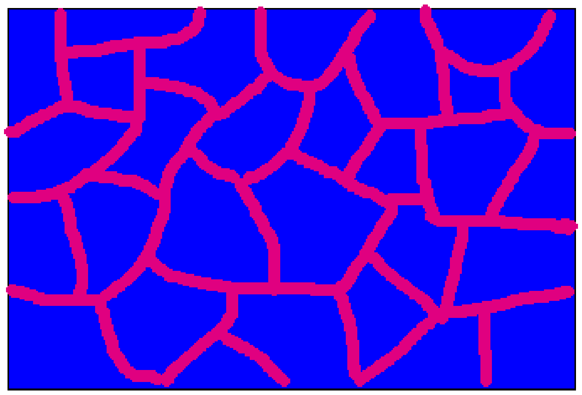


Figure 7: The partition of images

The patch-pasting algorithm works as follows.

Step 1: Randomly draw a partition $\pi \in \Omega_\pi$, $d_1 \leq \phi(\pi) \leq d_2$. The sizes d_1 and d_2 affect the effectiveness and speed of the synthesis.

Step 2: For each $R \in \pi$, repeat the following

a) Randomly draw a point $v \in \Lambda_o$ so that $v \in R$ after a translation.

b) Paste $\mathbf{I}_{\Lambda_o}$ inside R.

In practice, we first randomly draw a patch of irregular shape from Λ_o and then paste it onto Λ at a random location with occlusion between patches. To randomly paste texture patches over Λ while reducing overlap of the patches, we control the pasting locations using Arnold's Cat Map from the field of deterministic chaos [13]. To reduce the seams between pasted texture patches, we apply alpha-blending at the edges of pasted patches.

Algorithm Analysis: This simple algorithm does not correctly handle the boundary conditions. Nevertheless, the algorithm is asymptotically admissible. Suppose $\mathbf{h}^*$ is the statistics extracted by human vision, and $\mathbf{h} = \mathbf{h}^*(\mathbf{I}_{\Lambda_o})$ is the observed statistics, and thus the Julesz ensemble of this texture is $\Omega_\Lambda(\mathbf{h})$. The statistics of the synthesized image is a mixture of statistics from the interior and boundary,

$$\mathbf{h}(\mathbf{I}_\Lambda) = \sum_{i=1}^{n} [\ \frac{|R_i^I|}{|R_i|}\mathbf{h}(\mathbf{I}_{R_i^I}) + \frac{|\partial R_i^I|}{|R_i|}\mathbf{h}(\mathbf{I}_{\partial R_i^I}).$$

Texture Size	T(HB)	T(PP)	Mem(HB)	Mem(PP)
400×400	157	0.026	8Mb	4.8Mb
512×512	278	0.042	12Mb	5.0Mb
800×800	579	0.101	27Mb	6.2Mb
1024×1024	1112	0.151	44Mb	7.4Mb

Table 1: Timing and memory usage comparison between the patch-pasting method and (Heeger and Bergen, 1995). The "T(HB)" column lists timings of (Heeger and Bergen, 1995) in seconds. The "T(PP)" column lists timings of the patch-pasting algorithm in seconds. The "Mem(HB)" and "Mem(PP)" report the memory usages of (Heeger and Bergen, 1995) and the patch-pasting method respectively.

For large observation Λ_o, the interior regions can be made big enough so that the boundary statistics are negligible, i.e. $\frac{|\partial R_i^I|}{|R_i|}$ goes to zero and thus the algorithm becomes admissible.

Beside the problems at the patch boundaries, the patch-pasting algorithm does not suffer other admissibility problems. This makes the algorithm more admissible than (Efros and Leung, 1999), which estimates conditional probabilities in the order of 20x20 dimensions and can have very large estimation errors.

In terms of effectiveness, the volume of the ensemble is in the order of $O(\ |\Omega_\pi| \cdot m^n\ )$, where n is the largest number of region in a partition, and m is the number of different patches in $\mathbf{I}_{\Lambda_o}$ that can fill a region. We can sacrifice effectiveness to improve the speed by using large patches, since m, n and Ω_π all decrease when the patch sizes increase. In an extreme case, we use a tiling method, where the baseset of the ensemble has only one element[4]! It is the fastest algorithm one can get, but it is the most ineffective algorithm for texture sampling.

3.4 Experimental Results

Synthesis Speed: To compare our synthesis speed with existing texture synthesis methods, we have implemented (Heeger and Bergen, 1995), which is one of the faster methods. Table 1 provides the statistics for synthesizing textures of various sizes from an 128×128 input texture sample. These statistics were gathered on a

[4]The tiling method in the image space corresponds to a single point inside the Julesz ensemble (Figure 3).

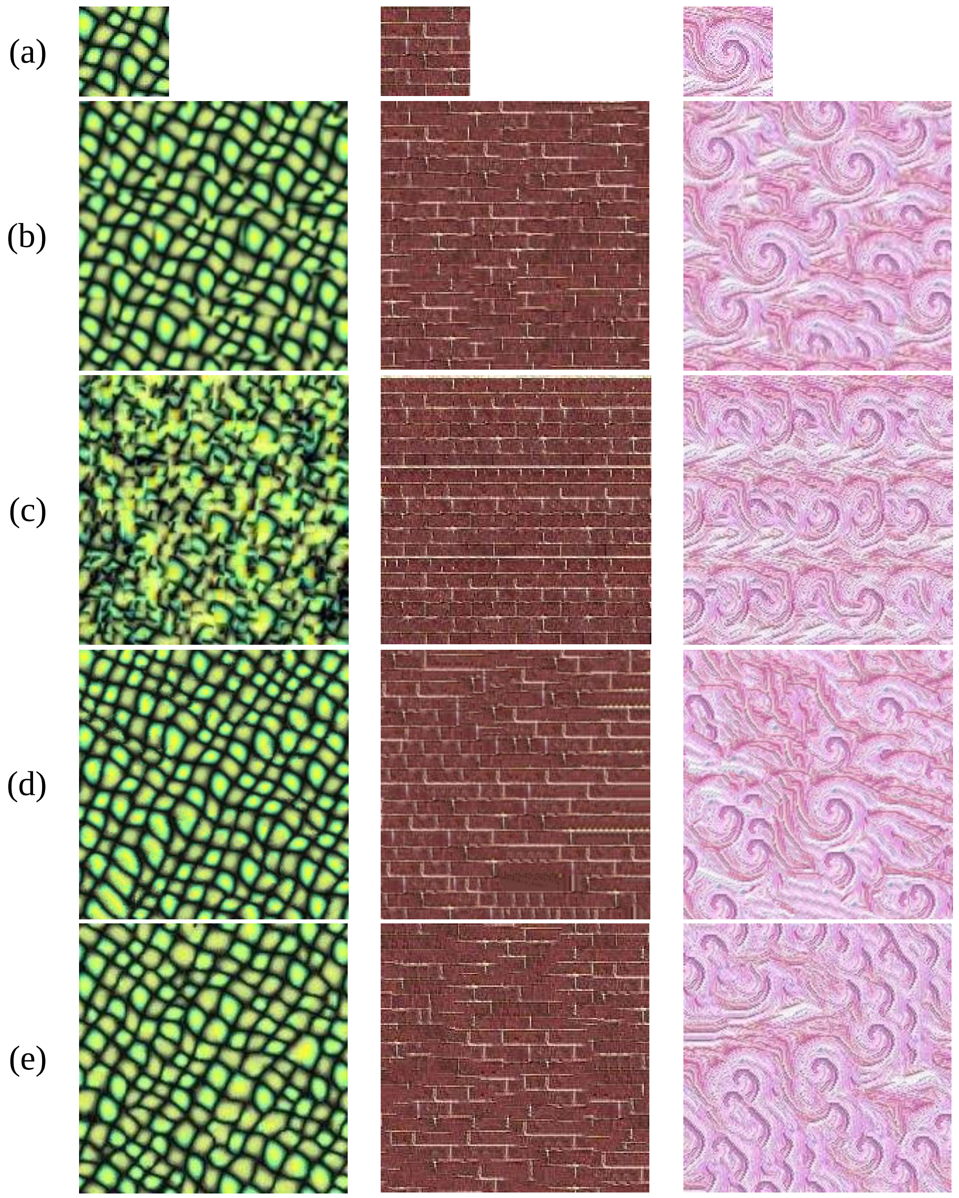


Figure 8: Comparison of the patch-pasting method and previous methods on three different input texture patterns. (a) Samples; (b) our patch-pasting method; (c) De Bonet 1997; (d) Wei and Levoy 2000; (e) Efros and Leung 2000.

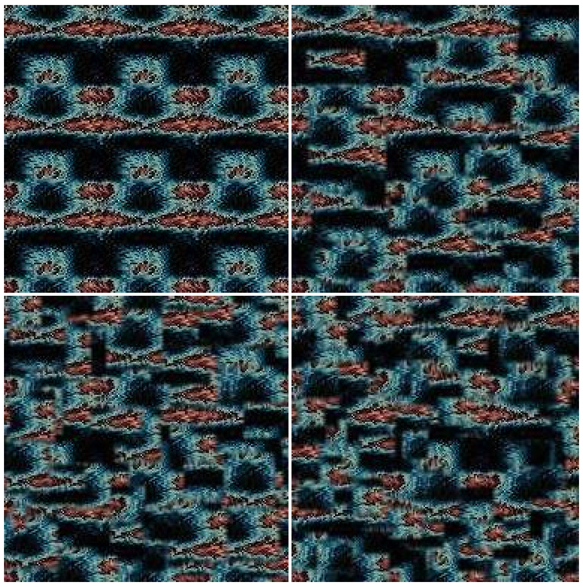


Figure 9: The effect of patch size on the randomness of the texture. From left to right, the first texture is a tiling obtained with the maximum patch size and the other textures are obtained with increasingly smaller patch sizes.

PC with a 450 Mhz Pentium III processor and 128 Mb of main memory. Roughly speaking, the patch-pasting method is about 6,500 times faster than (Heeger and Bergen, 1995). Notice that as the the size of the synthetic image increases, the memory usage grows much faster for (Heeger and Bergen, 1995) than for the patch pasting.

Texture Quality: Fig. 5 and Fig. 6 show some of our texture synthesis results. In terms of quality, both (Heeger and Bergen, 1995) and the patch-pasting algorithm work well on noisy textures with non-distinguishable features. When the input sample texture has distinguishable features, (Heeger and Bergen, 1995) ceases to be effective while the patch-pasting algorithm continues to produce good results. We shall compare the patch-pasting method with (De Bonet, 1997) and (Wei and Levoy, 2000), which are more successful in capturing distinguishable features.

In Fig. 8 compares the results the patch-pasting method, (De Bonet, 1997), and (Wei and Levoy, 2000) on three different input sample textures. In top row, (Wei and Levoy, 2000) generates the best results, whereas (De Bonet, 1997) fails to capture the local features of the texture. In the middle row, (De Bonet, 1997) performs better than the patch-pasting method as well as (Wei and Levoy, 2000).

In the bottom row, all three methods generate satisfactory results, with the result of patch-pasting method most resembling the given sample texture.

Controlling Patch Size and Randomness:

The patch pasting algorithm has a parameter, i.e. the diameter of the partitions which shall be selected once only off-line. It is a trade-off between admissibility and effectiveness. Intuitively, the patch size should be chosen as the smallest upon it produces textures with statistics error below a perception threshold ϵ

```
Step 1:Choose a set of sufficient statistics $\mathbf{h}^*$.
Step 2:Start with a small size $A$ and repeat the following.
    a) Synthesize $\mathbf{I}_\Lambda$ with size $A$.
    b) Compute $error = |\mathbf{h}^*(\mathbf{I}_\Lambda) - \mathbf{h}^*(\mathbf{I}_{\Lambda_o})|$.
    c) If $error \leq \epsilon$ then output $A$, otherwise $A \leftarrow A + \delta$.
       Otherwise stop.
```

Fig. 9 shows the effect of patch size on the randomness of the texture. When the entire input sample texture is one patch, we get a tiling. As the patch size becomes increasingly smaller, the synthesized texture becomes more random. This is because a smaller patch captures smaller local features of the input sample texture and randomly distributing these features in the synthesized texture increasing its randomness.

3.5 Texture Synthesis by Patch-based Sampling

Taking a close look at the examples in which the patch-pasting perform poorly, we can see that the main problem is the mismatched features across the patch boundaries. We have extended our patch-pasting method to match boundary statistics to achieve seamless texture synthesis. The method, called *patch-based sampling*, is described in detail in our technical report [7]. Similar idea has also been independently developed by Erfos and Freeman [4]. Figure 10 shows some of the examples by our patch-based sampling approach[5].

[5]More examples are available at the following web site $www.cis.ohio - state.edu/oval/Texture/MSR_Texture/Homepage/datahp1.htm.$

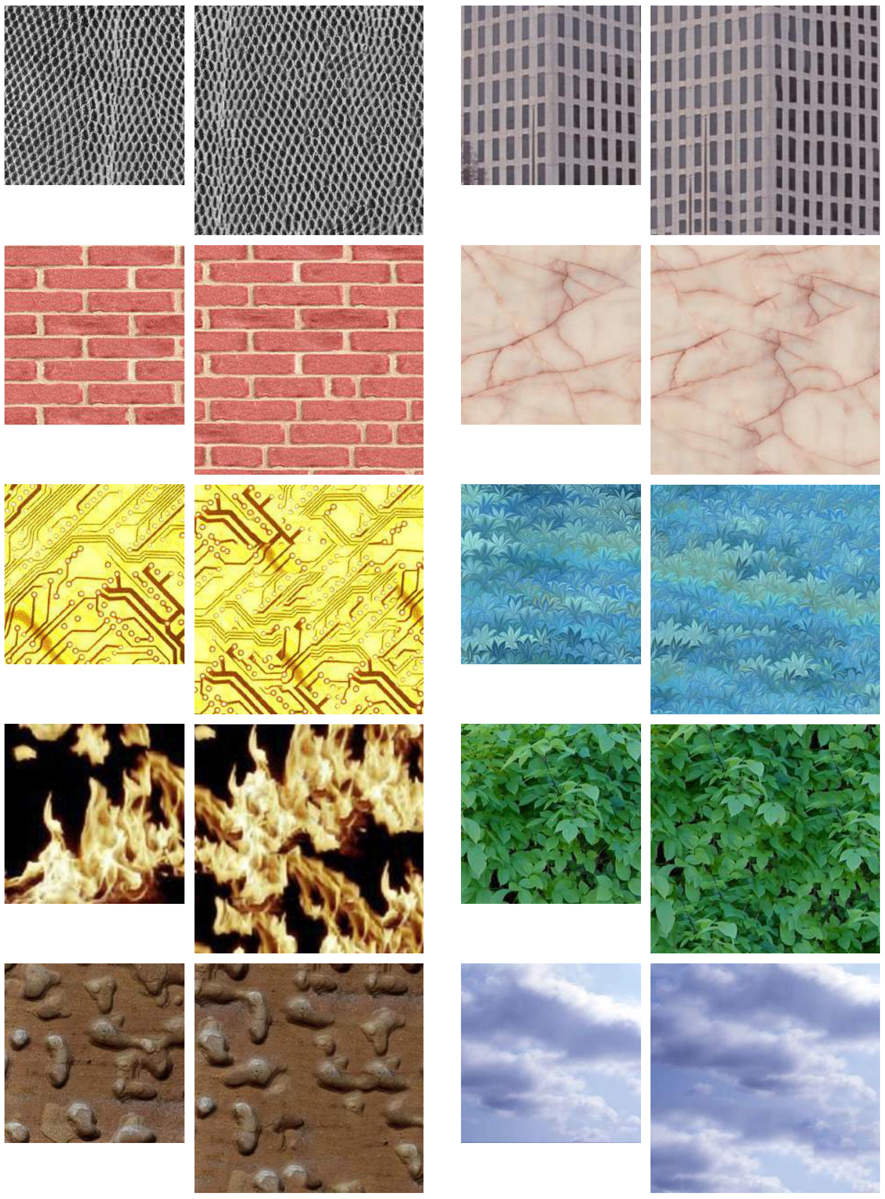


Figure 10: Texture synthesis examples by combining patch-pasting and boundary matching.

4 Summary and Discussion

In this paper, we have introduced a theoretical framework for designing and analyzing texture sampling algorithms. Specifically, we measure a texture sampling algorithm by its admissibility, effectiveness and sampling speed. After analyzing existing texture sampling algorithms, we propose different design criteria for texture analysis algorithms in vision and texture synthesis algorithms in graphics. We have developed a novel texture synthesis algorithm which simply pastes texture patches from the sample texture, yet generates good quality results. In particular, we combine patch pasting with boundary matching to obtain seamless synthesized textures. Our algorithm can synthesize high-quality textures extremely fast.

Texture images are record of the effects erected by natural stochastic processes in the physical world. Texture perception in human vision intends to characterize texture phenomena with a certain "purpose". Thus it summarizes texture by statistics $\mathbf{h}$. Procedural (physics based) texture synthesis algorithms intended to simulate the physics, while statistics based algorithms approximate human perception. The latter are obviously more general.

Almost all texture modeling and synthesis are based on the notion that perception extracts some essential statistics. We believe this assumption is just a first order approximation to human perception. In future work, we should study texture algorithms that capture the internal representations in perception beyond feature statistics. This should lead to hierarchical texture models.

References

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

[2] J. S. De Bonet. Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Image. In *Computer Graphics Proceedings, Annual Conference Series*, pages 361–368, August 1997.

[3] A. A. Efros and T. K. Leung. Texture Synthesis by Non-Parametric Sampling. In *Proceedings of International Conference on Computer Vision*, 1999.

[4] A. Erfos and W. Freeman. Quilting for Texture Synthesis and Transfer. In *To Appear at Siggraph2001*, August 2001.

[5] D. J. Heeger and J. R. Bergen. Pyramid-Based Texture Analysis/Synthesis. In *Computer Graphics Proceedings, Annual Conference Series*, pages 229–238, July 1995.

[6] B. Julesz. Visual Pattern Discrimination. *IRE Transactions of Information Theory*, (IT8):84–92, 1962.

[7] L. Liang, C. Liu, Y. Q. Xu, B. Guo, and H. Y. Shum. Real-time Texture Synthesis by Patch-based Sampling. In *Microsoft Research Technical Report MSR-TR-2001-40*, April 2001.

[8] K. Popat and R.W. Picard. Novel cluster-based probability model for texture synthesys, classification, and compression. In *Proceedings of SPIE Vis. Comm.*, 1993.

[9] J. Portilla and E. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. Journal of Comp. Vision*, (40(1)):49–71, 2000.

[10] E. Praun, A. Finkelstein, and H. Hoppe. Lapped Texture. In *Computer Graphics Proceedings, Annual Conference Series*, pages 465–470, July 2000.

[11] L. Y. Wei and M. Levoy. Fast Texture Synthesis Using Tree-Structured Vector Quantization. In *Computer Graphics Proceedings, Annual Conference Series*, pages 479–488, July 2000.

[12] Y. N. Wu, S. C. Zhu, and X. W. Liu. Equivalence of Julesz Ensemble and FRAME Models. *Int'l Journal of Computer Vision*, 38(30):245–261, 2000.

[13] Y. Q. Xu, B. Guo, and H. Y. Shum. Chaos Mosaic: Fast and Memory Efficient Texture Synthesis. In *Microsoft Research Technical Report MSR-TR-2000-32*, April 2000.

[14] S. C. Zhu, X. Liu, and Y. Wu. Exploring Texture Ensembles by Efficient Markov Chain Monte Carlo. *IEEE Trans. on PAMI*, 22(6), 2000.

[15] S. C. Zhu, Y. Wu, and D. B. Mumford. Minimax Entropy Principle and Its Application to Texture Modeling. *Neural Computation*, (9):1627–1660, 1997 (first appeared in CVPR96).