User Constrained Multiscale MRF Model for Texture Mixture Synthesis and its Application to Texture Replacement

Xuejie Qin Yee-Hong Yang
{xuq, yang}@cs.ualberta.ca
Department of Computing Science
University of Alberta

Abstract

The original multiscale MRF texture model proposed by Paget (IEEE Transactions on Image Processing, 1998, page 925-931) can be used to synthesize a broad range of textures but is limited to taking a single input texture and outputting a homogeneous texture similar to the input. This is insufficient for textures that have combined visual characteristics from several different sources. To address this problem, this paper presents a new method, called User Constrained multiscale MRF model, for synthesizing a new texture mixture from multiple input textures. Since the Gibbs sampler and exhaustive search are used in the original multiscale MRF model, a brute force implementation of the algorithm is slow. To overcome this problem, an existing fast neighborhood search technique is adapted for our model, and the run time is decreased by a factor of 500-1000. We also demonstrate that our method can be used in texture replacement. The experimental results show that our algorithm performs well in the quality of results.

Key words: *Texture synthesis, texture mixture, texture replacement, Markov random field, local conditional probability function.*

1 Introduction

In this paper, a texture mixture is considered as a synthesized texture that has the appearance of two or more input textures. In the literature, there is not much research done related to texture mixtures, and only a handful of methods have been published. In Heeger and Bergen's paper [14], they give a brief discussion at the end of their paper on how to generate texture mixtures using their histogram-matching method with steerable pyramids. Unfortunately, their approach fails to generate satisfactory results. In Bar-Joseph et al.'s work [2], a statistical learning tree is used to generate texture mixtures from multiple sources. The limitations of their approach include the substantial memory requirement for the learning trees and the blurring effects in the output textures if the input textures are different from each other. In Portilla and Simoncelli's work [27], a wavelet based statistical model is presented for texture analysis and synthesis. As an application of their model, they show that texture mixtures could be generated by sampling linear combinations of global statistics measured at wavelet subbands of input textures. It is obvious that a texture mixture generated by their approach is a simple blend of input textures because linear interpolation between global parameters of the input textures is employed. Wei in his Ph.D. thesis [31] addresses the problem of texture mixtures within the framework of multiple source texture synthesis, which is an extension of his texture synthesis approach for single input texture [30]. One problem of Wei's approach is that the algorithm generates blurred edges of structural textures (see Figure 2). The second problem of Wei's approach, which is also a limitation of all the other approaches described before, is that it is difficult for the algorithm to incorporate user constraints which is crucial for mixing textures because, otherwise the results are difficult to adjust according to user's needs [1, 15, 23].

To address the above problems, in this paper, we present a new user constrained multiscale MRF model for generating textures from multiple input sources, which is an extension of the original multiscale MRF texture model for single input texture synthesis proposed by Paget [25].

Given two or more input textures, our algorithm generates a texture mixture by synthesizing each pixel of the output image with user-specified constraints, which is specified as a look-up table with the same size as the output texture, each entry of which specifies how the corresponding pixel in the output texture is sampled from the input textures. For each pixel in the output image, our algorithm first looks up the table to get the local constraint information on that pixel, then calculates the local conditional probability density function (LCPDF) from the input textures with the specified constraint, and finally samples a value of the pixel based on the calculated LCPDF using stochastic relaxation such as the Metropolis algorithm, the Iterative Conditional Modes (ICM) algorithm, or the Gibbs sampler [12, 26].

Because of the expensive calculation of the LCPDFs during sampling, a straightforward implementation of

the algorithm is slow. We demonstrate that the sampling process based on the ICM algorithm can be approximated by a nearest neighborhood search process. Therefore, a fast neighborhood search algorithm, for example, the one described in Ashikhmin's paper [1] can be adapted to speed up the algorithm. Finally, we demonstrate in the experimental section that our model can be used in the application to texture replacement.

In summary, the contributions of this paper include: (1) a new user constrained MRF model for generating texture mixtures, which is an extension to the original multiscale MRF model by Paget [25], (2) a fast implementation of the algorithm, and (3) the application of our model to texture replacement.

The rest of the paper is organized as follows. The related work is described in the next section. In Section 3, we describe some background knowledge on MRF texture models. In Section 4, we present our general user constrained multiscale MRF model for texture mixtures. In Section 5, we discuss about the implementation and acceleration of our algorithm. In Section 6, we present the experimental results on both texture mixtures and texture replacement. The limitations of our current implementation are also discussed. Finally, the conclusion and discussion are presented in Section 7.

2 Related Work

Our work is closely related to texture synthesis. The earliest technique for generating synthetic textures is texture mapping [13], which is still an active research topic (e.g. [28, 33], etc). However, texture mapping suffers the problems of distortion, discontinuity, and unwanted seams. To solve those problems, procedural texturing (i.e. solid texturing) [8, 20] can be used, in which procedures are developed to generate synthetic textures without requiring input textures. By calling a compact procedure, textures are generated directly on surfaces of 3D objects without seams and without discontinuity.

Another important technique is called texture analysis and synthesis using statistical approaches. Followed Julesz's pioneer work [17], several statistical texture models have been proposed such as, to name a few, the MRF model [6, 12], the FRAME model [35], the parametric texture model based on joint statistics of complex wavelet coefficients [27], and the cooccurrence-based texture models [24]. Other well known statistical approaches includes the multiresolution histogram-matching [14], the feature pyramid sampling procedure [4], the non-parametric sampling [9], the fast texture synthesis [31], and etc.

In the above approaches, the synthesis process is performed at the pixel level of images, thus, in general, they are slow. For real time applications, there are patch-based texture synthesis approaches (e.g. [10, 19, 22], etc), in which synthesized textures are generated by sampling small patches of input textures. It is noted that image-based rendering techniques can also be used for texture synthesis such as, to name a few, Bernardini et al.'s image-based registration technique [3] for generating high-quality textures from multiple scans, Veryovka et al.'s and Fung et al's techniques for generating artistic style textures [11, 29]. There are also recent works (e.g. [21, 32, 34], etc) on synthesizing textures onto 3D surfaces, and we shall not review them in this paper because of page limitations.

Although our work has some resemblance to that of Ashikhmin's [1], Hertzmann's [15], Kim and Pellacini's [18], and Liu et al.'s [23], ours is different from theirs in some significant points. In both Ashikhmin's and Hertzmann's work, different types of textures in the output image are essentially the component textures from the same single input source image; while in our work, different types of textures in the output image are from different input texture images. In Kim and Pellacini's work [18], instead of using multiple input textures, they use a set of image tiles (image of objects) to fill a user specified container image. Thus their approach is best for image mosaics. In Liu et al.'s work [23], user selected patterns in the source image are used to replace all similar target patterns in a given image. In essence, their approach is more appropriate for texture editing [5].

3 MRF Models

The MRF models are important techniques in computer vision, graphics, and image processing. The underlying theory of MRF texture models is that the information at a pixel location is dependent on the information of its neighboring pixels [6, 12, 25].

In MRF models, an image is modeled as a random field X defined on a finite rectangular lattice *S*. Let X_s be a *random variable* at site s. The *random field* $\boldsymbol{X}$ on S is the set of all random variables X_s, i.e. $\boldsymbol{X} = \{X_s \mid s \in S\}$. The set of all possible values of X_s is called the *state space* of *s*, which is denoted by Λ_s. In this paper, we assume a common discrete state space for all *s*, i.e. $\Lambda_s = \Lambda = \{0,1,...,255\}$ for all $s \in S$. The *configuration space* on $\boldsymbol{X}$ is denoted by Ω, which is defined as $\Omega = \prod_{s \in S} \Lambda_s$. A *joint probability* on Ω is denoted by P and the *Local Conditional Probability Density Function* (LCPDF) at site $s \in S$ is given by $LCPDF(s) = P(X_s = x_s \mid X_r = x_r, r \neq s)$, which implies that the probability of site s having pixel value

$x_s \in \Lambda_s$ depends on the pixel values of its neighboring pixels.

The *neighborhood* of s with *order* d, denoted by N_s^d, is given by $N_s^d = \{r \in S \mid 0 < |r - s|^2 \le d\}$. The set of all neighborhoods N_s^d is called a *neighborhood system* of order d on S, which is denoted by $\boldsymbol{N} = \{N_s^d \mid s \in S\}$. The neighborhood system has two important properties: for any $s, t \in S$, (1) $s \notin N_s^d$, and (2) $s \in N_t^d$ if and only if $t \in N_s^d$.

Mathematically, an MRF is defined as a random field $\boldsymbol{X} = \{X_s \mid s \in S\}$ with a joint probability function P defined on it, which has the following three properties:

(*i*) $P(\boldsymbol{x}) > 0, \forall \boldsymbol{x} \in \Omega$

(*ii*) $\sum_{x \in \Omega} P(\boldsymbol{x}) = 1$

(*iii*) $LCPDF(s) = P(x_s \mid x_r, r \in N_s^d), \forall \boldsymbol{x} \in \Omega, \forall s \in S$

Properties (*i*) and (*ii*) ensure that P is a probability distribution with a positive value for any configuration $\boldsymbol{x} \in \Omega$. Property (*iii*) is defined on a neighborhood system, which ensures that the random field is an MRF, i.e. the LCPDF of a given site s in a given image $\boldsymbol{x}$ can be calculated using the information of its neighboring pixels.

Based on the equivalence between the MRF and Gibbs distribution established by the Hammersley-Clifford Theorem [12], many MRF models have been proposed in the literature (see [26] for a complete survey). In the next section, we describe our new user constrained multiscale MRF model for texture mixtures.

4 Proposed Model

The simplest and commonly used MRF models are the auto-models based on clique structures. Since clique structures cannot efficiently capture texture features, the auto-models can only model very limited types of textures [26, 35]. To model a wide range of textures, a multiscale representation can be incorporated into MRF models, which is described in the next subsection.

4.1 Multiscale MRF Texture Model

Given an image $\boldsymbol{X}$ of size $M \times N$, let S be the rectangular lattice on which the image $\boldsymbol{X}$ is defined, i.e.:

$$S = \{s = (i, j) \mid 0 \le i < M, 0 \le j < N\}. \quad (1)$$

The multiscale representation of image $\boldsymbol{X}$, is defined as a set of images $\boldsymbol{X}^l$, where $l \ge 0$ is the level index of image $\boldsymbol{X}^l$. For a given value of l, image $\boldsymbol{X}^l$ is defined on a lattice $S^l \subset S$, where,

$$S^l = \{s = (2^l i, 2^l j) \mid 0 \le i < M/2^l, 0 \le j < N/2^l\}. \quad (2)$$

For each level $\boldsymbol{X}^l$ of the multiscale representation of $\boldsymbol{X}$, the neighborhood N_s^l of $s = (2^l i, 2^l j) \in S^l$ is defined as:

$$N_s^l = \{r = (2^l p, 2^l q) \mid 0 < (p - i)^2 + (q - j)^2 \le d\}, \quad (3)$$

where d is the order of the neighborhood system.

It is noted that the image at level 0, i.e. $\boldsymbol{X}^0$ is the original image $\boldsymbol{X}$ since $S^0 = S$, and that S^{l+1} is a subset of S^l, i.e. $S^{l+1} \subset S^l$. To infer the intensity value of each pixel from the current level to the next level, we use a *local decimation* scheme [26]. For image $\boldsymbol{X}^{l+1}$, let x_s^{l+1} be the intensity value of the pixel at site $s = (2^{l+1} i, 2^{l+1} j) \in S^{l+1}$. By local decimation, we mean that the value of x_s^{l+1} is directly copied from x_s^l - the value of pixel s at level l in the image $\boldsymbol{X}^l$, i.e.,

$$x_s^{l+1} = x_s^l. \quad (4)$$

Consider the case of synthesis using a single input texture. Let $\boldsymbol{Y}$ be the sample input image, and $\boldsymbol{X}$ the synthesized output image, which is initialized as random noise at the beginning. The multiscale MRF synthesis algorithm first constructs a multiscale representation for each of $\boldsymbol{X}$ and $\boldsymbol{Y}$ using Eq. 1 - 4. Starting from the lowest resolution image $\boldsymbol{X}^l$, the algorithm employs stochastic relaxation (SR) to iteratively update each pixel value in $\boldsymbol{X}^l$ based on the LCPDF calculated from sample input image $\boldsymbol{Y}^l$ until an equilibrium state is reached. After $\boldsymbol{X}^l$ is synthesized, the algorithm goes to the next higher resolution image $\boldsymbol{X}^{l-1}$, applies SR on that level with the *constraint* that all pixel information from the previous level must be maintained at that level through the SR process. By the advantage of local decimation used in the multiscale representation described before, the constraint can be imposed by fixing the value of each pixel from the previous level, i.e. by setting $x_r^{l-1} = x_r^l$ for $\forall r \in S^l$, and the rest of pixels at $\boldsymbol{X}^{l-1}$ undergo a non-constraint SR.

The SR process uses the LCPDF to sample pixel value at each level. Let S_x^l be the lattice for output image $\boldsymbol{X}^l$, and S_y^l for input image $\boldsymbol{Y}^l$. Let $\boldsymbol{N}^l$ be a neighborhood system at level l. Note that for simplicity reasons we use the same neighborhood system configuration for both $\boldsymbol{X}$ and $\boldsymbol{Y}$. For each pixel site $s \in S_x^l$, let x_s be its pixel value, the LCPDF at s is calculated by:

$$P(x_s \mid x_r, r \in N_s^l) = \frac{1}{Q(s)} \sum_{\substack{p \in S_y^l \\ N_p^l \in S_y^l}} \exp\left[\frac{-1}{2h^2} \| z_{sp} \|^2\right], \quad (5)$$

where $Q(s)$ is the normalizing function, h the optimal Parzen window parameter [26], and z_{sp} a d-dimensional vector. The calculations of $Q(s)$, h, and z_{sp} are given as follows:

$$Q(s) = \sum_{\lambda_s \in \Lambda} \sum_{p \in S_y^l, N_p^l \subset S_y^l} \exp\left[-\frac{1}{2h^2} \| z_{sp} \|^2 \right], \quad (6)$$

$$h = \sigma \{ 4 / [n(1+2d)] \}^{1/(4+d)}, \quad (7)$$

$$z_{sp} = [x_s - y_p, (x_r - y_{r-s-p})(1-t_r), r \in N_s^l]^T, \quad (8)$$

where in Eq. 7, n is the size of the input image $\boldsymbol{Y}^l$, $d = |N_s^l| + 1$, and σ^2 is the variance of the histogram of image $\boldsymbol{Y}^l$.

In Eq. 8, t_r is the temperature at pixel r and is initialized to 1 for all pixels in the output image $\boldsymbol{X}$ at the beginning of the SR process. At the end of each iteration, the temperature t_s of a given target pixel $s \in S^l$ is updated by the following temperature function:

$$t_s = \max\{0, (-1.0 + \textstyle\sum_{r \in N_s^l} t_r) / |N_s^l|\}, \quad (9)$$

where $|N_s^l|$ is equal to the total number of pixels in the neighborhood N_s^l.

When the temperature value of a given pixel site s is equal to zero, then s is said to be in an equilibrium state. When all of the pixels in $\boldsymbol{X}^l$ are in the equilibrium state, image $\boldsymbol{X}^l$ is said to be in an equilibrium state, and the SR process stops for $\boldsymbol{X}^l$ and proceeds to the next resolution image $\boldsymbol{X}^{l-1}$.

4.2 User Specified Constraints

Given two or more input textures $\boldsymbol{Y}_1$, $\boldsymbol{Y}_2$, …, $\boldsymbol{Y}_m$, we want to synthesis a texture $\boldsymbol{X}$, which has the combined visual appearance of all the input textures. To enable multiple source texture synthesis, we incorporate user specified constraints into the multiscale MRF texture model described earlier. The purpose of incorporating user constraints is to tell the SR process which input textures to use, and in particular, how they are used to sample a given target pixel in the output texture.

A user specified constraint is expressed as a look-up table $\boldsymbol{T}$, which has the same size as the output texture image $\boldsymbol{X}$. Each entry in $\boldsymbol{T}$ is an m-dimensional vector consisting of m non-negative weights with values in the range of $[0,1]$. Each weight specifies the contribution from each input texture $\boldsymbol{Y}_i$ when calculating the LCPDF for a given target pixel using Eq. 5 during the SR process. Let $s = (i, j)$ be a target pixel in $\boldsymbol{X}$ to be synthesized. Let $\boldsymbol{w}_s = (w_{si})_{1 \le i \le m}$ be the table entry at location $s = (i, j)$ in the look-up table $\boldsymbol{T}$. Let P_{si} be the LCPDF calculated from $\boldsymbol{Y}_i$ using Eq. 5 - 8, then the weighted sum of P_{si}, which is given by Eq. 10 below, is used by the SR process to sample a value of s:

$$P(x_s \mid x_r, r \in N_s) = \sum_{1 \le i \le m} w_{si} P_{si}. \quad (10)$$

4.3 Algorithm

As a summary of our user constrained multiscale MRF model for multiple source texture synthesis, we present the complete algorithm in pseudo code shown in Figure 1 for easy reference. In the next section, we discuss some issues on the implementation and a new acceleration of the algorithm.

User Constrained Multiscale MRF Multiple Source Texture Synthesis

Input:
$\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_m \leftarrow$ training texture images.
$\mathbf{T} \leftarrow$ user constrained table.
Output:
$\mathbf{X} \leftarrow$ the synthesized texture mixture.

Begin
1. Initialize **X** as a random noise image, for each pixel s in **X**, set its initial temperature $t_s = 1.0$.
2. $L \leftarrow$ the number of levels of the multiscale.
3. Build the multiscale $\{\mathbf{X}^l, S_x^l\}$ for **X** using Eq. 1-4.
4. Build the multiscale $\{\mathbf{Y}_i^l, S_{iy}^l\}$ for each $\mathbf{Y}_i$ using Eq. 1-4, $0 \le i \le m$.
5. For $l = L - 1$ to 0 **do**
  While $\mathbf{X}^l$ is not in the equilibrium state **do**
    5.1. Randomly choose any site s in S_x^l for which $t_s > 0$.
    5.2. For $j = 0$ to m **do**
      5.2.1. Calculate the LCPDF P_{sj} for site s using sample image $\mathbf{Y}_j$ by Eq. 5-8.
    5.3. Calculate the user constrained LCTDF P for s using Eq.10.
    5.4. Choose new pixel value x_s for s by sampling the LCPDF calculated from 5.3 using Gibbs sampler or ICM algorithm.
    5.5. Update site s's temperature t_s using Eq.9.
  End of while
End of for
End of begin

Figure 1: The user constrained multiscale MRF multiple source synthesis algorithm.

5 Implementation and Acceleration

Using a straightforward implementation of our algorithm shown in Figure 1, the run time performance is slow because of the expensive calculation of the LCPDF at each pixel site per iteration. To speed it up, we adopt the fast neighborhood search method as described in Ashikmin's paper [1] based on the following observations. A much better way to do this is discussed in the experimental section, which is inspired by Hertzmann's work on image analogies [15].

When the ICM algorithm [26] is used in the sampling process in step 5.4 (Figure 1), the pixel value at site s is updated by x_s with the largest LCPDF value. From Eq. 5 in the previous section, one can see that a pixel p in sample image $\boldsymbol{Y}$ with the closest neighborhood match gives the smallest negative number in the exponential part of the right side of Eq. 5, and thus gives the largest LCPDF value. This implies that the ICM-based sampling can be replaced by a sampling process based on the closest-neighborhood-match search, which can be implemented efficiently using fast neighborhood search methods. In our implementation, a fast neighborhood search algorithm similar to the one used in Ashikhmin's work on natural texture synthesis [1] is used. It is noted that the non-causal neighborhood is used in our algorithm instead of the causal neighborhood used in the original Ashikhmin's algorithm.

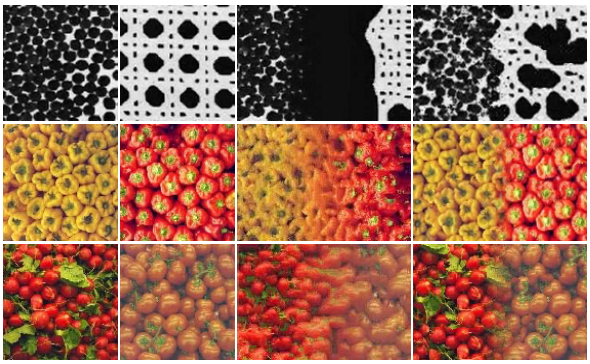
With respect to the running time of our algorithm, for two grey scale input images of size 64×64, the straightforward implementation of the algorithm takes about 4 hours on average to generate an output texture of size 128×128 on a 1.7GHz Penntium 4 PC running Windows XP Professional, while with the faster implementation, the algorithm takes less than one minute on average.

6 Experimental Results

In the first subsection, we present results for texture mixtures including comparison between that of ours with that of Wei's [31]. In the second subsection, we present some results in the application of texture replacement using our approach.

6.1 Texture Mixtures

Figure 2 gives some comparison results between our approach and Wei's approach [31]. In the figure, images in columns (a) and (b) are the input textures, images in (c) are the results from Wei's algorithm, and images in (d) are generated using our algorithm with the order of neighborhood $d=17$ and the user constrained table is $\boldsymbol{T}=(\boldsymbol{w}_{ij})$ with $\boldsymbol{w}_{ij}=(1,0)$ for all pixels in the left half of the images in (d), and $\boldsymbol{w}_{ij}=(0,1)$ for pixels in the right half of the images.



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

Figure 2: The results from Wei's approach are shown in column (c) and our approach in column (d), where images in column (a) and (b) are the input textures. The gray scale input images are taken from the Brodatz image set, and the color input images are taken from the Vistex image set. Images in column (c) are copied from Wei's thesis [31] with permission.

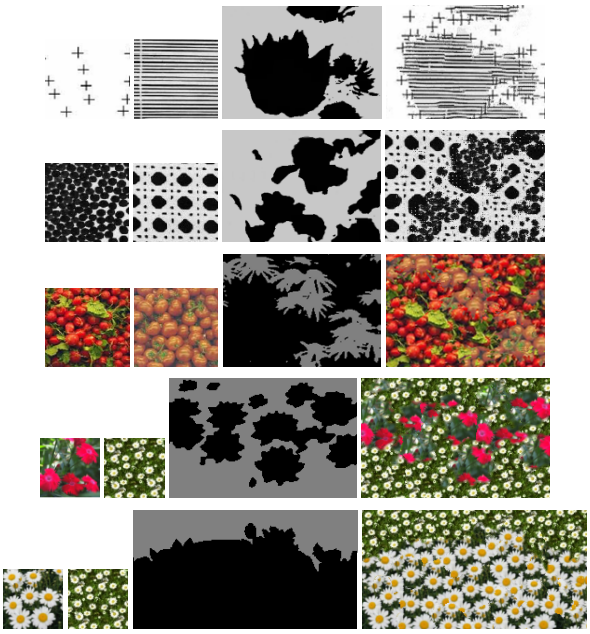


Figure 3: Results for different user constraints. In each row, the first two images are the inputs, the third image specifies the user constraint, and the last image is the synthesized texture mixture.

In other words, table $\boldsymbol{T}$ imposes a constraint such that the SR process samples only from image (a) for pixels in the left half of the output texture, and from image (b)

for pixels in the right half. One can see that the transition regions between two textures are blurred in results using Wei's approach, while they are not in our approach.

Figure 3 gives some texture mixture results with different user specified constraints. In each row of the figure, the first two images (from left to right) are the input textures, the last image is the texture mixture generated from the input textures with constraints specified by the third image.

6.2 Application to Tetxure Replacement

Tetxure replacement is usually considered as constrained texture synthesis [7, 9, 16]. We shall not consider the complex situation as that described in Drori's paper for image completion [7]. Instead, we consider texture images that contain flawed regions such as scratches, undesirable objects, or background textures to be replaced. Figure 4 gives some examples of texture replacement using our user constrained approach.

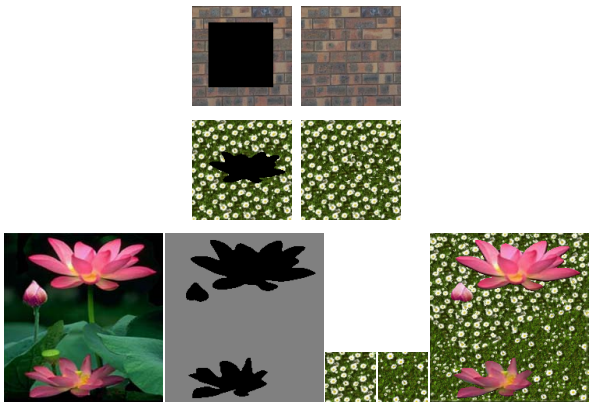


Figure 4: Results for texture replacements.

In each of the first two rows in Figure 4, the first image has a hole in it, which has to be filled by textures similar to the surrounding ones. This situation can be considered as a special case of our user constrained multiscale MRF texture mixture model. For example, to fill the hole in the flower image in the second row, we first take a flower patch in the flawed image. Then we synthesize the hole by starting at the boundary of the hole. We randomly choose a pixel on the boundary and synthesize it using our algorithm. After all pixels in the boundary have been processed, i.e. the old boundary is synthesized, we repeat the process on the new boundary until the hole is filled. Note that if one fills the hole in raster scan order or even randomly but without going outside to inside along the boundary of the hole, visible boundaries can be seen around the hole.

A more complex and interesting texture replacement example is illustrated in the last row of Figure 4. In this case, for example, we want the surrounding area (background) of the upper lotus flowers to be texture-replaced by one type of texture (e.g. the third texture image in the last row in Figure 4), while the surrounding area of the lower lotus flower to be texture-replaced by another type of textures (e.g. the fourth texture image in the last row in Figure 4), and the transition between the two types of replaced textures to be kept smooth with no visible seams. This can be done by performing our user constrained texture synthesis at two boundaries: the one surrounding the upper two lotus flowers and the other one surrounding the lower lotus flower, which is specified by the grey scale image in Figure 4. Like the hole-filling process described before, the constrained synthesis is performed on boundaries. However, instead of going from the outside to the inside, the direction of the synthesis process is going away from the object (the lotus flowers in Figure 4). The smooth transition between regions of different types of background textures are handled by specifying a smooth user constrained probability table.

6.3 Limitations of Current Implementation

All of the color images presented in this paper are generated by the fast version of our algorithm described in Section 5. In our current implementation, we have some limitations as follows.

The first one is due to the limitation of Ashikmin's algorithm [1]. For the fast neighborhood search, it is possible that there is no best neighborhood match found in the input texture for a given pixel in the synthesized texture. This is due to the following observation: during the synthesis process a pixel in the output image is chosen randomly (see step 5.1 in Figure 1), therefore, it is possible that none of its neighboring pixel has been synthesized (processed), and this causes an empty candidate list for the best neighborhood match searching [1]. In this case, our algorithm randomly choose a pixel in the input texture and copy its color to that of the target pixel. This introduces some random noises in the output textures (see Figure 3).

The second limitation is that some of our output textures presented in this paper have sharp edges (again see Figure 3). This is because the heuristic edge handling method as described in Ashikmin's paper [1] has not been implemented in our fast neighborhood search algorithm.

We are now investigating a better way to overcome the above two problems such as using the approximate nearest neighborhood (ANN) search method described in Hertzmann's paper on image analogies [15], which can be used to overcome problems in both Wei's fast

TSVQ searching algorithm [30] and in Ashikmin's fast searching algorithm [1].

7 Conclusion and Discussion

In the literature, the problem of texture mixtures is not well understood or solved, and only a few published methods are related to it. In this paper, we address the problem of texture mixtures with constraints. We present a user constrained multiscale MRF model for texture mixtures based on stochastic relaxation, which is an extension to the original multiscale MRF model [25]. In the model, a texture image is represented as a Markov random field defined on a multiscale lattice system. The multiscale representation of an image is constructed based on local decimation to make sure that the texture information can transfer from higher levels to lower levels without loss of information.

Given two or more input textures, a texture mixture is synthesized by statistically sampling the LCPDFs calculated from the input textures with user specified constraint, which is expressed as a look-up table with the same size as the output texture. Each entry of the user constrained table is a vector of weights in the range of $[0,1]$, each of which is used to calculate a weighted sum of the LCPDFs for a given target pixel. Then the SR algorithm is used to generate a value for the pixel by sampling the weighted sum of LCPDFs.

A straightforward implementation of our algorithm is slow. A faster version is proposed by incorporating the idea of Ashikmin's fast neighborhood searching algorithm [1], and the running time is decreased by a factor of 500 - 1000. The experimental results show that our algorithm performs well in the quality of results and outperforms Wei's algorithm [31] in the quality of the synthesized textures.

We have also demonstrated that our user constrained MRF texture model can be used in the application to texture replacement. It is, however, noted that the current implementation of our algorithm has some limitations as discussed in Section 6.3, which can be addressed by the ANN search method described in Hertzmann's paper [15].

Acknowledgments

The authors would like to acknowledge the generous funding from: NSERC, the Department of Computing Science, the University of Alberta, and the ARC Scholarship. As well, they also would like to thank members of the Computer Graphics Group for discussions. Finally, the authors would like to thank the anonymous reviewers for their comments.

References

1. Ashikhmin, M., *Synthesizing Natural Textures.* ACM Symposium on Interactive 3D Graphics, 2001: p. 217-226.
2. Bar-Joseph, Z., et al., *Texture mixing and texture movie synthesis using statistical learning.* IEEE TVCG, 2001. **7**(2): p. 120-135.
3. Bernardini, F., I. Martin, and H. Rushmeier, *High Quality Texture Reconstruction from Multiple Scans.* IEEE TVCG, 2001. **7**(4): p. 318-332.
4. Bonet, J.S.D., *Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images.* ACM SIGGRAPH, 1997: p. 361-368.
5. Brooks, S. and N. Dodgson, *Self-Similarity Based Texture Editing.* ACM SIGGRAPH, 2002: p. 653-656.
6. Cross, G.C. and A.K. Jain, *Markov Random Field Texture Models.* IEEE PAMI, 1983. **5**(2): p. 25-39.
7. Drori, I., D. Cohen-Or, and H. Yeshurun, *Fragment-Based Image Completion.* ACM SIGGRAPH, 2003. **22**(3): p. 303-312.
8. Ebert, D.S., et al., *Texturing and Modeling: A Procedural Approach. Second Edition*. 1998, Academic Press.
9. Efros, A. and T. Leung, *Texture Synthesis by Non-Parametric Sampling.* IEEE ICCV, 1999: p. 1033-1038.
10. Efros, A.A. and W.T. Freeman, *Image Quilting for Texture Synthesis and Transfer.* ACM SIGGRAPH, 2001: p. 341-346.
11. Fung, J. and O. Veryovka, *Pen-and-ink textures for real-time rendering.* Graphics Interface, 2003: p. 131-138.
12. Geman, S. and D. Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian Restoration of Images.* IEEE PAMI, 1984. **6**(6): p. 721-741.
13. Heckbert, P.S., *Fundamentals of Texture Mapping and Image Warping. Master's Thesis*, in *Dept. of Elec. Eng. and Compt. Sci.* 1989, Univ. of California: Berkeley.
14. Heeger, A.J. and J.R. Bergen, *Pyramid-Based Texture Analysis/Synthesis.* ACM SIGGRAPH, 1995: p. 229-238.
15. Hertzmann, A., et al., *Image Analogies.* ACM SIGGRAPH, 2001: p. 327-340.
16. Igehy, H. and L. Pereira, *Image Replacement Through Texture Synthesis.* International Conference on Image Processing, 1997. **3**: p. 186-189.
17. Julesz, B., *Visual Pattern Discrimination.* IEEE Transactions on Information Theory, 1962: p. 84-92.
18. Kim, J. and F. Pellacini, *Jigsaw Image Mosaics.* ACM SIGGRAPH, 2002: p. 657-664.

19. Kwatra, V., et al., *Graphcut Textures: Image and Video Synthesis using Graph Cuts.* ACM SIGGRAPH, 2003. **22**(3): p. 227-286.
20. Lefebvre, L. and P. Poulin, *Analysis and Synthesis of Structural Textures.* Graphics Interface, 2000: p. 77-86.
21. Lensch, H., W. Heidrich, and H. Seidel, *A Silhouette-Based Algorithm for Texture Registration and Stitching.* Graphical Models, 2001. **63**(4): p. 245-262.
22. Liang, L., et al., *Real-Time Texture Synthesis by Patch-Based Sampling.* ACM SIGGRAPH, 2001. **20**(3): p. 127-150.
23. Liu, Z., et al., *Pattern-Based Texture Metamorphosis.* Pacific Graphics, 2002: p. 184-191.
24. Lohmann, G., *Co-occurrence-Based Analysis and Synthesis of Textures.* ICPR, 1994. **1**: p. 449-453.
25. Paget, R. and I.D. Longstaff, *Texture Synthesis via a Noncausal Nonparametric Multiscale Markov Random Field.* IEEE Transactions on Image Processing, 1998. **7**(6): p. 925-931.
26. Paget, R., *NonParametric Markov Random Field Models for Natural Texture Images*, in *Dept. of Computer Science & Electrical Engineering*. 1999, The Univ. of Queensland, Australia: Queensland.
27. Portilla, J. and E.P. Simoncelli, *A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients.* Int'l Journal of Computer Vision, 2000. **40**(1): p. 49-71.
28. Soler, C., M. Cani, and A. Angelidis, *Hierarchical Pattern Mapping.* ACM SIGGRAPH, 2002. **21**(3): p. 673-680.
29. Veryovka, O., *Animation with Threshold Textures.* Graphics Interface, 2002: p. 9-16.
30. Wei, L. and M. Levoy, *Texture Synthesis Using Tree-Structured Vector Quantization.* ACM SIGGRAPH, 2000: p. 479-488.
31. Wei, L., *Texture Synthesis by Fixed Neighborhood Searching*, in *The Dept. of Elec. Eng.* 2001, Stanford Univ.: Stanford.
32. Ying, L., et al., *Texture and Shape Synthesis on Surfaces.* Eurographics Workshop on Rendering, 2001: p. 301-312.
33. Zelinka, S. and M. Garland, *Interactive Texture Synthesis on Surfaces using Jump Maps.* Eurographics Symposium on Rendering, 2003: p. 90-96.
34. Zhang, J., et al., *Synthesis of Progressively-Variant Textures on Arbitrary Surfaces.* ACM SIGGRAPH, 2003. **22**(3): p. 295-302.
35. Zhu, S.C., Y. Wu, and D. Mumford, *Filters, Random Fields and Maximum Entropy - Towards a Unified Theory for Texture Modeling.* Int'l Journal of Computer Vision, 1998. **27**(2): p. 1-20.