Texture Synthesis by a Neural Network Model

B. B. Chaudhuri and P. Kundu

Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Calcutta, India

In this paper we propose a neural network model to synthesise texture images. The model is based on a continuous Hopfield-like network where each pixel of the image is occupied by a neuron that is eight-connected to its neighbours. A state of the neuron denotes a certain grey level of the corresponding pixel. The firing of the neuron changes its state, and hence the grey level of the corresponding pixel. Different two-tone and grey-tone texture images can be synthesised by manipulating the connection weights and by varying the algorithm iteration number. For grey-tone texture synthesis, a Markov chain principle has been employed to decide on the multiple state transition of a neuron. The model can be employed for texture propagation with the advantage that it allows propagation without showing any blocky effect.

Keywords: Computer graphics; Image processing; Texture synthesis

1. Introduction

Computer-generated images can appear highly realistic if the surfaces are rendered with adequate textures. Texture synthesis can be useful in computer graphics and animation, design of textiles, wallpaper and laminate finishing, camouflaging techniques, texturing of missing or incomplete portions, etc. Texture synthesis can also be applied in scientific problems involving analysis by synthesis.

Use of formal grammar is an approach towards texture synthesis. Fu and Lu [1] used a stochastic tree grammar approach for the purpose. Another approach is the growth model introduced by Yokoyama and Haralick [2]. The model is based on three operations, namely seed distribution operation, skeleton growth operation and muscle growth operation. Another approach proposed by Ahuja [3] and Schacter *et al.* [4] is based on creating contiguous piecewise patterns by mosaic and bombing models. Some approaches to texture synthesis are based on a computer drafting approach. For example, Mezei *et al.* [5] describe a model for plotting patterns of natural objects such as rocks, wool, brush, bark and fur. In this method, a set of primitive shapes like lines, curves, quadrilaterals, ovals, etc. are chosen and distributed on a surface with a dense uniform random pattern. Some stochastic texture synthesis models are also available. For example, textures may be synthesised using Markov chains [6–9]. Another powerful class of texture synthesis is based on fractal geometry [10,11].

In this paper we propose a neural network model for synthesising two tone as well as grey tone texture images. This model is based on a continuous Hopfield-like network. For grey-tone texture generation, a Markov chain principle has been used to decide on the multiple state transitions corresponding to multiple grey levels.

The basic model is discussed in Sect. 2, while two-tone and grey tone texture synthesis approaches are described in Sect. 3. This system can be used in texture propagation, i.e. growing the texture field without having a blocky effect. The texture propagation aspect is described in Sect. 4. The synthesis results are presented and discussed in Sect. 5.

2. The Model

The proposed model consists of a continuous Hopfield-like network of m neurons, in which the action

potential between a neuron pair is simulated by the positive and negative signals circulating in the net. Each neuron accumulates signals as they arrive from its connecting neurons, and the potential of the neuron increases or decreases depending on whether the arriving signal is positive or negative. The potential of a neuron N_i is determined by the arriving signals weighted by the connection strengths between N_i and its connecting neurons. The node value of a neuron signifies its state. Thus, if X_i is the node value of the neuron N_i and w_{ij} is its connection strength with the neuron N_j whose node value is X_j, then the potential of the neuron N_i is given by

$$e_i = \sum_{j=i,\, j\neq i}^{n} I(X_i - X_j) w_{ij} \tag{1}$$

where $I(\cdot)$ is an indicator function defined by

$$I(x) = \begin{cases} -1 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

Note that $w_{ij} \approx 0$ if N_i and N_j are not directly connected. The quantity $I(X_i - X_j)w_{ij}$ represents the amount of signal coming from neuron N_j to neuron N_i.

The output of neuron N_i is determined from the total input signal e_i using the logistic function

$$P_i = \frac{1}{1 + \exp\{-(e_i - \alpha)/\beta\}} \tag{3}$$

where α, β are constants with $\beta > 0$.

Eq. (3) can be written as

$$P_i = \frac{1}{1 + \exp\{-E_i/\beta\}} \tag{4}$$

where $E_i = e_i - \alpha$.

A neuron can change its state by *firing*. Firing depends on the output P_i of the neuron. In the conventional Hopfield net, the neuron fires and changes its state for $E_i > 0$ if P_i exceeds a threshold $T \in [0,1]$. For our texture synthesis we have considered a different approach of firing in which the choice of threshold (which is user dependent) can be avoided. The approach and the reason of using it are discussed in Sect. 3.

Figure 1 shows the plot of P_i against E_i. It can be noted that P_i is a proper cumulative distribution function.

The parameter β determines the steepness of the curve; the smaller the value of β, the steeper is the curve. For $\beta = 0$, the model behaves exactly like the Hopfield model [13–15], in which a neuron fires and changes its state ($P = 1$) only when $E > 0$ and remains in its previous state ($P = 0$), otherwise. See Fig. 2 for an illustration.

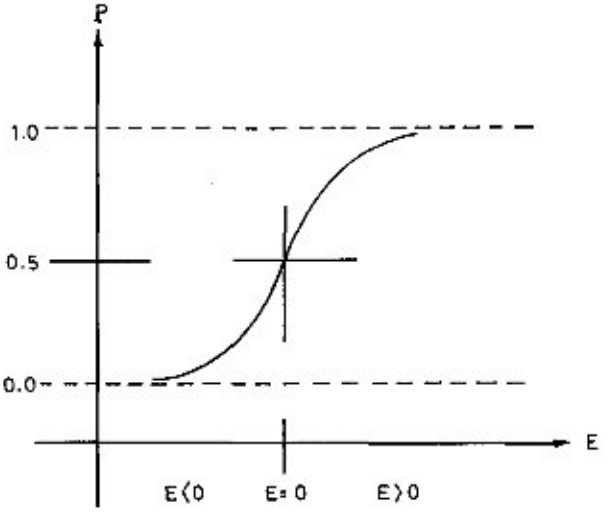

Fig. 1. Plot of the output function of Eq. (4).

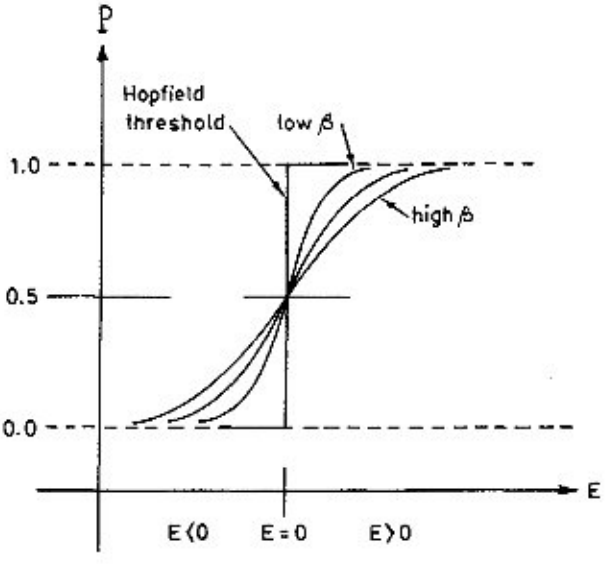

Fig. 2. Variation in the slope of the curve of Eq. (4) with β.

From Eq. (3) it is seen that the negative value of α encourages the neuron to fire, while the positive value of α discourages the neuron from firing. Thus, the parameter α indirectly controls the firing process of the neurons in the net. Also the value of the parameter β in Eq. (4) influences the firing rate of neurons. The output P of a neuron increases for $\beta < 1$ and increases for $\beta > 1$, and has no effect for $\beta = 1$.

3. Texture Synthesis

Let us now apply our model to synthesise artificial texture images in an $n \times n$ pixel space. We associate

a neuron $N_{i,j}$ to each pixel (i,j) in the image plane, and let there be $n \times n$ neurons in the net. The state $f_{i,j}$ of the neuron $N_{i,j}$ can be interpreted as the grey value of the pixel at (i,j). The topology of our proposed network for texture synthesis is shown in Fig. 3 in which each neuron is connected to at most eight nearest neurons. This network corresponds to the second order neighbourhood system. The topology for a higher order neighbourhood system may be considered in a similar way.

Let $w_{i+k,j+l}$, $k,l \in \{-1,0,1\}$ be the connection weights of the neuron $N_{i,j}$ with neurons $N_{i+k,j+l}$, $k,l \in \{-1,0,1\}$. We start with an image having uniformly distributed random grey values from $\{0,1,\ldots,G-1\}$, G being the total number of grey values. These grey values correspond to the initial states of the neurons in the net.

Then the potential of the neuron $N_{i,j}$ is

$$E_{i,j} = \sum_{k,l \in \{-1,0,1\}} w_{i+k,j+l} I(f_{i+k,j+l} - f_{i,j}) - \alpha;$$
$$\text{where } w_{i,j} = 0 \tag{5}$$

and by Eq. (4) the output of the neuron $N_{i,j}$ is

$$P_{i,j} = \frac{1}{1 + \exp\{-E_{i,j}/\beta\}} \tag{6}$$

The motivation of choosing such a neural model for texture synthesis is as follows. A texture image is, in general, characterised by the neighbourhood configuration of its pixels. The clustering of the pixels with identical grey values in a particular fashion gives rise to a particular texture. For example, a directional texture is obtained if there is a tendency of grey value clustering in a particular direction. Such directional clustering has been achieved by an appropriate choice of connection weights and the firing condition of the neuron. In our model, the total accumulation of signals to a neuron from its surroundings determines its change of state. Consider a neuron which corresponds to a black pixel surrounded by all white pixels. We may call the state of this neuron unstable, as its node value differs from the node values of all its neighbouring neurons. The neuron should fire and change to its stable state, which is white. From Eq. (5) it is seen that the potential E of this neuron is high and consequently, the output P (from Eq. (6)) is also high.

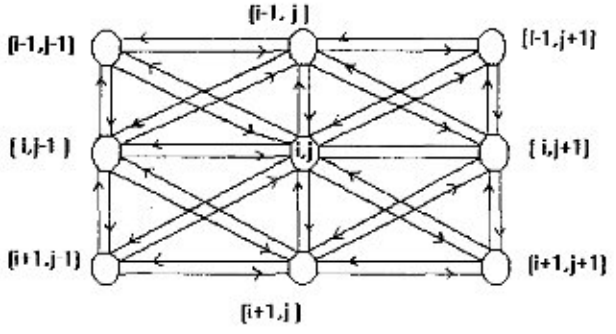

Fig. 3. Topology of the proposed neural net model.

It is possible to use a threshold T so that when $P \geq T$, the neuron fires and its state is changed. But instead of doing so, a threshold-free firing decision can be made as follows:

- Compute P.
- Assume that the state of the neuron is changed and compute the output. Let it be P'.
- If $P \geq P'$ then the neuron actually fires and the state is changed.

Note that firing is activated to achieve lower output and hence a more stable state of a neuron. If, for a specific configuration, the lowest output state is achieved, then the neuron will never fire again. In the example cited above, once the black neuron fires and changes to white, it will have the lowest output state and will never change its state again. Thus, the firing condition satisfies our clustering need.

The specific ways of synthesising two-tone and grey-tone textures are described below.

3.1. Two-Tone Texture Synthesis

In this case $G = 2$. The initial image is formed by randomly assigning a state from $\{0,1\}$ to each pixel so that the states are equally likely. The initial configuration of the net is thus formed. Let the neuron $N_{i,j}$ be in state $g, g \in \{0,1\}$. Given a set of connection weights of the neighbouring neurons, we find the potential $E_{i,j}$ and the corresponding output $P_{i,j}$ of the neuron $N_{i,j}$ using Eqs (5) and (6). Now assuming the state of neuron $N_{i,j}$ to be $1 - g$ and the states of all other connecting neurons to be the same as before, we find the corresponding potential $E'_{i,j}$ and the output $P'_{i,j}$ using the same equations. If $P_{i,j} \geq P'_{i,j}$, then the next state of neuron $N_{i,j}$ is chosen as $1 - g$. Otherwise, the state of the neuron is not changed.

The above procedure is applied iteratively to each neuron of the net for a specified number of times. It is seen that the total number of changes in states of the neurons decreases with an increasing number of iterations (one iteration is the full scan of $n \times n$ neurons). This effect is expected, since the firing of a neuron decreases its potential so that it will have less chance of firing in the next iteration. Because

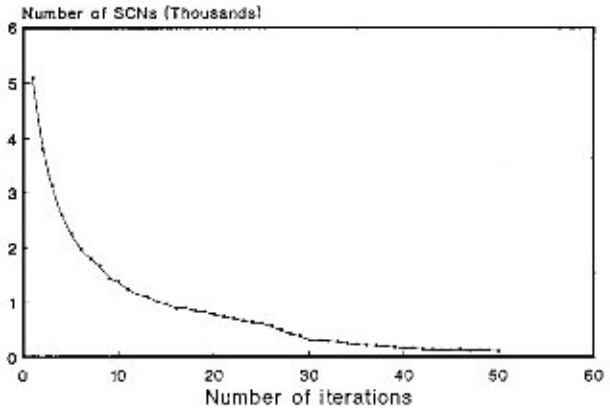

Fig. 4. Stability curve. (For four levels synthesised texture of Fig. 6(a).)

the net has a tendency to minimise the potential of its states, a stable state is attained after some iterations. The tendency of stability is illustrated in Fig. 4, in which a plot of the number of State Changing Nodes (SCN) against the number of iterations is given. From this graph it can be seen that the number of SCNs decreases with the increase in the number of iterations. After 10 iterations the number of SCNs is about 10% of the whole image size, and this number still decreases in the subsequent iterations.

We see that high values of horizontal weights will generate a texture image with grey value clustering in the horizontal direction, generating a (horizontal) directional texture. Examples of two-tone directional textures are shown in Fig. 5. Note that the node values (states) of the neurons are directly used as grey values of the corresponding pixels. Similarly, the other directional textures can be synthesised if high values are assigned to the corresponding directional weights and low values of other weights. If there is no bias to any weight, isotropic texture image results. (See Fig. 8(a), for example).

The clustering effect is also seen to vary with the number of iterations. An increase in the number

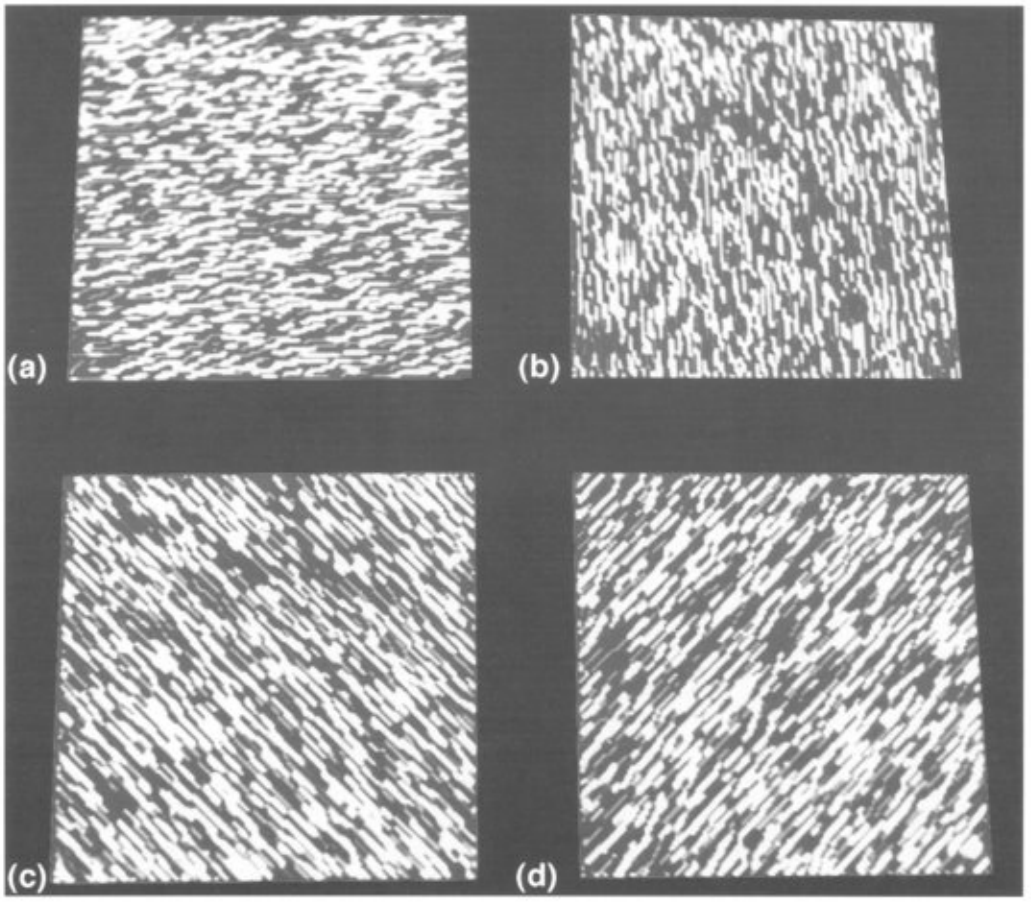

Fig. 5. Two-tone directional textures. (a) Horizontal texture ($w_0 = 2.2$, $w_1 = 2.0$, $w_2 = w_3 = w_4 = 0.0$); (b) vertical texture ($w_0 = 2.2$, $w_2 = 2.0$, $w_1 = w_3 = w_4 = 0.0$); (c) left diagonal texture ($w_0 = 2.2$, $w_3 = 2.0$, $w_1 = w_2 = w_4 = 0.0$); (d) right diagonal texture ($w_0 = 2.2$, $w_4 = 2.0$, $w_1 = w_2 = w_3 = 0.0$). For each image $\beta = 1.0$.

of iterations changes a micro-texture (less clustered) to a macro-texture (more clustered). This is expected as each iteration produces more stable neurons than those in the preceding iteration.

3.2. Grey-Tone Texture Synthesis

Suppose we want to construct a grey-tone image containing the grey levels $\{0,1,\ldots,G-1\}$. Then each neuron in the net can have one of several states. The state of each neuron is depicted by the grey value of the pixel to which it is associated. If the current state of a neuron $N_{i,j}$ is g_1, $g_1 \in \{0,1,\ldots G-1\}$, then given a set of connection weights of the neighbouring neurons, one can find its output $P_{i,j}$ using Eqs (5) and (6). If the neuron fires then it will change its state. From Eq. (6) we can only determine whether a neuron will fire or not. This equation does not indicate the state to which the neuron should travel to if it fires. This situation does not arise in the binary case described above, where a firing neuron has only one alternative state to go to. Hence, if a neuron can have one of several states and fires, the state transition is to be determined by the corresponding conditional transition probabilities, i.e. the probability of transition to a state g_2 given that it is in state g_1, where $g_1,g_2 \in \{0,1,\ldots G-1\}$. We use a Markov chain to determine such conditional probabilities for the transition of the neuron to different states.

3.2.1. Markov Chain. Given a set of events $S_0,S_1,\ldots,S_{m-1}$ and given a system for which these events follow one another with known probabilities $\pi(a,b)$ (the probability of state S_b following state S_a), the system may be represented fully by an $m \times m$ matrix $\{\pi(a,b)\}$ called a transition matrix of a finite Markov chain.

Let M be a transition matrix of a finite Markov chain (i.e. a chain having a finite number of states). The matrix M is regular if and only if for some integer L, M^L has no zero entries. A finite Markov chain having a regular transition matrix is called a regular Markov chain. Thus, a regular Markov chain is one in which any of its states can exist after some number of steps L, no matter what the starting state is. In other words, a regular Markov chain has no transient state, and has a single ergodic set with only one cycle. The following theorem [16] plays a significant role in the convergence to the equilibrium state from any initial state.

Theorem 1. If M is a J × J regular transition matrix, then there exists a matrix A satisfying

(i) $\lim_{L\to\infty} M^L = A$.

(ii) Each row of A is composed of the same probability vector of $\boldsymbol{\alpha} = (\alpha_1,\alpha_2,\ldots,\alpha_J)$, $\alpha_n \geq 0$, and $\sum_{n=1}^{J} \alpha_n = 1$, that is, $A = \boldsymbol{\xi}\boldsymbol{\alpha}$ where $\boldsymbol{\xi} = (1,1,\ldots 1)'$.

The matrix A is called a limiting matrix of M.

The above theorem suggests that if a regular Markov chain is used to determine the states of transition of firing neurons, then after a number of steps the net will be stable.

3.2.2. Synthesis Procedure. Now we use our model equipped with a regular Markov chain to generate a grey-tone texture image.

Let the starting image have uniformly distributed grey values from $\{0,1,\ldots,G-1\}$ at every pixel which constitutes the initial configuration of the net. Let neuron $N_{i,j}$ be in state g_1, $g_1 \in \{0,1,\ldots G-1\}$. Given a set of connection weights of the neighbouring neurons, we find the output $P_{i,j}$ of the neuron $N_{i,j}$ using Eqs (5) and (6).

Let $M = \{\pi(a,b)\}$ be a given regular transition matrix of order $G \times G$, where $\pi(a,b)$ is the conditional probability of transition from state a to state b and $a,b \in \{0,1,\ldots G-1\}$. We randomly select a state g_2 with probability $\pi(g_1,g_2)$. To do this, a random number r which is uniformly distributed over [0,1] is generated. If

$$\pi(g_1,g_2-1) < r < \pi(g_1,g_2-1) + \pi(g_1,g_2), \text{ where } \pi(g_1,-1) = 0 \tag{7}$$

then the selected state is g_2.

Assuming that the state of the neuron $N_{i,j}$ is g_2 and the states of the neighbouring neurons are the same as before, we find the corresponding potential $M'_{i,j}$ and the output $P'_{i,j}$ using Eqs (5) and (6).

If $P_{i,j} \geq P'_{i,j}$, then the next state of neuron $N_{i,j}$ is chosen to be g_2; otherwise, it remains in its current state g_1. In this way, the states of the neurons are changed to achieve a more stable configuration.

The above procedure is applied iteratively to each neuron of the net for a specified number of times. It is seen that the total number of changes in states of the neurons decreases with iterations and the net tends to attain a stable state after some iterations. This effect is expected as the chain is regular, and it attains the limiting equilibrium according to Theorem 1.

Thus, a variety of grey-tone texture images can be synthesised from any randomly (uniformly) dis-

tributed grey image with a given Markov chain (i.e. a regular transition probability matrix) and a set of connection weights. The node values (states) of the neurons are directly used as grey values of the corresponding pixels to generate the image. The effects of weights and the number of iterations on the generation of grey-tone texture images is seen similar to the case of two-tone texture images. For example, the clustering effect in a particular direction is seen to be more prominent if the weight in the corresponding direction is increased. See, for example, Fig. 6.

4. Texture Propagation

Suppose that a region is covered with synthesised texture and we want to cover some more regions around it with the same texture. The approach of doing so may be called *texture propagation*. It is possible to synthesise the texture separately in the regions and append them to the central region. But a blocky effect may be visible at the boundary of the regions. It is desirable that the propagation is smooth and no 'blocky' effect is visible. We propose a texture propagation method as follows.

A synthesised texture is characterised by the given set of quantities $\{M,W,n,G\}$, where M is the state transition matrix, W is the weight vector, n is the number of iterations and G is the number of grey levels. Now, if a small portion of the texture image I together with the set $\{M,W,n,G\}$ are stored, then applying the generation algorithm, the same texture image of any size can be reproduced whenever necessary.

Let I be an $m \times m$ texture image synthesised by our proposed model using the given set $\{M,W,n,G\}$. Suppose we want to propagate this image in a particular direction, say, from left to right starting from the right mth column. Let the desired image be of size $m \times N$, $N > m$. We first fill up the $N-m$ columns on the right-hand side of the image I by uniformly distributed random grey values from $\{0,1,\ldots,G-1\}$. Then the generation algorithm

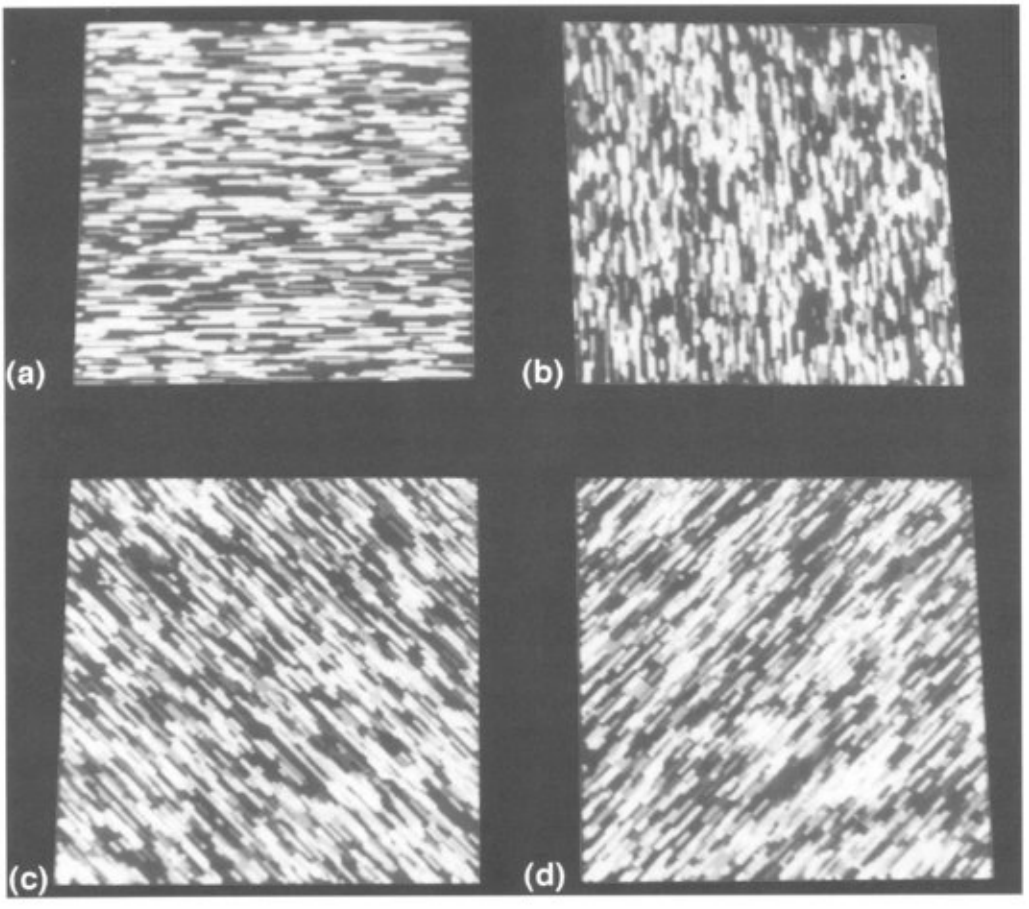

Fig. 6. Four levels directional textures. (a) Horizontal texture ($w_0 = 2.2$, $w_1 = 5.0$, $w_2 = w_3 = w_4 = 0.0$); (b) vertical texture ($w_0 = 2.2$, $w_2 = 5.0$, $w_1 = w_3 = w_4 = 0.0$); (c) left diagonal texture ($w_0 = 2.2$, $w_3 = 5.0$, $w_1 = w_2 = w_4 = 0.0$); (d) right diagonal texture ($w_0 = 2.2$, $w_4 = 5.0$, $w_1 = w_2 = w_3 = 0.0$). For each of image $\beta = 1.0$.

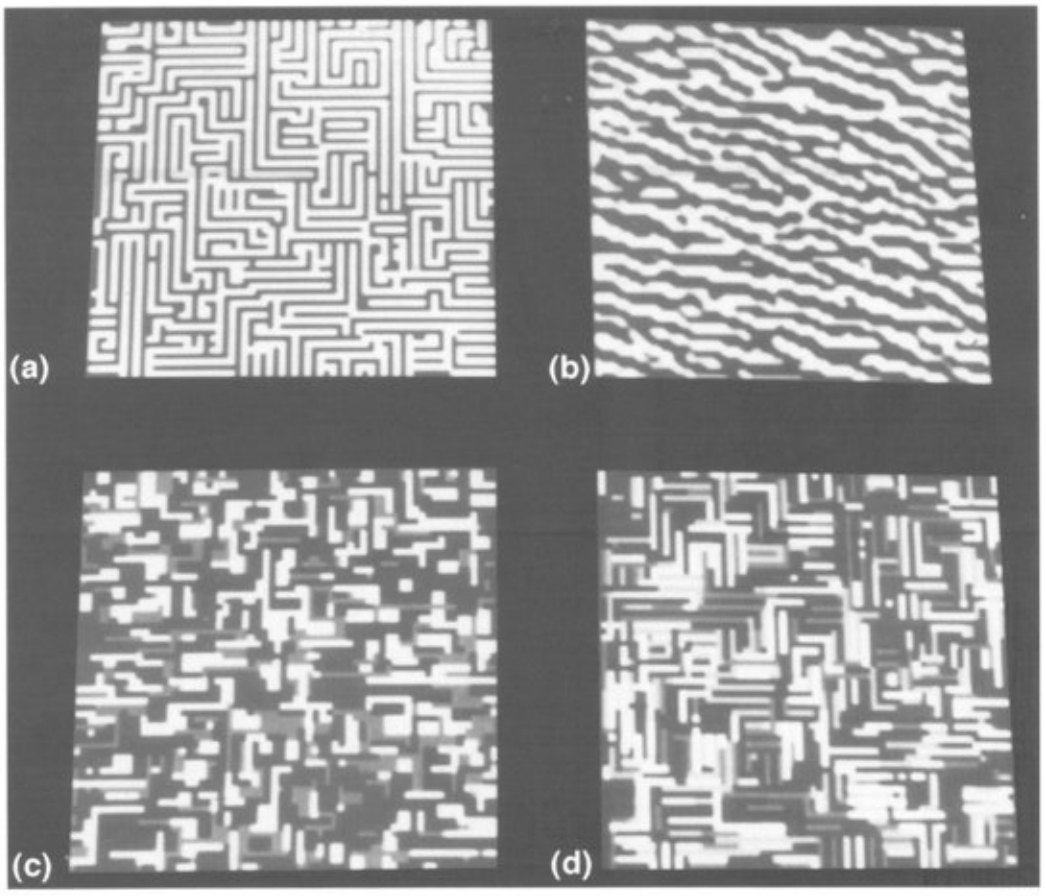

Fig. 7. Inhibited textures. (a) Two levels diagonal inhibited texture ($w_0 = 0.16$, $w_1 = 1.06$, $w_2 = 1.05$, $w_3 = -2.03$, $w_4 = -2.10$); (b) two levels vertical and one diagonal inhibited texture ($w_0 = 0.6$, $w_1 = 3.8$, $w_2 = -2.3$, $w_3 = 1.9$, $w_4 = -2.5$); (c) four levels diagonal inhibited texture ($w_0 = 2.2$, $w_1 = 1.8$, $w_2 = 1.7$, $w_3 = -0.9$, $w_4 = -0.9$); (d) eight levels diagonal inhibited texture ($w_0 = 4.6$, $w_1 = 2.6$, $w_2 = 1.05$, $w_3 = -2.1$, $w_4 = -2.1$). For each image $\beta = 2.0$.

described in Sect. 3 is applied on these columns, but starting from $(m-k)$-th column using the set $\{M,W,n,G\}$, where k is a small positive integer. The propagated image does not show any blocky effect along the mth column and a continuity is maintained along the mth row. This is because the algorithm starts from the $(m-k)$th column. In a similar manner, image I can be propagated in any direction.

5. Results and Discussion

We present here some examples of texture generated according to various settings of our model parameters (weights). These images are representative of a variety of texture families attributed as isotropic, anisotropic, attraction-repulsion, inhibition, etc. In each example, w_1, w_2, w_3, w_4 represent connection weights corresponding to horizontal, vertical and two diagonal directions respectively while w_0 represents the parameter α. Figures 5(a)–(d) and 6(a)–(d) show a series of 128×128 anisotropic textures in each of which thick and noisy lines are seen in a particular direction. The clustering in a particular direction corresponds to a relatively greater magnitude of the weight associated with that direction. If no bias is shown to a particular directional weight, a family of isotropic texture images may be generated.

If negative values are assigned to any two directional weights and positive values to other directional weights, inhibition results in the directions corresponding to the negative weights. For example, in Figs 7(a)–(d) the horizontal and vertical weights are positive while the two diagonal weights are negative. As a result, clustering in the horizontal and vertical directions are seen, but clustering in the diagonal directions are inhibited. Another inhibited texture is shown in Fig. 7(b).

If all the weights are positive, a tendency to cluster in all directions is seen, and consequently grainy isotropic textures result. Such textures are shown in Fig. 8. Figures 8(a)–(d) are clustered

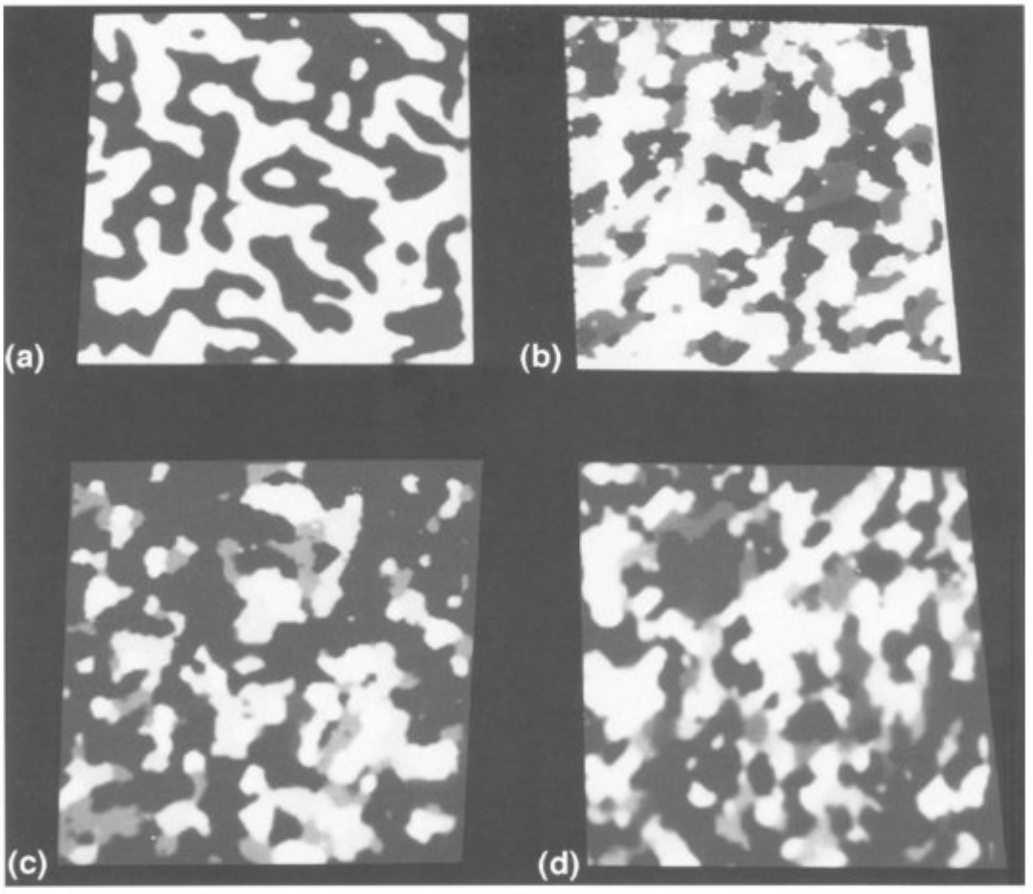

Fig. 8. Clustered textures, with (a) 2 levels ($w_0 = 1.0$, $w_1 = 5.0$, $w_2 = 4.0$, $w_3 = 5.0$, $w_4 = 4.0$); (b) with 4 levels ($w_0 = 3.0$, $w_1 = 1.0$, $w_2 = 0.2$, $w_3 = 0.1$, $w_4 = 0.8$); (c) with 8 levels ($w_0 = 3.0$, $w_1 = 5.0$, $w_2 = 4.2$, $w_3 = 4.4$, $w_4 = 3.8$); (d) with 12 levels ($w_0 = 3.0$, $w_1 = 4.8$, $w_2 = 5.0$, $w_3 = 4.4$, $w_4 = 4.5$). For each image $\beta = 2.0$.

images with 2, 4, 8 and 12 grey levels, respectively. As mentioned before, a regular Markov transition matrix is necessary to synthesise the grey level textures. The transition matrix for four grey level textures of Fig. 6 is

$$\begin{bmatrix} \frac{3}{8} & \frac{1}{8} & \frac{1}{8} & \frac{3}{8} \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & \frac{3}{8} \\ \frac{3}{8} & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} \end{bmatrix}$$

and the transition matrix for eight grey level textures of Figs 7(d) and 8(c) is

$$\begin{bmatrix} \frac{3}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{8} & \frac{1}{8} & \frac{1}{16} & \frac{3}{16} & \frac{3}{16} \\ \frac{1}{16} & \frac{1}{16} & \frac{1}{8} & \frac{1}{8} & \frac{1}{16} & \frac{3}{16} & \frac{3}{16} & \frac{3}{16} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{8} & \frac{1}{16} & \frac{3}{16} & \frac{3}{16} & \frac{3}{16} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{16} & \frac{3}{16} & \frac{3}{16} & \frac{3}{16} & \frac{1}{16} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{16} & \frac{3}{16} & \frac{3}{16} & \frac{3}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{8} \\ \frac{1}{16} & \frac{3}{16} & \frac{3}{16} & \frac{3}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{8} & \frac{1}{8} \\ \frac{3}{16} & \frac{3}{16} & \frac{3}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{8} & \frac{1}{8} & \frac{1}{16} \\ \frac{3}{16} & \frac{3}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{8} & \frac{1}{8} & \frac{1}{16} & \frac{3}{16} \end{bmatrix}$$

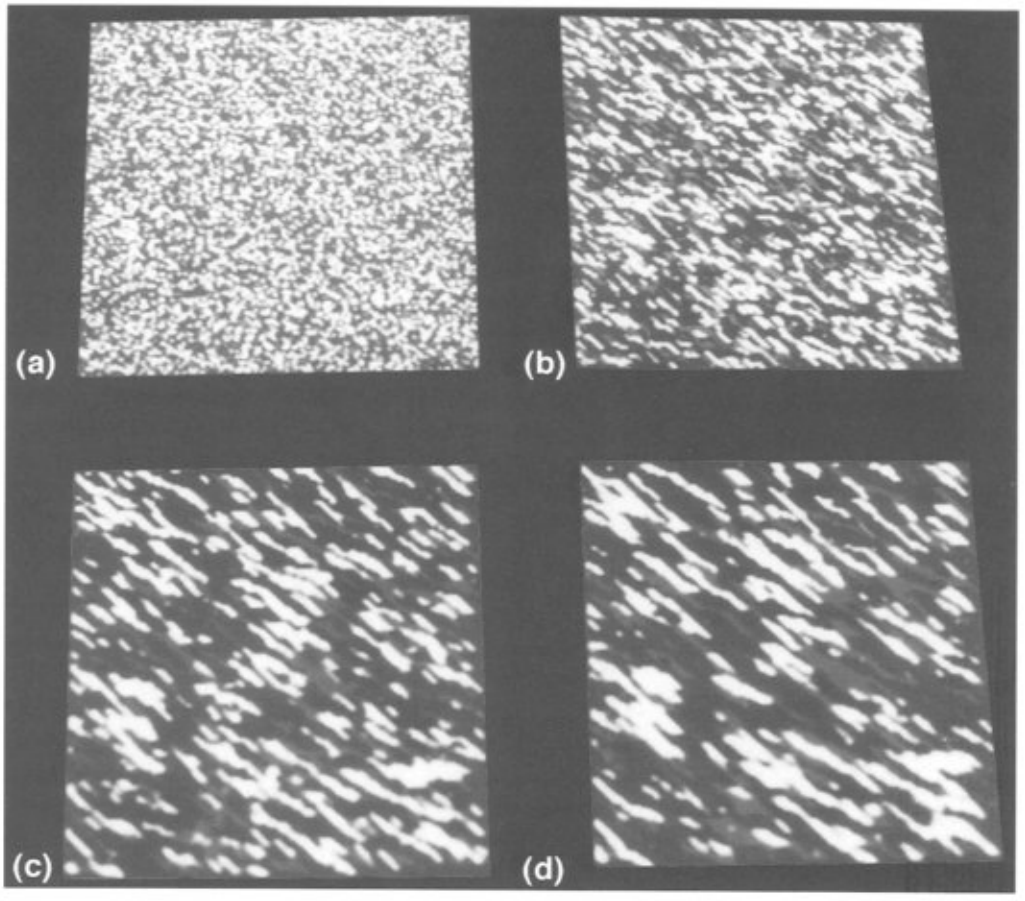

Fig. 9. Clustering effect with the number of iterations. (a) initial image with uniformly distributed four grey values; (b) after five iterations; (c) after 20 iterations; (d) after 40 iterations ($w_0 = 2.2$, $w_1 = 1.4$, $w_2 = 1.2$, $w_3 = 1.4$, $w_4 = 1.2$ and $\beta = 1.0$.)

If the number of iterations is increased, the clustering effect is increased. The variation of clustering with the number of iterations is shown in Fig. 9.

We present the texture propagation results in Fig. 10. Figure 10(a) is a 64×64 synthesised image with 4 grey levels. This image has been propagated in four directions – left, right, top, bottom – and the propagated image of size 256×256 is shown in Fig. 10(b). Note that no blocky effect is seen interior to the propagated image, and it is as continuous and homogeneous as the original one.

The proposed texture synthesis approach can be used in a wide variety of applications. One application concerns realistic animation in computer graphics where texture is rendered on the surface of 3D objects and the scene generated by graphical means. Texture wrapping is also useful for depth estimation in computer vision problems. Synthetic texture has an ideal structure and its distortion due to object depth imaged under ideal illumination is also ideal. A model of texture-based depth estimation can be properly formulated for such an ideal situation. Moreover, our algorithm can be useful in

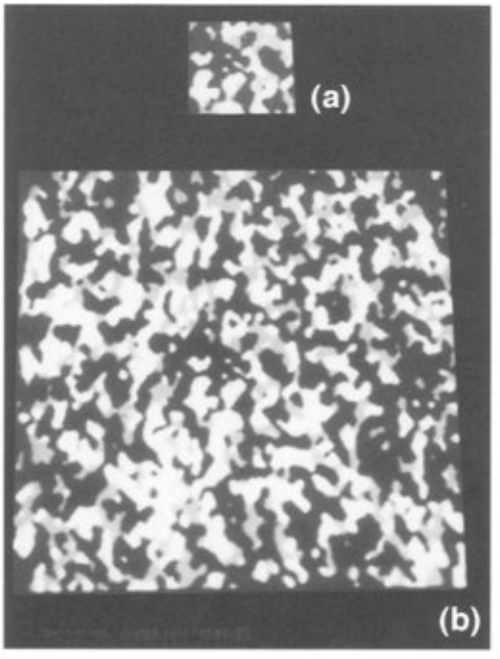

Fig. 10. Texture propagation. (a) Given image of size 64×64; (b) propagated image of size 256×256.

texture analysis by a synthesis approaches. Consider the segmentation of texture mosaic in a scene. To evaluate a segmentation algorithm for robustness, textures of different natures can be synthesised and presented to the algorithm.

The advantage of the current technique over some other existing ones is that one can control the texture grain size and texture directionality very conveniently by changing the parameters. Also, the neural net allows operations in parallel mode when it becomes computationally very fast. Another advantage is the texture propagation capability whereby one can dynamically increase the domain of texture synthesis while most existing techniques synthesise texture over a fixed domain only.

Acknowledgement. The authors wish to thank Dr C. A. Murthy for helpful discussion.

References

1. Fu KS, Lu SY. Stochastic tree grammer for texture synthesis and discrimination. Computer Graph Image Process 1979; 9: 234–245.
2. Yokoyama R, Haralick RM. Texture synthesis using a growth model. Computer Graph Image Process 1978; 8: 369–381.
3. Ahuja N. Mosaic models for image analysis and synthesis. PhD dissertation, Department of Computer Science, University of Maryland, 1979.
4. Schacter B, Rosenfeld A, Davis LS. Random mosaic models for textures. IEEE Trans System, Man and Cyber 1978; 8: 694–702.
5. Mezei L, Puzin M, Conroy P. Simulation of patterns of nature by computer graphics. Inform Process 1974; 74: 52–56.
6. Yokoyama R, Haralick RM. Texture pattern image generation by regular Markov chain. Pattern Recog 1979; 11: 225–234.
7. Chellappa R, Chatterjee S, Bagdazin R. Texture synthesis and compression using Gaussian–Markov random field models. IEEE Trans System, Man Cybern 1985; 15: 298–303.
8. Cross GR, Jain AK. Markov random field texture models. IEEE Trans Pattern Anal and Machine Intell 1983; 5: 1983.
9. Onural L, Gürelli MI. Generation and parameter estimation of Markov random field textures by highly parallel networks. From Pixels to Features 2, ESPRIT BRA Workshop on Parallelism in Image Process, Bonas, France, August 1990.
10. Mandlebrot BH. Fractals – Form, Chance, Dimension. W. H. Freeman, San Francisco, CA, 1977.
11. Voss RF. Fourier synthesis of Gaussian fractals: 1/f noises, landscapes and flakes. Tutorials on state of the art image synthesis. SIGGRAPH 1983, Detroit, MI, Vol. 10, 1983.
12. Johnson NL, Kotz S. Continuous Univariate Distribution – 2. Wiley.
13. Hopfield JJ. Neural networks and physical systems with emergent collective properties. Proc Nat Acad Sci USA 1982; 79: 2554–2558.
14. Hopfield JJ. Neurons with graded response have collective computational properties like those of two-state neurons. Proc Nat Acad Sci USA, Biophysics 1984; 81: 3088–3092.
15. Amin S, Gell M. Investigation of the Hopfield model and its attractors. Neural Comput and Applic 1994; 2: 129–133.
16. Kemeny J, Snell J. Finite Markov Chain. Springer, New York, 1960.