1991

M. Haindl

Texture synthesis

Computer Science/Department of Interactive Systems Report CS-R9139 September

Texture Synthesis

Michal Haindl
CWI
P.O.Box 4079, 1009 AB Amsterdam, The Netherlands

To enhance realism in graphical systems it is necessary to cover generated surfaces with natural-like textures. A texture in this report is assumed to be a realization of a random field. Problems of parameter estimation and random field synthesis of a given random field model are studied. The most flexible models for natural-like textures synthesis seem to be simultaneous autoregressive and Gaussian Markov random field models.

1980 Mathematics subject classification: Primary: 65C20. Secondary: 68U10.

Key Words & Phrases: texture synthesis, texture analysis, random fields.

Supported by an ERCIM (European Research Consortium for Informatics and Mathematics) Fellowship. This work was done while the author was visiting at CWI.

Report CS-R9139
ISSN 0169-118X
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Contents

1. Introduction

In this report , we consider texture synthesis methods. Our intention is to survey methods, which are capable of reproducing natural-like textures for enhancing realism in graphical systems. The advantage of digitized textures is their potential for realism. Digitalised solid textures are far less convenient, since it involves the 2D digitalization of a large number of cross-sectional slices through some material. Synthetic textures are more flexible than digitized textures, in that synthetic textures can be designed to have certain desirable properties or meet certain constrains; for example can be made smoothly periodic, so that it can be used to fill an infinite texture space without visible discontinuities. While a digitized texture must be stored in a tabular form and evaluated by table lookup, a synthetic textures may be evaluated directly in procedural form. Our goal is to reproduce a given digitalized texture image so that both natural and synthetic texture will be indiscernible in terms of Julesz [31]. Therefore we limit ourselves to synthesis methods for which the analysis step is known, neglecting different ad hoc methods [45] or models with unknown parameter estimation. Chapter 4. deals with analysis of texture , again there we limit our attention to subset of analysis methods, which are useful for chapter 3. texture synthesis methods. We are neglecting a large area of analysis methods useful for texture segmentation like for example the co-occurence matrices, gray level difference method etc. Interesting reader can refer any standard image processing textbook. Chapter five contains a description of 31 programs or procedures used in experimentation, results of which are covered in chapter six. While there exists an abundance of literature covering the texture analysis , much lesser attention was paid to the texture synthesis problem. So far even does not exist a survey covering state of art in this field so our selection of methods is based on papers referenced in the chapter seven.

2. Terminology

There is no exact definition for texture, although some authors claim to give it. Texture is generaly a visual property of surface, representing the spatial information contained in object surfaces. Texture may represent [4] an information that permits the human eye to differentiate between image regions. Anotherattempt to define texture is given [18]: A visual texture is a sensory impression obtained through the eyes, and thus a visual perception. Texture is considered here to be the visual perception, by which two neighboring, possibly structured, parts of the visual field may be effortlessly (spontaneously) separated by means of observation with fixed eyes. Another definition [22] states: Texture is a structure which is made of a large ensemble of elements that resemble each other "very-much", with some kind of an "order" in their locations, so that there is no one element which attracts the viewer's eye in any special way. The human viewer gets an impression of uniformity when he looks at a "texture". Some authors [39] use for a textured area element the therm texel or texon. We will not multiply here the number of rather philosophical definitions of texture, instead we understand an image as a realization of random field and our effort is simply to produce a copy of a given image using modelling technique. We do not differentiate between random variables and its realizations in our notation, the precise meaning is clear from context.

We use further on following symbols:

$M \times N$ is an image size

K is the number of gray levels (classes)

ω_i true class

r, s, u double indices ($r = \{i, j\}$)

I the image lattice (set of all indices)

Y random vector (or its realisation) of image in some arrangement

Y_r random variable (pixel) at r location

$Y_{(r)}$ random vector Y with excluded Y_r

γ set of model parameters

Q energy function

L likelihood function

Z Gibbs normalizing constant

I_r or $I_{i,j}$ a neighborhood of pixel $r = \{i, j\}$

μ mean value

σ^2 variance

3. Texture synthesis methods

Several approaches to texture synthesis exist, each of them has its advantages and also limitations. Existing methods can be categorize using different criteria . Englert and Sakas [18] divide methods into "models focusing on the description of spatial organization and the visual appearance of a texture field" and "models arising from texture generation method", other possible division is to deterministic and stochastic [22] methods. Similarly Bennis and Gagalowicz [4] have also two major groups structural (macroskopic) methods [3] considers texture as a spatial arrangement of a set of basic patterns according to some placement rules, while microscopic methods do not assume well organized structure. This textures present a homogeneous visual aspect even if without discernible spatial arrangement. In the first group we will mention random mozaic models, fractals, syntactic models and periodic tessellated model. We will focus our attention on the second group and especially on the autoregressive and Markov random field models, because of our strong belief that this models are far more flexible to cover natural-like textures than other approaches surveyd in this report.

3.1. Random mozaic models

Texture region is first divided into cells, than every cell is independently assign class (gray level, color) ω_i according to the fixed set of probabilities $P_1, \ldots, P_K$ Let r, s are two points distance d apart, $P(d) = P(Y_r = \omega, Y_s = \omega)$ where ω is a common true class for points r, s, then [54]:

$$P(Y_r = \omega_j | Y_s = \omega_i) = P_j(1 - P(d)) \tag{1}$$

$$P(Y_r = \omega_i | Y_s = \omega_i) = P_i + (1 - P_i)P(d) \tag{2}$$

Because of rectangular character of the rotated checkerboard model generated texture, we cannot expect its fit very much with natural textures. This model can be useful for certain man-made textures. Similarly also the Poisson line model do not fit very well [54] with natural textures. Some other models [54] of this class like for example the Johnson-Mehl model, the occupancy model and the bombing model are not mentioned here, because they do not fulfill our criterium to have solved also the analytical step (parameter estimation).

3.1.1. The Poisson line model

Texture area is divided into convex cells by a set of straight lines with random positions and orientations. Points (α, r) are generated by a Poisson process of intensity τ/π).

$0 \leq \alpha < \pi \qquad -\infty < r < \infty$ Each line is defined :

$$r = x \cos \alpha + y \sin \alpha \tag{3}$$

$$P(d) = \exp\{-2\tau d/\pi\} \tag{4}$$

3.1.2. The rotated checkerboard model

The origin of an coordinate system as well as the orientation of axes is chosen with uniform probability (axes in the interval $[0, \pi]$). Using resulting coordinate grid, the plane is tessallated into square cells of side lenght b. Probability of the same class for two distant points [54]:

$$P(d) = 1 - 4d/\pi b + d^2/\pi b^2 \qquad for \qquad d \leq b$$

$$P(d) = 1 - 2/\pi \frac{4}{\pi \cos(b/d)} - d^2/\pi b^2 + 4/\pi(d^2/b^2 - 1)^{1/2} \quad for \quad b < d \leq b\sqrt{2} \tag{5}$$

$$P(d) = 0 \qquad for \qquad d > b\sqrt{2}$$

Rotated Hexagon Model is analogous [55] to the checkerboard model , except that hexagons are used in place of squares. Another possibility is the rotated triangular model, where equilateral triangular tessellation can be formed by connecting the growth centers of neighboring cells in hexagonal tessellation.

3.2. Periodic tessellated models

In these models, a texture is divided into fixed-size windows, each window represents a unit pattern. The texture field is constructed by regularly replicating this unit over the whole image array [53].

3.3. Syntactic models

In the structural approach a texture is considered to be defined by subpatterns which occur repeatedly according to a set of well-defined placement rules within the overall pattern. A texture pattern is composed of fixed-sized windows as subpatterns. Each window is represented by a tree of which each mode corresponds to a pixel in the window. A pattern primitive is the label representing the gray level of each pixel. A tree grammar is then used to characterize the windowed texture pattern. Some examples of this approach can be found in [23],[24].

3.4. Fractal models

Fractals are sets, whose Hausdorff-Besicovitch dimension, which in general is a real number, is strictly larger than their topological dimension [46],[2]. Important properties of fractals is similarity of each segment to all others and statistical invariance over wide transformation of scale. The primary control that fractal techniques provide over the resulting object is by the value of a single parameter determining the fractal dimension. The only examples of fractal objects being used to model natural phenomena are based on fractional Brownian function. A random function Y_r is a fractional Brownian function (surface for 2D) if for all r, d

$$P(\frac{Y_{r+d} - Y_r}{||d||^H} < t) = F(t) \tag{6}$$

where $F(t)$ is a cumulative distribution function. If Y_r is scalar , then the fractal dimension of Y_r is

$$D = 2 - H \tag{7}$$

If $H = 0.5$ and $F(t)$ is the standartised Gaussian distribution, then Y_r is the classical Brownian function. A surface can be accurately approximated by a single fractal function, if the fractal dimension is stable over a wide range of scales. The texture is defined to be fractal [47] if

$$E\{|Y_{r+d} - Y_r|\} \parallel d \parallel^{-H} = const. \tag{8}$$

is valid. Several useful properties of the fractal Brownian function were proven in [47]. The fractal dimension of a fractal Brownian function is invariant over transformation of scale. A 3D surface with a spatially isotropic fractal Brownian shape produces an image whose intensity surface is fractal Brownian and whose fractal dimension is identical to that of components of the surface normal, given a Lambertian surface reflectance function and constant illumination and albedo. The oposite proposition is valid also. The fractal textures can be synthesize using one of following methods [17].

3.4.1. Fourier transformation method

A random Gaussian field is spatially filtered to generate well defined second-order connections between pixels. The phase of the Fourier transformed random field is unaltered , but the modulus is forced to assume a form f^{-P}. The parameter P can be altered to cause the inverse Fourier transformation of this modified field to have any particular fractal dimension between 2 and 3. Experiments on fractal textures generated using this method with identical first order statistics show [17], that the minimal resolvable difference in fractal dimension is 0.06.

3.4.2. Cylindrical integration method

A path within a random Gaussian field of zero mean painted on a cylinder is integrated over to produce a Brownian variation in image brightness. The path in cylindrical coordinates , is defined as $x\cos(\theta) + y\sin(\theta), \theta$, where $r = (x, y)$ is the rectangular coordinate of the cell in the Brownian field, and θ is the variable of integration. Although this method is able to produce true fractals, method is unable to generate a range of different dimension fractals.

3.4.3. Midpoint displacement method

The brigtness at the center of a square is interpolated from the four corner brigtness values. This value is perturbed by an amount related to the size of the square. The four subsquares are treated similarly and so on down to the resolution of the image. The result was shown not to be truly fractal, however the method is quick and efficient to implemented.

3.5. Second order statistical model

This texture model generates random field, which has similar second-order statistics with natural texture, we are trying to simulate. It is assumed, that although Julesz [35] conjecture, that second-order statistics are sufficient in terms of human visual texture discrimination, were proved not to be generaly valid, such an approximation is still fairly accurate.

3.5.1. Algebraic reconstruction technique

We assume that a reconstructed random field is homogeneous (translation invariant). The problem is stated as to find joint density function $p(Y_1, \dots, Y_{MN})$ from known marginals $p(Y_s, Y_r)$. This problem corresponds to linear programming problem with K^{MN} linear equations and inequalities. The use of standard technique is limited with large dimensionality of problem. The ART method starts [21] with generating all simulated pixels from uniform distribution so

$$p^{(0)}(Y_1, \dots, Y_{MN}) = \frac{1}{K^{MN}} \tag{9}$$

In each iteration the sum of the differences between the actual and the reconstructed marginal is computed and evenly divided amongst the K^{MN-2} reconstruction elements.

$$p^{(n+1)}(Y_1, \dots, Y_{MN}) = \max\{0, p^{(n)}(Y_1, \dots, Y_{MN}) + tc^{(n)}(Y)\} \qquad \forall\{Y\} \tag{10}$$

$$c^{(n)}(Y) = \frac{1}{K^{MN-2}\binom{MN}{2}} \sum_{r=1}^{MN-1} \sum_{s=r+1}^{MN} (p(Y_r, Y_s) - p^{(n)}(Y_r, Y_s)) \tag{11}$$

The convergence of ART algorithm depends on a choice of constant t.

3.5.2. Second order spatial averages

The principle of method is to generate a homogeneous random field with the second order spatial averages

$$\tilde{p}_d(\omega_i, \omega_j) = \frac{1}{n} \sum_{r=1}^{n} \delta(Y_r - \omega_i)\delta(Y_{r+d} - \omega_j) \tag{12}$$

equal with a given texture field. The first step of the method is synthesizing a texture field, which is a realization of a homogeneous white noise and whose histogram is equal to the desired ($\tilde{p}^*$) histogram [25].

$$\tilde{p}(\omega_i) = \tilde{p}^*(\omega_i) \qquad \forall i = 1, \dots, K$$

In the second step, the texture field is modified point by point to minimize the mean square error between the desired second order spatial averages and current second order spatial averages. It was experimentally verified [25], that the selection of modified points should be random, otherwise the resulting texture is not homogeneous.

3.5.3. Autocovariance and histogram model

This second order statistical model is defined by set of histogram and autocovariance parameters:

$$B(d) = \frac{1}{MN\sigma^2} \sum_{r=1}^{MN} (Y_r - \mu)(Y_{r+d} - \mu) \tag{13}$$

The first synthesis step is equal to the first step of 3.5.2. The second one is also random location pixel modification so to minimize the mean square error between template feature vector and the feature vector computed on the synthesized texture field, but the feature vector is now concatenated from texture histogram and autocovariance function [3].

3.6. Autoregressive model

Autoregressive or simultaneous autoregressive (SAR) models from the class of simultaneous models have several favourite properties, which cause their wide application in image analysis. Simultaneous models are basically generalizations of 1D time series model into 2D. Such a generalization is not quite straightforward because unlike in 1D time series, where the existence of a preferred direction (time) is assumed, no such preferred ordering exists on the discrete lattice. Other groups from the class of simultaneous models are simultaneous moving average (SMA) and simultaneous autoregressive and moving average (SARMA) models. This other models have complicated parameter estimation. For example estimation of ARMA model parameters is a nonlinear problem even in 1D, and some approximations have to be used. The advantage of SAR models is their close relation with Markov random field models (MRF). For every SAR model there exist a unique conditional MRF with equivalent spectral density function, but the converse is not always true (except for Gaussian case [5]). The conditional MRF model is generaly characterized by more parameters than the equivalent SAR model (if exists). SMA and SARMA models are not subsets of MRF models.

SAR model can be written:

$$Y_r = \sum_{\forall s \in I_r} a_s Y_{r-s} + e_r \tag{14}$$

where e_r is a white Gaussian noise with zero mean has a constant variance σ^2 and is uncorrelated with data from I_r. If I_r is symmetric neighborhood, we assume that symmetrically opposite neighbors parameters are equal. Otherwise, the parameters may not be identifiable [5].

We can rewrite SAR model in the matrix form:

$$Y_r = \gamma X_r + e_r \tag{15}$$

where

$$\gamma = [a_1, \ldots, a_s] \tag{16}$$

and X_r is corresponding vector of Y_{r-s}.

If we replace Y_r in (14) by $\acute{Y}_r$ (defined (41)), the joint probability density of process (14) can be written

$$p(Y) = (2\pi\sigma^2)^{-MN/2} |\Psi| \exp\{-1/2\sigma^{-2}(Y-U)^T \Psi^T \Psi (Y-U)\} \tag{17}$$

where Ψ is defined (39) and is for a torus lattice I a block circulant matrix , each block being again a circulant matrix, U is defined in (38).

We can syntetize image described by SAR from equation

$$Y = A^{-1}(E - B) \tag{18}$$

where the noise vector (19)

$$E = [e_1, \ldots, e_{MN}]^T \tag{19}$$

has arrangement corresponding with Y. B is vector with border conditions and A is $MN \times MN$ block Toeplitz type matrix of a_s coefficients with unit diagonal. If we will assume zero boundary value ($B = 0$), symmetric coefficients ($a_{r-s} = a_{r+s}$) and regularity of A, then A is regular, symmetric and positive definite. Under this condition the solution of (18) can be obtained by iterative the conjugate gradient method [51],[13] (Fletcher-Reeves version):

a) initiate working space $\{v_r\}$ and $\{w_r\}$ from white noise generator

$$v_r^{(0)} = w_r^{(0)} = e_r$$

b) iterate steps c)-g) from $n = 0$ until the root mean square of $v_r^{(n)}$ becomes smaller than a few percent of ϵ_r^n

c) $\alpha^{(n)} = \frac{\sum_r v_r^{(n)} w_r^{(n)}}{\sum_r w_r^{(n)} (w_r^{(n)} - \sum_{\forall s \in I_r} a_s w_s^{(n)})}$

d) $Y_r^{(n+1)} = Y_r^{(n)} + \alpha^{(n)} w_r^{(n)}$

e) $v_r^{(n+1)} = e_r - (Y_r^{(n+1)} - \sum_{\forall s \in I_r} a_s Y_{r-s}^{(n+1)})$

f) $\beta^{(n)} = \frac{\sum_r (v_r^{(n+1)})^2}{\sum_r (v_r^{(n)})^2}$

g) $w_r^{(n+1)} = v_r^{(n+1)} + \beta^{(n)} w_r^{(n)}$

Where $Y^(0)$ is an arbitrary starting vector. For functions which cannot be well approximated by a quadratic form is preferable to use the Polak-Ribiere version of method :

f) $\beta^{(n)} = \frac{\sum_r (v_r^{(n+1)})^2 - \sum_r v_r^{(n+1)} v_r^{(n)}}{\sum_r (v_r^{(n)})^2}$

3.6.1. Woods iterative synthesis

Woods has proven [58], that random field which can be described (14), where $\{e_r\}$ are independent random variables with bounded absolute moments (possibly non-Guassian) can be generated using iterative procedure (20),(21)

$$Y_r^{(n+1)} = \sum_{\forall s \in I_r} a_s Y_{r-s}^{(n)} + e_r \tag{20}$$

$$Y_r^{(n+1)} = Y_r \tag{21}$$

and stability assumption

$$\sum_{\forall s \in I_r} |a_s| < 1$$

.

Where equation (21) is used if r is from boundary area. Equation (20) for non boundary area is initialized (22).

$$Y_r^{(0)} = e_r \tag{22}$$

This iterative procedure converges to the mean absolute solution of (14), with boundary condition

$$Y_r = Y_r^{boundary}$$

.

3.6.2. Autoregressive model with conditional expectations

Let us assume now to have chosen a partion of a neighborhood I_r, we will denote I_r^i i-th such a subset of I_r and l the number of these subsets. Now the autoregressive model with conditional expectations can be put in form [20]:

$$Y_r = \sum_{i=1}^{l} a_i E\{Y_r | Y_s \forall s \in I_r^i\} + e_r \tag{23}$$

or in the matrix form (15),(16) , where X_r is now

$$X_r = [E\{Y_r|Y_s \forall s \in I_r^1\}, \ldots, E\{Y_r|Y_s \forall s \in I_r^l\}]^T \tag{24}$$

Main disadvantage of synthesis model (23) is necessity to identify and store a large number of parameters. If we denote the cardinality of I_r^i as q_i , then the number of model parameters is $\sum_{i=1}^{l} K^{q_i} + l$.

3.6.3. Optimal model selection

The selection of an appropriate AR model is important to receive good results in modelling a given random field. If the contextual neighborhood is too small we can not capture all details of RF. On the contrary larger than necessary contextual neighborhood can introduce problems with numerical accuracy and can be too time consuming. Apart from visual comparison , which is subjective , we can choose the appropriate model using pairwise hypothesis testing [5],[12], Akaike's information criterion (AIC) [36] or the Bayesian approach ($\max_i P(model_i|Y)$). The main disadvantage of the hypothesis testing approach is that the resulting decision rule is not transitive and also not consistent . The AIC method gives transitive decision rules but not consistent one [37]. The approximate Bayesian decision rule for AR model was given in [37]: Choose the AR model k if

$$k = \arg\min_j \{D_j\} \tag{25}$$

where

$$D_j = -\sum_{r \in I} \ln(1 - \gamma_k C_r + \gamma_k Q_r Q_r^T) + MN \ln \sigma_k^2 + m_k \ln(MN) \tag{26}$$

and $\gamma_k, \sigma_k^2, C_r, S_r$ are defined in (87),(88),(95),(96) respectively, m_k is number of neighbors in k-th model and

$$Q_r = S_r S_r^T + C_r C_r^T \tag{27}$$

3.7. Markov random field models

Markov random field (MRF) is a family of random variables with a joint probability density on the set of all possible realizations Y of the lattice I, subject to following conditions:

$$p(Y) > 0 \qquad \forall Y \tag{28}$$

(positivity condition)

$$p(Y_r | Y_s \forall s \in I) = p(Y_r | Y_s \forall s \in I_r) \tag{29}$$

(Markovianity in a strict sense [38], or local Markov property)

We limit ourselves here only to homogeneous and symmetric neighborhood systems I_r. A hierarchy of MRF models can be defined, using such a neighborhood systems, as follows from figure, where numbers on the place of contextual neighbors indicate the order of model (up to fifth) relative to x [5].

$$\begin{array}{ccccc}
5 & 4 & 3 & 4 & 5 \\
4 & 2 & 1 & 2 & 4 \\
3 & 1 & x & 1 & 3 \\
4 & 2 & 1 & 2 & 4 \\
5 & 4 & 3 & 4 & 5
\end{array}$$

The neighborhood systems I_r of the n-th order MRF model then contains all neighbors of x with position numbers $1, 2, \ldots, n$. We define a clique to be a set of points that consists either of a single point, or has the property that each point in the set is a neighbor of all the others. For example for the first-order MRF there are associated cliques $\{(i,j)\}, \{(i,j-1),(i,j)\}$, and $\{(i-1,j),(i,j)\}$,. A corresponding sufficient statistic for MRF require $2(K-1)^2 + K - 1$ independent parameters (isotropic MRF $(K-1)^2 + K - 1$).

The Hammersley-Clifford theorem [5] states that Y is MRF with strictly positive distribution and neighborhood system I_r, if and only if the distribution of Y can be written as a Gibbs distribution with cliques induced by the neighborhood system I_r. The function $G_{r,s,\ldots,u}$ in (22) may be non-null if and only if the sites $r, s, \ldots, u$ form a clique . Subject to this restriction the G - functions may be chosen arbitrarily. For a given neighborhood system a Gibbs density function can be expressed in the form

$$p(Y) = \exp\{-Q(Y)\}/Z \tag{30}$$

where the normalization constant is

$$Z = \sum_Y \exp\{-Q(Y)\} \tag{31}$$

Global energy function (or Gibbs energy function) is according to the Hammersley-Clifford theorem:

$$Q(Y) = \sum_{\forall cliques \in I_r} Q_j =$$

$$Q(0) + \sum_{r} G_r Y_r + \sum_{r,s} G_{r,s} Y_r Y_s + \ldots + G_{r,s,\ldots,u} Y_r, \ldots, Y_u \tag{32}$$

Local conditional densities , for example for the first-order model, can be determined from (32):

$$p(Y_r | Y_{(r)}) = \exp\{-Y_r G_r - \sum_{s \in I_r} G_{rs} Y_r Y_s\} / \acute{Z} \tag{33}$$

$$\acute{Z} = \sum_{r} \exp\{-Y_r G_r - \sum_{s \in I_r} G_{rs} Y_r Y_s\} \tag{34}$$

Boundary pixels in a finite image have less neighbors than the interior pixels. This missing data problem is solved using two approximations. Free boundary condition assumes the interaction potential between boundary pixel and its missing neighbors to be zero. The second approximation assumes a toroidal lattice that results in a random field which is wrapped around in a torus structure [11]. Unlike 1D time series, where we can differentiate between past and future data and accordingly between causal and non-causal models, respectively, no preferred direction is assumed in 2D data. Therefore 2D causal models seem somehow artificial, but their advantage lies in the possibility of recursive parameter estimation for some models.

3.7.1. Gaussian Markov random field model

If the local conditional density of MRF model (35) is Gausssian , we obtain GMRF model.

$$p(Y_r | Y_s \forall s \in I_r) = (2\pi\sigma^2)^{-1/2} \exp\{-1/2\sigma^{-2}(Y_r - \acute{\mu}_r)^2\} \tag{35}$$

where the mean value is

$$E\{Y_r | Y_s \forall s \in I\} = \acute{\mu}_r = \mu_r + \sum_{s \in I_r} a_s (Y_{r-s} - \mu_{r-s}) \tag{36}$$

and the joint probability density function is

$$p(Y) = (2\pi\sigma^2)^{-MN/2} |\Psi|^{1/2} \exp\{-[(Y - U)^T \Psi (Y - U)] / (2\sigma^2)\} \tag{37}$$

where $(MN \times 1)$ mean vector consists of single location means

$$E\{Y\} = U = [\mu_{1,1}, \ldots, \mu_{M,N}]^T \tag{38}$$

and the covariance matrix has unit diagonal elements [5] and off diagonal elements are coefficients a_r. Ψ is symmetric, but also is required to be positive definite. Coefficients of symmetric sites have to be equal i.e. $a_{r-s} = a_{r+s}$.

$$E\{(Y-U)^T(Y-U)\} = \sigma^{-2}\Psi^{-1} \tag{39}$$

.

The argument in the exponent of $p(Y)$ can be put in form:

$$Q(Y) = \sum \acute{Y}_{i,j}^2 + a_{1,0}\sum \acute{Y}_{i,j}\acute{Y}_{i+1,j} + a_{0,1}\sum \acute{Y}_{i,j}\acute{Y}_{i,j+1} + \dots \tag{40}$$

where

$$\acute{Y}_{i,j} = Y_{i,j} - \mu \tag{41}$$

.

The GMRF is parametrized by $\gamma = \{\mu, \sigma^2, a_{1,0}, a_{0,1}, \dots\}$. GMRF can be expressed as a stacionary noncausal 2D autoregressive process described by difference equation

$$Y_{i,j} = \mu_{i,j} + \sum_{k,l \in I_{i,j}} a_{k,l}(Y_{i-k,j-l} - \mu_{i-k,j-l}) + e_{i,j} \tag{42}$$

where $a_{k,l} = a_{-k,-l}$, $\mu_{i,j}$ is the mean value of $Y_{i,j}$, $e_{i,j}$ is a stacionary Gaussian noise with zero mean and autocorrelation given

$$R_e(k,l) = \begin{cases} \sigma^2 & \text{if (k,l)=(0,0)} \\ -\sigma^2 a_{k,l} & \text{if (k,l)} \in I_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

The power spectral density associated with $\acute{Y}_{i,j}$ is [19]:

$$S_{\acute{Y}}(i,j) = \sigma^2 / [1 - 2 \sum_{(k,l)\in I^*_{i,j}} a_{k,l}\cos\{2\pi(ki/M + lj/N)\}] \tag{44}$$

where $I^*_{i,j}$ is the nonsymmetric half-plane of $I_{i,j}$ and 2D Fourier transformation of $\acute{Y}_{i,j}$:

$$\mathcal{F}\{\acute{Y}_{i,j}\} = \sum_{\forall(k,l)\in I} \sum \acute{Y}_{k,l} \exp\{-(-1)^{1/2} 2\pi(ki/M + lj/N)\} \tag{45}$$

It can be shown for toroidal lattices [58],[11] that the discrete Fourier transformation $\mathcal{F}\{\acute{Y}\}$ is a white Gaussian field.

$$p(\mathcal{F}\{\acute{Y}\}) = \prod_{i,j} [2\pi MNS_{\acute{Y}}(i,j)]^{-1/2} \exp\{-\sum_{i,j} \parallel \mathcal{F}\{\acute{Y}\} \parallel^2 [2MNS_{\acute{Y}}(i,j)]^{-1}\} \tag{46}$$

GMRF can be generated from

$$Y = \mathcal{F}^{-1}\{\hat{Y}\} + U \tag{47}$$

where mean vector U is defined (38) and $\hat{Y}$ is generated from $\mathcal{N}(0, NMS_{\hat{Y}}(i,j))$.

Another possibility for generating GMRF on a toroidal matrix $M \times M$ is [10]:

$$Y = \frac{1}{M^2} \sum_{\forall r \in I} \frac{f_r x_r}{\sqrt{\epsilon_r}} \tag{48}$$

where

$$x_r = f_r^{*T} \eta \tag{49}$$

η is M^2 vector from $\mathcal{N}(0,1)$.

$$f_r = [1, \lambda_i, \lambda_i^2 t_j, \ldots, \lambda_i^{M-1} t_j]^T \quad for \quad r = (i,j) \tag{50}$$

$$t_j = [1, \lambda_j, \lambda_j^2, \ldots, \lambda_j^{M-1}]^T \tag{51}$$

$$\lambda_i = \exp\{\sqrt{-1}\frac{2\pi i}{M}\} \tag{52}$$

$$\epsilon_r = 1 - 2\gamma\phi_r \tag{53}$$

$$\phi_r = [\cos\frac{2\pi r^T s}{M}, \forall s \in I_r]^T \tag{54}$$

For GMRF models which have an equivalent AR model, we can use Woods iterative algorithm 3.6.1. GMRF models which do not have an equivalent AR model, can be generated by modified Woods algorithm, but nothing is known about its convergence properties, because the convergence theorem [58] is not valid any more. For the optimal GMRF model selection we can use modification of the Bayesian method 3.6.3. , where m_k is now a number of asymmetrical half neighborhood parameters and

$$D_j = -\sum_{r \in I} \ln(1 - 2\gamma_k C_r) + MN \ln \sigma_k^2 + m_k \ln(MN)$$

.

3.7.2. Mutually compactible Gibbs random field model

We shall assume contextual window consisting from three neighbors $I^*_{i,j} = \{(i,j),(i-1,j),(i-1,j-1),(i,j-1)\}$.We shall assume a constant mean global energy function

$$E\{Q\} = \sum_Y Q(Y)p(Y) = E_c \tag{55}$$

where the global energy function is defined

$$Q(Y) = \sum_{i=1}^{M}\sum_{j=1}^{N} Q_{ij}(Y_{i,j}, Y_{i-1,j}, Y_{i-1,j-1}, Y_{i,j-1}) \tag{56}$$

It is convenient to define the local transfer function (LTF) $\sigma_{i,j}$ as

$$\sigma_{i,j}(Y_l, Y_m, Y_n, Y_o) = \exp\{-Q_{i,j}(Y_l, Y_m, Y_n, Y_o)\} \tag{57}$$

Definition 1: A set of sites A is called the primary sublattice of lattice I if $A \subseteq I, \{(m,n) : 1 \leq m \leq M_A, n = 1\} \subseteq A, \{(m,n) : m = 1, 1 \leq n \leq N_A\} \subseteq A$ for some (M_A, N_A) with $1 \leq M_A \leq M, 1 \leq N_A \leq N$, and if $(m,n) \in A$, then $\{(i,k),(i-1,k),(i-1,k-1),(i,k-1)\} \subseteq A$.

Definition 2: The GRF defined over a rectangular lattice I, and whose joint probability measure is given by eq. (30),(56),(57) is called a mutually compactible Gibbs random field (MC-GRF) [30], if its restriction Y_A over any primary sublattice A of lattice I is also GRF with joint probability measure

$$p(Y_A) = 1/Z_A \prod \prod_{(i,j)\in A} \sigma_{i,j}(Y_{i,j}, Y_{i-1,j}, Y_{i-1,j-1}, Y_{i,j-1}) \tag{58}$$

where

$$Z_A = \sum_{Y_A} \prod \prod_{(i,j)\in A} \sigma_{i,j}(Y_{i,j}, Y_{i-1,j}, Y_{i-1,j-1}, Y_{i,j-1}) \tag{59}$$

Theorem 1: A necessary and sufficient condition for a GRF to be a MC-GRF is

$$\sum_x \sigma_{ij}(x,t,z,y) = k_{ij} \tag{60}$$

for $1 \leq i \leq M, 1 \leq j \leq N$ and for every triplet (t,z,y) , where k_{ij} is a constant independent of (t,z,y).

Theorem 2: A GRF whose LTF satisfies (60) for every (i,j), is statistically equivalent to a unilateral MRF. A RF is statistically equivalent to a MC-GRF if and only if it is statistically equivalent to a unilateral MRF.

Definition 3: The GRF Y is called a GRF with a homogeneous LTF if

$$\sigma_{i,1}(Y_{i,1}, Y_{i-1,1}) = \sigma_v(Y_{i,1}, Y_{i-1,1}), \qquad for \quad i = 2,3,\ldots,M \tag{61}$$

$$\sigma_{1,j}(Y_{1,j}, Y_{1,j-1}) = \sigma_h(Y_{1,j}, Y_{1,j-1}), \qquad for \quad i = 2,3,\ldots,N \tag{62}$$

and

$$\sigma_{i,j}(Y_{i,j}, Y_{i-1,j}, Y_{i-1,j-1}, Y_{i,j-1}) = \sigma(Y_{i,j}, Y_{i-1,j}, Y_{i-1,j-1}, Y_{i,j-1}),$$

$$for \quad i = 2, 3, \ldots, M, \qquad for \quad i = 2, 3, \ldots, N \tag{63}$$

Theorem 3: A GRF defined over a rectangular lattice I is a translation invariant GRF if and only if it is MC-GRF with homogeneous LTF such that:

$$\sum_z \sigma_{11}(z)\sigma_h(v, z) = \sigma_{11}(v) \tag{64}$$

$$\sum_z \sigma_{11}(z)\sigma_v(y, z) = \sigma_{11}(y) \tag{65}$$

$$\sum_y \sigma_v(y, z)\sigma(x, t, z, y) = \sigma_v(x, t) \tag{66}$$

for every $x, t, z \in I_A$ and

$$\sum_t \sigma_h(t, z)\sigma(x, t, z, y) = \sigma_h(x, y) \tag{67}$$

Simulation of a translation invariant GRF is therefore based on simulation of a translation invariant MC-GRF. Simulation of MC-GRF is simplified task due to causal type of model, free boundary condition and the special form of global energy function (56). Therefore we can lexicographically generate MC-GRF from a set of known LTF. Simulation started from specified σ_v, σ_h. The initial probability σ_{11} is computed from (64),(65), then the probability $\sigma(x, t, z, y)$ is calculated from (66),(67). It has to be solved $2(K-1)K^2$ equations with $(K-1)^3$ unknowns. In the case of translation invariant and isotropic GRF additional constrains (68)-(70) should be satisfied.

$$\sigma_v(v, z) = \sigma_h(v, z) = \sigma_*(v, z) \tag{68}$$

$$\sigma_*(v, z) = \sigma_*(z, v) \tag{69}$$

$$\sigma(x, t, z, y)\sigma_*(t, z) = \sigma(t, x, y, z)\sigma_*(x, y) = \sigma(t, z, y, x)\sigma_*(x, y) \tag{70}$$

3.7.3. Binomial Markov random field model

We shall assume the conditional probability of point r having the class k to be binomial.

$$p(Y_r = k | Y_s \forall s \in I_r) = \binom{K-1}{k} \theta^k (1-\theta)^{K-1-k} \tag{71}$$

where

$$\theta = \exp(T)/(1 + \exp(T)) \tag{72}$$

In binary case ($K = 2$) the conditional probability is:

$$p(Y_r = k | Y_s \forall s \in I_r) = \exp(kT)/(1 + \exp(T)) \tag{73}$$

A first-order model has the form for T:

$$T_1 = a_1 + a_2(Y_{m,n-1} + Y_{m,n+1}) + a_3(Y_{m-1,n} + Y_{m+1,n}) \tag{74}$$

A second-order model is

$$T_2 = T_1 + a_4(Y_{m-1,n-1} + Y_{m+1,n+1}) + a_5(Y_{m-1,n+1} + Y_{m+1,n-1}) \tag{75}$$

, a third-order model is

$$T_3 = T_2 + a_6(Y_{m-2,n} + Y_{m+2,n}) + a_7(Y_{m,n-2} + Y_{m,n+2}) \tag{76}$$

etc.

Simulation of BMRF can be done for example using the Metropolis algorithm. First we generate RF from uniform random generator, then using relaxation Metropolis algorithm RF will converge to our specified BMRF.

3.7.4. Monte Carlo simulation of binary MRF

Monte Carlo method can be used be used for generating prescribed MRF if number of possible MRF conditional probabilities (pixel classes) is reasonably small [32].

a) generation of 2D independent identically distributed binary random array on the toriodal lattice

b) 2D random array is converted into thematic map, using sliding through the corresponding MRF neighborhood configuration

c) thematic map is converted into two vectors, location vector in which pixels are ordered according to class membership and lookup vector in which the pixels are arranged in the order of their location on the lattice

d) using the flipping mechanism [32], based on comparision between theoretical and empirical conditional probabilities, we change frequences of occurence possible pixel classes, until the theoretical (given) probabilities are reached

3.7.5. Metropolis algorithm

Given the state of MRF Y^t , another random configuration X is chosen. Ratio

$$\alpha = p(X)/p(Y^t)$$

is computed. If $\alpha > 1$, then $Y^{t+1} = X$, otherwise the transition is made with probability α. A variable ξ is selected from standardized uniform distribution, if $\xi \leq \alpha$ then $Y^{t+1} = X$, otherwise $Y^{t+1} = Y^t$. Modification of algorithm is the Exchange algorithm [12], where X is obtained from Y^t by exchanging values of two randomly chosen pixels. The disadvantage of this method is sensitivity on initial configuration [11]. In the "single-flop" [28] algorithm X is obtained from Y^t by changing value of one randomly chosen pixel.

3.7.6. Gibbs sampler

The Gibbs sampler [28] generates realization from a given MRF using relaxation technique, similar to the Metropolis algorithm. The stationary configuration Y^0 is arbitrary. Using repeatedly visiting all sites (for example by raster scanning) we always replace one pixel with value generated from local characteristic of Gibbs distribution.

$$p(Y^t) = p(Y_r^t | Y_s^{t-1} \forall s \neq r) p(Y_s^{t-1} \forall s \neq r) \tag{77}$$

where

$$p(Y_r^t | Y_s^{t-1} \forall s \neq r) = \exp\{-Q_r^t(Y)\}/\acute{Z} \tag{78}$$

Convergence of the algorithm is assured by the relaxation theorem.

Theorem: Assume that for each site from lattice the visit sequence contains this site infinitely often. Then for every starting configuration Y^0 and every configuration Y

$$\lim_{t \to \infty} p(Y^t = Y | Y^0) = \exp\{-Q(Y)\}/Z$$

Proof see [28].

4. Texture analysis

In this chapter we address the problem of parameter estimation for models described in the chapter three. In this attitude we differ from the usual meaning of a texture analysis problem, where the goal is not to reproduce a given texture but to discriminate between several different textures instead. Texture model parameters can serve as a basis for texture discrimination too.

4.1. Random mozaic models analysis

If there is defined a distance between two classes (for example gray levels), we can modify definition of the variogram ([54],[55]) of a random mozaic.

$$V(d) = E\{(Y_r - Y_{r+d})^2\} \tag{79}$$

In a special case of constant gray level in each cell and uncorrelated gray levels in different cells the variogram can be simplified [54]:

$$V(d) = 2\sigma^2(1 - P(d)) \tag{80}$$

where σ^2 is assumed to be the standard deviation of texture's classes (gray levels) and $P(d)$ is the probability that two points distance d apart are in the same cell. For the Poisson line model 3.1.1. and rotated checkerboard model 3.1.2. we can compute for a given σ^2 theoretical variogram as a function of the Poisson parameter τ or checkerboard square size b. given variogram of a real texture, these parameters can be obtained using fitting technique.

4.3. Inference of syntactic models

In order to model image of interest , it is necessary to have the stochastic tree grammar actually inferred from the available image samples. Such an inference procedure requires the inference of both the tree grammar and its production probabilities. Unfortunately, a general inference procedure for stochastic tree grammars does not exist and is still a subject of research [24]. Only some very special cases can be treated by existing inference algorithms, like for example [24]:

a) A windowed patterns are grouped into clusters using a similarity measure.

b) Each cluster consists of a finite set of trees having the same structure. A stochastic tree grammar is then inferred for each cluster.

c) The final texture grammar is the union of placement rules and all stochastic tree grammars.

Using such an algorithm is still difficult to solve grammar inference for more then very simple structures, therefore is difficult to model natural-like textures and algorithm is computationally demanding due to emploing large set searching procedure.

4.4. Fractal models

The fractal dimension of Brownian function can be measured directly or from Fourier power spectrum $P(z)$ of Y_r as the spectral density of fractal Brownian function is proportional [47] to z^{-2H-1}.

4.4.1. Peleg's method

The brigtness field of a texture is imagined to form a surface with brightness represented by height. The method [46] is based on the volume v , occupied by all points distant ϵ or less from the surface. The surface area is obtained by dividing the volume by 2ϵ and is found to be a function of ϵ. The method uses the increase in apparent area $A(\epsilon)$ at each increment of ϵ to obtain a signature for the texture from which the relation (81) can be fitted to obtain H.

$$A(\epsilon) = F\epsilon^{(H-1)} \tag{81}$$

4.4.2. Pentland's method

If the texture is fractal , then (8) must hold so we can estimate the fractal dimension from relation (82),

$$E\{|Y_{r+d} - Y_r|\}||d||^{-H} = E\{|Y_{r+\acute{d}} - Y_r|\} \tag{82}$$

where $||\acute{d}|| = 1$. Using (82) and a least square regression for different d , we can estimate H . If the relation (82) holds, then the viewed surface must be a 3D fractal Brownian surface and can be modeled using this fractal model.

4.5. Second order statistic model

The set of second-order joint density functions $p_{s,\acute{s}}(Y_1, Y_2)$ can be estimated from natural texture field using coocurence matrices computed from a given texture, first order statistics are estimated from corresponding histogram.

4.6. Autoregressive model parameter estimation

Similarly as in GMRF case the ML estimates require numerical optimization method due to log-likelihood function, which is nonquadratic in γ. The ML estimate for σ^2 is also (86). To avoid computationally expensive numerical methods, it is possible to use LS, MPL or some approximation method like 4.6.2.

4.6.1. The prediction error variance minimization

If we define model prediction:

$$\tilde{Y}_r = E\{Y_r\} = \gamma X_r \tag{83}$$

The parameters are now adjusted to minimize the variance of the prediction error:

$$E\{(Y_r - \tilde{Y}_r)^2\} \tag{84}$$

The LS solution can be found in the form :

$$\hat{\gamma} = E\{X_r X_r^T\}^{-1} E\{X_r Y_r\} = [\sum X_r X_r^T]^{-1} \sum X_r Y_r \tag{85}$$

$$\sigma^2 = \frac{1}{MN} \sum_r (Y_r - \hat{\gamma} X_r)^2 \tag{86}$$

The drawback of LS estimator (85) is its nonconsistency for nonunilateral neighborhoods [37].

4.6.2. The iterative estimation method

This method is ML estimation with quadratic approximation of determinant [37].

$$\gamma^{(n+1)} = (R - \sigma_n^{-2} D)^{-1} (V - \sigma_n^{-2} W) \tag{87}$$

$$\sigma_n^2 = \frac{1}{MN} \sum_r (Y_r - \gamma^{(n)} \acute{X}_r)^2 \tag{88}$$

$$\acute{X}_r = X_r - [\hat{\mu}, \ldots, \hat{\mu}]^T \tag{89}$$

$$\hat{\mu} = \frac{1}{MN} \sum_r Y_r \tag{90}$$

$$D = \sum_r \acute{X}_r \acute{X}_r^T \tag{91}$$

$$W = \sum_r \acute{X}_r (Y_r - \hat{\mu}) \tag{92}$$

$$V = \sum_r C_r \tag{93}$$

$$R = \sum_r (S_r S_r^T - C_r C_r^T) \tag{94}$$

$$C_r = [\cos(2\pi (MN)^{-1/2} (r-1)^T s), \forall s \in I_r]^T \tag{95}$$

$$S_r = [\sin(2\pi (MN)^{-1/2} (r-1)^T s), \forall s \in I_r]^T \tag{96}$$

Where $\gamma^{(0)} = D^{-1} W$.

4.6.3. Estimation of the SAR model with conditional expectations

To estimate parameters in model (23), we can use the prediction error variance minimization. The solution [20] is similar with (85),(86)

$$\hat{\gamma} = [E\{X_r X_r^T\}]^{-1} E\{X_r Y_r\} = W^{-1} V \tag{97}$$

,

where

$$\begin{aligned} w_{ij} &= E\{E\{Y_r | Y_s \forall s \in I_r^i\} E\{Y_r | Y_s \forall s \in I_r^j\}\} = \\ &= \sum_{Y_s \forall s \in I_r^i \cup I_r^j} E\{Y_r | Y_s \forall s \in I_r^i\} E\{Y_r | Y_s \forall s \in I_r^j\} p(Y_s \forall s \in I_r^i \cup I_r^j) \end{aligned} \tag{98}$$

and

$$v_i = E\{E\{Y_r | Y_s \forall s \in I_r^i\} Y_r\} = \sum_{Y_r, Y_s \forall s \in I_r^i} Y_r E\{Y_r | Y_s \forall s \in I_r^i\} p(Y_r, Y_s \forall s \in I_r^i) \tag{99}$$

.

The only complication in comparison with (85),(86) is knowledge of joint density function in (98),(99), which has to be estimated from given real texture data. Conditional expectations are computed from (100).

$$E\{Y_r | Y_s \forall s \in I_r^i\} = \sum_{Y_r = 0}^{K-1} Y_r p(Y_r | Y_s \forall s \in I_r^i) \tag{100}$$

Parameter estimation is computationaly demanding, especially for larger contextual neighborhoods. Also memory requirements of model (23) are higher than for simpler model (14).

4.7. MRF parameter estimation

ML parameter estimation of MRF model is complicated by difficulty associated with computing the normalization constant Z. Generaly we have K^{MN} possible realizations for which $\exp\{-Q\}$ has to be computed. Therefore except for trivial tasks it is necessary to use some approximation method, like for example the coding method. To overcame the difficulties caused by mutual correlation of lattice pixels, the coding method [5] codes lattice pixels to obtain independent lattice pixels in a given contextual neighborhood. Such a coding for a first-order scheme, for example is:

$$\begin{array}{ccc} . & \times & . \\ \times & . & \times \\ . & \times & . \end{array}$$

Using such a coding system, variables associated with $\times$,(.) sites are according to the Markov assumption mutually independent. A disadvantage of this method is that the estimates thus obtained are not efficient [5],[37] due to a partial utilization of the data. The different coding estimates for the same parameter can in some cases considerably differ.

ML estimate for . sites is obtained from

$$\max_{\gamma} \prod_{r \in .} p(Y_r | \forall Y_s \in \times) \tag{101}$$

or

$$\max_{\gamma}\{-(\log \acute{Z} + Y_r G_r + \sum_{s \in I_r} G_{r,s} Y_s Y_r)\} \tag{102}$$

This maximazitation is nonlinear and an iterative solution is needed . For the case of small neighborhood, few possible levels and a low order model, there is possible to approximate (102) by a set of linear equations, if we approximate $p(Y_r | Y_{(r)})$ by its frequency of occurence.

4.7.1. GMRF parameter estimation

A normalization constant (31) is easy obtainable for a GMRF model, so we can obtain ML estimate maximizing either (37) or (46). The variance ML estimate for $U = 0$ is:

$$\sigma^2 = \frac{1}{MN} \acute{Y}^T \Psi \acute{Y} \tag{103}$$

To obtain Ψ estimation we have to solve

$$\min_{\Psi}\{-\frac{1}{MN} \ln |\Psi| + \ln(\acute{Y}^T \Psi \acute{Y})\}$$

. This estimator is nonlinear (determinant) so an iterative solution is needed. Another computationaly attractive possibility is the coding method (104), where $\acute{M}, \acute{N}$ is number of independent rows and columns, respectively. The likelihood function can be simplified:

$$L = p(Y) = \prod_{r=1}^{\acute{M}\acute{N}} p(Y_r | Y_s \forall s \in I_r)$$

$$= (2\pi)^{-\acute{M}\acute{N}/2} \sigma^{-\acute{M}\acute{N}} \exp\{-1/2\sigma^{-2} \sum_{r=1}^{\acute{M}\acute{N}} (Y_r - \mu - \sum_{s \in I_r} a_s (Y_{r-s} - \mu))^2\} \tag{104}$$

Parameters are determined from equations:

$$\frac{\delta \log L}{\delta a_s} = 0$$

$$\frac{\delta \log L}{\delta \sigma^2} = 0$$

From this equations we obtain:

$$\frac{1}{\acute{M}\acute{N}} \sum_{r=1}^{\acute{M}\acute{N}} \acute{Y}_r \acute{Y}_{r-s} = \sum_{s \in I_r} a_s \frac{1}{\acute{M}\acute{N}} \sum_{r=1}^{\acute{M}\acute{N}} \acute{Y}_{r-s}^2 \tag{105}$$

$$\sigma^2 = \frac{1}{\acute{M}\acute{N}} \sum_{r=1}^{\acute{M}\acute{N}} (\acute{Y}_r - \sum_{s \in I_r} a_s \acute{Y}_{r-s})^2 \tag{106}$$

$$\mu = \mu_s = \frac{1}{\acute{M}\acute{N}} \sum_{r=1}^{\acute{M}\acute{N}} \acute{Y}_r \tag{107}$$

The final estimates of the parameters is the average value over all the possible codings (i.e. first-order process two codings, second-order process four codings, third and fourth-order nine codings etc.). Another possibility is the pseudo-likelihood estimator.

$$\max_{\gamma} \prod_{\forall r \in I} p(Y_r | Y_{(r)}) \tag{108}$$

The pseudolikelihood estimate for a_s parameters has form

$$\gamma_a = [a_{10}, a_{01}, \ldots] = [\sum_{\forall r \in I} X_r^T X_r]^{-1} \sum_{\forall r \in I} X_r^T \acute{Y}_r \tag{109}$$

where

$$X_r = [\acute{Y}_{i-1,j} + \acute{Y}_{i+1,j}, \acute{Y}_{i,j-1} + \acute{Y}_{i,j+1}, \ldots] \tag{110}$$

Under a torus structure , the likelihood function of Y is given [19].

$$p(\acute{Y}|\gamma) = \prod_{r \in I} (1/2MNS_{\acute{Y}}(r))^{1/2} \exp\{-\sum_{r \in I} |\mathcal{F}\{\acute{Y}_r\}|^2 / 2MNS_{\acute{Y}}(r)\} \tag{111}$$

The ML estimates of GMRF parameters are obtained maximizing likelihood function (111), where $\mathcal{F}\{\acute{Y}_r\}$ is defined in (45) and $S_{\acute{Y}}(r)$ in (44).

Woods [58] has given an estimate for $\tilde{M} \times \tilde{N}$ sublattice $\acute{I}$.

$$I_B = \{r : r \in I \wedge \exists s \in I_r (r - s) \notin I\}$$

$$\acute{I} = I - I_B$$

$$\gamma_a = [\sum_{\forall r \in \acute{I}} X_r^T X_r]^{-1} \sum_{\forall r \in \acute{I}} X_r^T \acute{Y}_r \tag{112}$$

$$\sigma^2 = \frac{1}{\tilde{M}\tilde{N}} \sum_{r=1}^{\tilde{M}\tilde{N}} (\acute{Y}_r - \gamma_a X_r^T)^2 \tag{113}$$

The estimate (112) was shown [37] to be asymptotically consistent.

4.7.2. ML estimation of MC-GRF parameters

The ML estimation of the LTF of a MC-GRF with a homogeneous LTF is [30]:

$$\sigma_h(i,j) = \frac{\nu_{i,j}}{\sum_{i=1}^{K} \nu_{i,j}} \tag{114}$$

$$\sigma_v(i,j) = \frac{\epsilon_{i,j}}{\sum_{i=1}^{K} \epsilon_{i,j}} \tag{115}$$

$$\sigma(i,j,k,l) = \frac{\eta_{i,j,k,l}}{\sum_{i=1}^{K} \eta_{i,j,k,l}} \tag{116}$$

where $\nu_{i,j}$ is the number of observed transitions in the first picture row from state j to state i, $\epsilon_{i,j}$ is corresponding number for the first column and $\eta_{i,j,k,l}$ is the number of observed transitions from states (j,k,l) to state i on remaining pixels. Estimation (114)-(116) are consistent , efficient and asymptotically Gaussian.

4.7.3. ML parameter estimation of binomial MRF

Using the coding technique, log-likelihood function can be written in form (117).

$$L = \sum_{r=1}^{\acute{M}\acute{N}} \log p(Y_r | Y_s \quad \forall s \in I_r) =$$

$$= \sum_{r=1}^{\acute{M}\acute{N}} \{\log \binom{K-1}{i} + i(T - \log(1-\exp T)) + (K-1-i)\log(1 - \frac{\exp T}{1+\exp T})\} \tag{117}$$

$$\frac{\delta L}{\delta a_i} = \sum_{r=1}^{\acute{M}\acute{N}} \{i(\frac{\delta T}{\delta a_i} + \frac{\exp T}{1-\exp T}\frac{\delta T}{\delta a_i}) - (K-1-i)\frac{\exp T}{1+\exp T}\frac{\delta T}{\delta a_i}\} \tag{118}$$

Maximalization of (118) is demanding task using numerical methods. Another possibility is the MPL method then $\acute{M} = M, \acute{N} = N$.

5. Software description

Our programs and procedures are written in Sun C. All programs assume generation of texture with maximum format 256*256 pixels , all resulting random fields are stored in standard ASCII disk file 256*256 in rowwise manner using procedure PRE. Programs are aimed for synthetizing random fields using SAR or GMRF model (chapters 3.6,3.7.1,3.7.6.,4.6.,4.7. and 4.7.1.). For this software description we accepted the NAG IPAL library documentation conventions with minor modifications. Several procedures are taken over from [51], where can be found their listings and precise explanations.

---

1. Name

AAE

2. Purpose

fills stack with window parameters for a random field synthesis, checks for entry errors

3. Specification

int aae(ncl,ncf,nrl,nrf,nc,nr,iw)

4. Description

Procedure fills stack with random field control size parameters. On exit returns an error code. Column range wrong (returns 1), row range wrong (returns 2), last column greater than column range (returns 3), last row greater then row range (returns 4). If parameters O.K. (returns 0).

5. Parameters

NCL - INT. On entry contains a number of the last image column.

NCF - INT. On entry contains a number of the first image column.

NRL - INT. On entry contains a number of the last image row.

NRF - INT. On entry contains a number of the first image row.

NC - INT. On entry contains a total number of image columns.

NR - INT. On entry contains a total number of image rows.

IW[6] - INT. On exit contains image window parameters in above described order.

6. Subroutine used

None.

---

1. Name

ARGEN

2. Purpose

simultaneous autoregressive model generator

3. Specification

main

4. Description

A simultaneous autoregressive model generator using the iterative cojugate gradient method (3.6.) in Fletcher-Reeves version. Parameters are assumed to be symmetric and the AR model can be up to fifth order. Checking of the stability condition.

5. Parameters

Input parameters are interactively read during processing.

6. Subroutine used

aae,ipar,gasdev,gauss,prar,pre,ran1

---

1. Name

ARNSGEN

2. Purpose

simultaneous autoregressive model generator

3. Specification

main

4. Description

A simultaneous autoregressive model generator using the Woods iterative algorithm (3.6.1.) . Program can generate up to 20 different random fields , each having maximum 25 non zero parameters. All parameters are first read in and then all generation are sequentially processed. AR models are assumed to be general asymmetrical or symmetrical of any order.

5. Parameters

Model parameters and corresponding row and column shifts are interactively read.

6. Subroutine used

aae,bpar,free_imatrix,free_vector,gasdev,gauss,imatrix,prarns,pre,ran1,vector,

---

1. Name

BPAR

2. Purpose

filling parameter stacks with RF input parameters

3. Specification

bpar(itg,it,or,mean,is,par,x,ca)

4. Description

Subroutine interactively reads parameters up to 20 different random fields generated with symmetrical or asymmetrical model, maximal number of parameters of each field can be 25, each can correspond to any lattice shift.

5. Parameters

ITG - INT. On input a number of random fields to be generated.

IT[20] - INT. On exit contains number of iteration for each RF.

OR[20] - INT. On exit contains number of parameters in each RF. Numbers are negative for symmetrical models and positive for asymmetrical ones.

MEAN[20] - INT. On exit contains mean value of each RF.

IS[20][25][2] - INT. On exit contains row and column shifts for each RF.

PAR[20][26] - FLOAT. On exit contains parameters of each RF.

X[20] - FLOAT. On exit contains (1-2sum of parameters)*mean for each RF.

CA[20][40] - CHAR. On exit contains file names of all generated RF.

6. Subroutine used

None.

---

1. Name

FOURN

2. Purpose

n dimensional fast Fourier transformation

3. Specification

void fourn(data,nn,ndim,isign)

4. Description

Fast Fourier transformation subroutine [51].

5. Parameters

DATA[] - FLOAT. On input complex n-dimensional data stored ..real,imaginary.., the data are stored in such a way, that the rightmost index of array increases most rapidly. On output the resulting transformed data.

NN[] - INT. On input vector containing the lengths of each dimension.

NDIM - INT. On input a number of dimensions.

ISIGN - INT.On input key for transformation direction (-1 inverse, 1 forward).

6. Subroutine used

None.

---

1. **Names**

   FREE-IMATRIX FREE-MATRIX FREE-IVECTOR FREE-VECTOR

2. **Purpose**

   deallocation of corresponding memory space

3. **Specification**

   void free_imatrix(m,nrl,nrh,ncl,nch)

   void free_matrix(m,nrl,nrh,ncl,nch)

   void free_ivector(v,nl,nh)

   void free_vector(v,nl,nh)

4. **Description**

   Deallocation of memory space for float matrix, vector and int matrix,vector [51].

5. **Parameters**

   **M - INT / FLOAT. Matrix previously allocated.

   NRL - INT. On input an index of the last row.

   NRH - INT. On input an index of the first row.

   NCL - INT. On input an index of the last column.

   NCH - INT. On input an index of the first column.

   *V - INT / FLOAT. Vector previously allocated.

   NL - INT. On input an index of the last column.

   NH - INT. On input an index of the first column.

6. **Subroutine used**

   None.

---

1. **Name**

   GASDEV

2. **Purpose**

   standard Gaussian generator

3. **Specification**

   float gasdev(idum)

4. **Description**

   Subroutine computes random values from standard Gaussian distribution using Box-Muller method [51].

5. Parameters

*IDUM - INT. On input initialize random sequence if negative.

6. Subroutine used

ran1

---

1. Name

GAUSS

2. Purpose

independent standard Gaussian random field

3. Specification

gauss(iw,a)

4. Description

Subroutine computes a realization of independent standard Gaussian random field on lattice described in iw.

5. Parameters

IW[6] - INT. On input random field window parameters (see AAE).

A[256][256] - FLOAT. On output contains resulting random field in corresponding location and size.

6. Subroutine used

gasdev,ran1

---

1. Name

GIBBS

2. Purpose

a Gaussian-Markov random field generator

3. Specification

main

4. Description

Program generates a sequence of realizations of Gaussian-Markov random fields using the Gibbs sampler algorithm (3.7.1.,3.7.6.). Program can generate up to 20 realizations of different random fields each having maximal number of 25 parameters (up to ninth order). All necessary parameters all read interactively first, the same parameters for different realizations can be mutually copied, afterwards all realizations are computed.

5. Parameters

An interactive input.

6. Subroutine used

aae,free_imatrix,free_vector,gasdev,gauss,imatrix,prarns,pre,ran1,vector

---

1. Name

GMFFTGEN

2. Purpose

a Gaussian-Markov random field generator

3. Specification

main

4. Description

Program generates a Gaussian-Markov random field using Fourier transformation method (3.7.1.). GMRF can be up to fifth order.

5. Parameters

GMRF parameters read interactively.

6. Subroutine used

aae,fourn,free_vector,gasdev,ipar,pre,psd,ran1,vector

---

1. Name

GMRFGEN

2. Purpose

a simultaneous autoregressive model or a Gaussian-Markov random field generator

3. Specification

main

4. Description

Program generates a random field from a simultaneous autoregressive model or a Gaussian-Markov model both up to fifth order using Woods iterative algorithm (3.6.1.). An autoregressive model is assumed to be symmetrical.

5. Parameters

Parameters are read interactively.

6. Subroutine used

aae,gasdev,gauss,ipar,prar,pre,ran1

---

1. **Names**

   IMATRIX IVECTOR MATRIX VECTOR

2. **Purpose**

   allocation of memory space

3. **Specification**

   int **imatrix(nrl,nrh,ncl,nch)

   int *ivector(nl,nh)

   float **matrix(nrl,nrh,ncl,nch)

   float *vector(nl,nh)

4. **Description**

   Subroutines allocate memory space for int matrix ,vector or for float matrix,vector [51].

5. **Parameters**

   see description of free_imatrix,free_ivector

6. **Subroutine used**

   None.

---

1. **Name**

   IPAR

2. **Purpose**

   interactive input of model parametrs

3. **Specification**

   ipar(p,ord)

4. **Description**

   Subroutine interactively read and store parameters up to fifth order of symmetrical space model.

5. **Parameters**

   P[13] - FLOAT. On output contains model parameters.

   ORD - INT. On input contains an order of model.

6. **Subroutine used**

   None.

---

1. **Name**

   LUBKSB

2. Purpose

solution of the set of linear equations

3. Specification

void lubksb(a,n,indx,b)

4. Description

Subroutine solves a set of linear equations using Cholesky decomposition [51].

5. Parameters

**A - FLOAT. On input a Cholesky factors of coefficient matrix. Unchanged on exit.

N - INT. On input a number of equations.

*INDX - INT. On input permutation vector from ludcmp. Unchanged on exit.

B[] - FLOAT. On input a vector of right side, on exit returns the solution.

6. Subroutine used

None.

---

1. Name

LUDCMP

2. Purpose

Cholesky decomposition of matrix

3. Specification

void ludcmp(a,n,indx,d)

4. Description

Subroutine computes the Cholesky decomposition of a given nxn matrix a [51].

5. Parameters

**A - FLOAT. On input contains a matrix to be factorize, on exit contains in lower triangular part of matrix the left Cholesky factor and in upper triangular part of matrix the right Cholesky factor.

N - INT. On input cointains a size of matrix a.

*INDX - INT. On output contains row exchanges.

*D - FLOAT. On ouput +1 if number of row permutations was even, otherwise -1.

6. Subroutine used

free_vector,vector

---

1. Name

MINV

2. Purpose

matrix inversion

3. Specification

minv(a,n)

4. Description

Subroutine computes the inversion of regular matrix a using Cholesky factorization method.

5. Parameters

**A - FLOAT. On input contains a matrix to be inverted, on output its inversion.

N - INT. On input contains a size of inverted matrix.

6. Subroutine used

free_ivector,free_matrix,free_vector,ivector,lubksb,ludcmp,matrix,vector

---

1. Name

MLI

2. Purpose

estimation of a symmetric AR model parameters

3. Specification

main

4. Description

Program computes the Kashyap-Chellappa iterative approximation of maximum likelihood estimation of a simultaneous autoregressive model parameters (4.6.2.). A symmetric AR model assumed.

5. Parameters

Control parameters are interactively read in.

6. Subroutine used

aae,free_imatrix,free_matrix,free_vector,imatrix,ludcmp,matrix, minv,pre,xas,xar

---

1. Name

MLINS

2. Purpose

estimation of an asymmetric AR model parameters

3. Specification

main

4. Description

Program computes the Kashyap-Chellappa iterative approximation of maximum likelihood estimation of a simultaneous autoregressive model parameters (4.6.2.). An asymmetric AR model assumed.

5. Parameters

Control parameters are interactively read in.

6. Subroutine used

aae,free_imatrix,free_matrix,free_vector,imatrix,ludcmp,matrix, minv,pre,xans,xarns None.

---

1. Name

MPL

2. Purpose

AR , GMRF parameter estimation

3. Specification

main

4. Description

Program computes the maximum pseudolikelihood or Woods estimation of parameters of symmetric AR or GMRF models up to fifth order.

5. Parameters

Control parameters are interactively read during the processing.

6. Subroutine used

aae,free_matrix,free_vector,ludcmp,matrix,minv,pre,vector,xar

---

1. Name

MPLNS

2. Purpose

AR , GMRF parameter estimation

3. Specification

main

4. Description

Program computes the maximum pseudolikelihood or Woods estimation of parameters of symmetric AR or GMRF models with any order, or asymmetric AR model with any shifts.

5. Parameters

Control parameters are interactively read during the processing.

6. Subroutine used

aae,free_imatrix,free_matrix,free_vector,imatrix, ludcmp,matrix,minv,pre,vector,xarns

---

1. Name

PRAR

2. Purpose

prediction of symmetrical model

3. Specification

float prar(iw,a,i,j,p,ord)

4. Description

Subroutine computes the prediction of a symmetrical model up to fifth order in i-th row , j-th column of random field. Toroidal lattice assumed.

5. Parameters

IW[6] - INT. On input field location and size parameters.

A[256][256] - FLOAT. On input random field. Unchanged on exit.

I - INT. On input a row index.

J - INT. On input a column index.

P[13] - FLOAT. On input model parameters. Unchanged on exit.

ORD - INT. On input an order (¡=5) of symmetrical model.

6. Subroutine used

None.

---

1. Name

PRARNS

2. Purpose

prediction of symmetrical/asymmetrical model

3. Specification

float prarns(iw,a,i,j,p,ish,ord)

4. Description

Subroutine computes the prediction of an asymmetrical or a symmetrical model up any order in i-th row , j-th column of random field. Toroidal lattice assumed.

5. **Parameters**

IW[6] - INT. On input field location and size parameters.

A[256][256] - FLOAT. On input random field. Unchanged on exit.

I - INT. On input a row index.

J - INT. On input a column index.

*P - FLOAT. On input model parameters. Unchanged on exit.

**ISH - INT. On input contains the parameter shifts.

ORD - INT. On input contains a number of non zero parameters.

6. **Subroutine used**

None.

---

1. **Name**

PRE

2. **Purpose**

input - output routine

3. **Specification**

pre(key,a,ix,c)

4. **Description**

Subroutine stores or retrieves an image file in ASCII form, maximal 256x256 pixel format and rowwise manner (readable for example for SCILIMAGE system). An image is supposed to be in 256 gray levels, and routine converts image into or from a float field.

5. **Parameters**

KEY - INT. On input contains the control key (key=1 reading, key !=1 writing).

A[256][256] - FLOAT. On input an image to be stored. Unchanged on exit. In retrieval case on output contains an image from disk.

IX - INT. On input a image size (square shape assumed) to be stored or retrieved.

C[] - CHAR. On input a disk file name.

6. **Subroutine used**

None.

---

1. **Name**

RAN1

2. **Purpose**

uniform random number generator

3. Specification

float ran1(idum)

4. Description

Subroutine generates pseudorandom numbers from uniform distribution [51].

5. Parameters

*IDUM - INT. On input if IDUM¡0 then initialization of generator.

6. Subroutine used

None.

---

1. Name

XANS

2. Purpose

fill stacks with sin and cos data centered in specified lattice shifts

3. Specification

xans(iw,i,j,ps,pc,ish,ord)

4. Description

Subroutine fills stack ps with sin data and pc with cos data centered in specified lattice positions. It is assumed to have toroidal lattice and a symmetrical or an asymmetrical neighborhood.

5. Parameters

IW[6] - INT. On input field location and size parameters. Unchanged on exit.

I - INT. On input a row index.

J - INT. On input a column index.

*PS - FLOAT. On output sin data.

*PC - FLOAT. On output cos data.

**ISH - INT. On input contains the parameter shifts. Unchanged on exit.

ORD - INT. On input contains a number of contextual neighbours. If ord¡0 symmetrical neighborhood assumed, otherwise an asymmetrical one.

6. Subroutine used

None.

---

1. Name

XAR

2. Purpose

fill stack with centered random field data from specified lattice locations and assuming a symmetric neighborhood

3. Specification

xar(iw,a,i,j,p,ish,ord,key,mean)

4. Description

Subroutine fills stack p with centered data from field (a) corresponding to data locations from a symmetrical neighborhood of maximal fifth order. Data lattice can be toroidal or with zero border condition. data centered in specified lattice positions.

5. Parameters

IW[6] - INT. On input field location and size parameters. Unchanged on exit.

A[256][256] - FLOAT. On input a data field. Unchanged on exit.

I - INT. On input a row index.

J - INT. On input a column index.

*P - FLOAT. On output contains centered data from symmetrical neighborhood.

ORD - INT. On input contains an order of neighborhood.

KEY - INT. If on input contains key=1 toroid lattice assumed, if key=2 zero border condition instead.

MEAN - FLOAT. On input contains a RF mean value.

6. Subroutine used

None.

---

1. Name

XARNS

2. Purpose

fill stack with centered random field data from specified lattice locations of any type of neighborhood

3. Specification

xarns(iw,a,i,j,p,ish,key,mean,ord)

4. Description

Subroutine fills stack p with centered data from field (a) corresponding to data locations (ish) . Data lattice can be toroidal or with zero border condition.

5. Parameters

IW[6] - INT. On input field location and size parameters. Unchanged on exit.

A[256][256] - FLOAT. On input a data field. Unchanged on exit.

I - INT. On input a row index.

J - INT. On input a column index.

*P - FLOAT. On output contains selected data.

**ISH - INT. On input contains the data shifts relative to I,J. Unchanged on exit.

KEY - INT. If on input contains key=1 toroid lattice assumed, if key=2 zero border condition instead.

MEAN - FLOAT. On input contains a RF mean value.

ORD - INT. On input contains a number of contextual neighbours. If ord¡0 symmetrical neighborhood assumed, otherwise an asymmetrical one.

6. Subroutine used

None.

---

6. Experimental results and conclusions

The main disadvantage of random mosaic models is that they produce a geometrical shapes (straight lines, rectangulars, circles) which are not appropriate to model natural textures. They can be used for limited modelling of certain made-made textures. Periodic tessellated models can be also useful mainly for generation man-made regular structures , it is also possible to fill single cells with data from random field generator. Syntactic models at present state of art are not possible to use for general texture generation because of unsolved grammar inference problem. Fractal models have an advantage that they can be approximated by a single fractal function over a range of scales and the second-order statistics are described by a single parameter. Such properties sound promising so further study would be necessary to find how credible are natural textures modelled by fractals and in what range are real fractals scale invariant.

Second order models were based on assumption (later found to be wrong) that the second order statistics are sufficient for a texture description. Although such an approximation is good enough for large amount of textures, synthesis step is done iteratively and is time consuming. Autoregressive models (AR) are highly flexible to model natural textures , their advantage over equivalent Gaussian Markov random field models (GMRF) is usually lesser number of parameters. On the contrary GMRF are more general, for some GMRF does not exist an finite order AR model. It is possible to model textures also using a causal autoregressive model, which can be generated directly and for which we can easily find ML or Bayesian parameter estimation, but in such a case we have to limit spatial correlation and introduce causal direction which is in most 2D cases unnatural. AR model on toroidal lattice can be generated using fast Fourier transformation. There is a possibility to use more complicated ARMA models for texture generation, their main problem is highly nonlinear parameter estimation. A disadvantage of AR model with conditional expectations is a large number of parameters and therefore problems with identifying and storing them as well as time consuming synthesis. Markov random field are the most flexible from all described possibilities to model natural-like textures. Problems of general MRF are difficulty with evaluation of normalizing constant, iterative synthesis and non-linear parameter estimation. An exception is the GMRF whose normalizing constant is easy to evaluate and if we assume the toroidal lattice, then we can use rapid FFT based synthesis algorithm. MC-MRF has an advantage with easy synthesis and analysis its disadvantage is a large number of parameters.

Analytical step of a texture synthetiser is not critical, because in graphical applications can be solved off-line . It is assumed that a graphical system will have a database of different textures parameters and will be able to use or modify textures from this basic sortiment. On the other hand the synthesis step is done in real time and therefore is necessary to generate textures as quickly as possible. Iteration based algorithms like for example algorithms of the Metropolis type are not very handy especially for larger data fields, because they are rather time consuming. Another their disadvantage is that although these algorithms converge to specified field, little is known about their convergence rate. Such a problem for limited texture sortiment of a graphical system can be solved experimentaly. GMRF generator , as well as AR generator , based on the fast Fourier transformation is especially advantageous because it needs only one inverse FFT to generate desired random field, what is much faster that any other iterative generator. Another advantage of this method is possibility to use special FFT processor to build a real time GMRF (AR) generator.

Although time is not important for the analytical step of a synthetizer, ML parameter estimation needs in the most cases of AR,MRF models numerical optimization, so we have no guarantee to reach a global optimum . In our experimentation seems MPL method (or Woods method) rapid and sufficient alternative at least for lower order models. The coding method of Besag uses only fraction of available data and offers no clear advantage over MPL. Similarly the iterative estimation method (4.6.2.) was in several experiments slowly converging.

We have tested Gaussian Markov random fields generators and simultaneous autoregressive model generators of relatively low order (up to fifth order) with four exceptions of ninth order Gaussian Markov random fields to generate natural textures from Brodatz album. We did not have a possibility to test both models on real digitized natural textures, so our examples in Fig.1. - Fig.24. are generated using parameters published in literature [10][38][19] or with our own experimentation. Even this limited experimentation confirms our belief of large flexibility and usefulness for natural texture modelling of both models.

Fig.1.-Fig.12. show different realizations of GMRF from parameters published [10], all were synthetized using the Gibbs sampler algorithm. Fig.1.,Fig.12. are the first order GMRF, Fig.2.,Fig.5.,Fig.6. are the second order GMRF , Fig.3.,Fig.4.,Fig.7.,Fig.10.,Fig.11. are modelled using the third order GMRF model and Fig.8.,Fig.9. are from the fifth order model. When we compare our results with the published ones , we can see that the convergence rate is different for each RF. While Fig.2.,Fig.3.,Fig.5.,Fig.10. and Fig.11. are already near their limiting RF after 30 iterations, others needed 70 iteration steps and for Fig.7. even this number is not sufficient. Fig.13. - Fig.18. were modelled using non causal symmetric AR model of the first, second or the third order and the Woods iterative algorithm (100 iterations). Parameters for these experiments were taken from [38] . Fig.6. demonstrates a violation of the stability condition of Woods algorithm. Fig.1. - Fig.18. were generated with the same mean value and dispersion and when we compare GMRF results with AR fields generated with similar corresponding parameters (9,14—5,15—3,13) we can see their mutual similarity demonstrating close relation between both models. Fig.19. - Fig.22. are realizations of ninth order GMRF. Fig.19. is wood grain , Fig.20. canvas and Fig.21. calf-leather all modelled using parameters from digitized Brodatz album textures [19]. Fig.23. is a second order GMRF synthetised using FFT method and Fig.24. is a third order AR realization.

Realistic modelling of natural surfaces needs color texture synthesis and combining several texture synthesis methods. It is possible to model macro features using periodic tesselated models or simple syntactic models with micro textures generated using RF models. Another tool for combining macro and micro features is Fourier transformation.

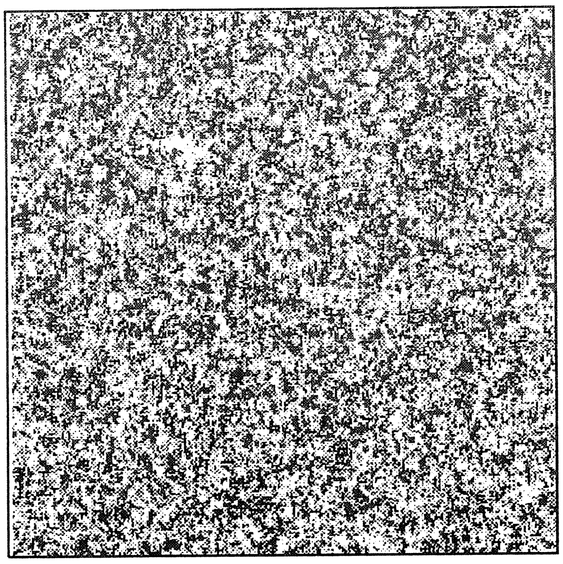
Fig.1.

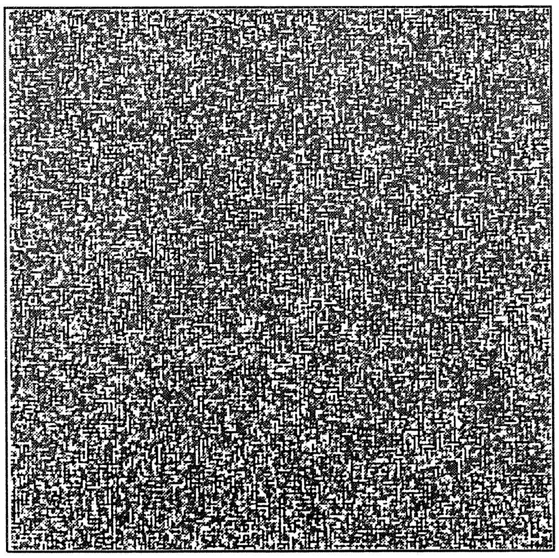
Fig.2.

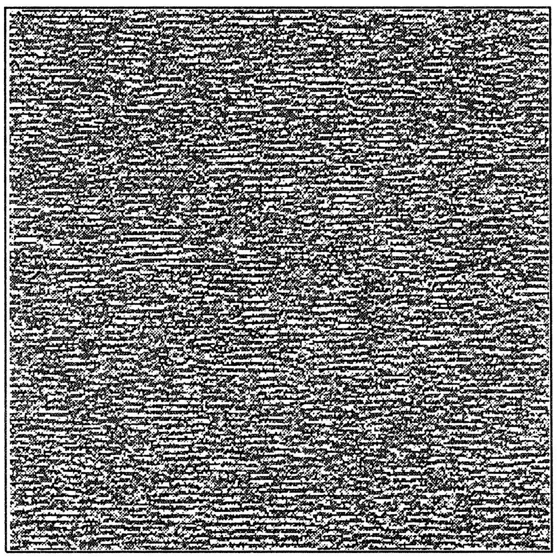
Fig.3.

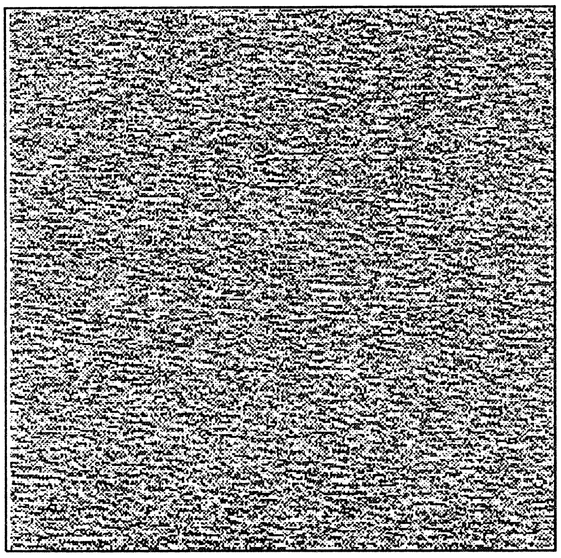
Fig.4.

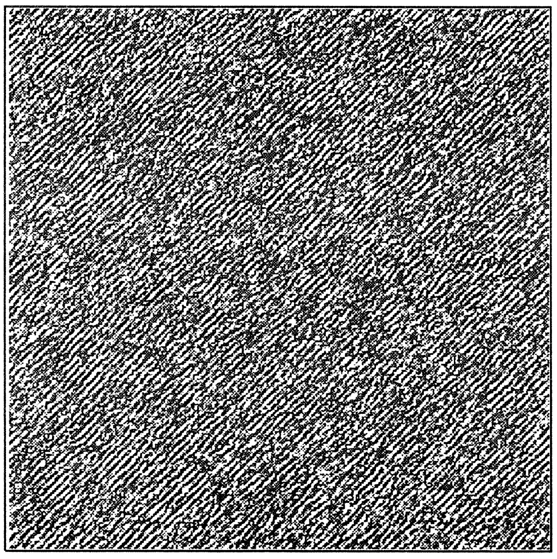
Fig.5.

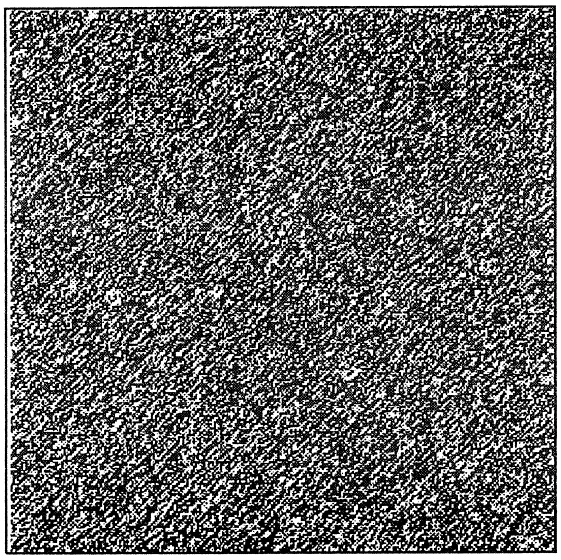
Fig.6.

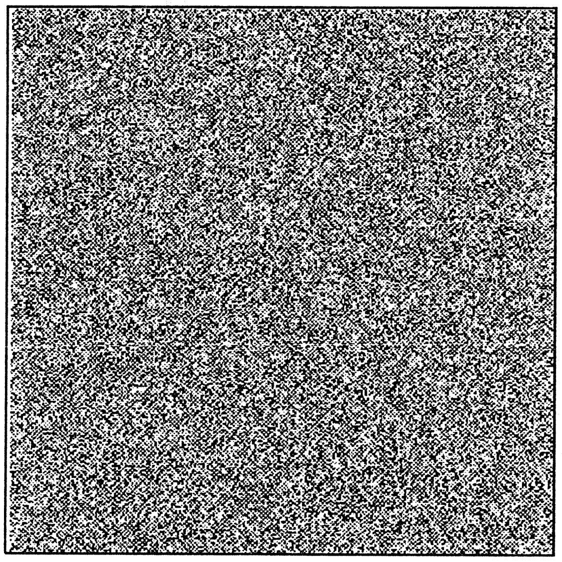
Fig.7.

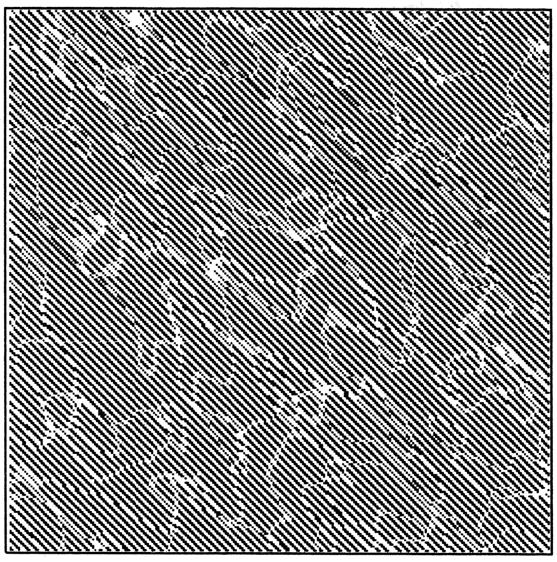
Fig.8.

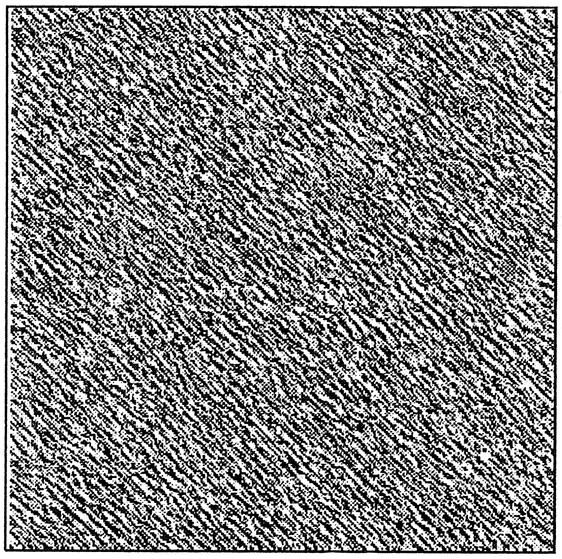
Fig.9.

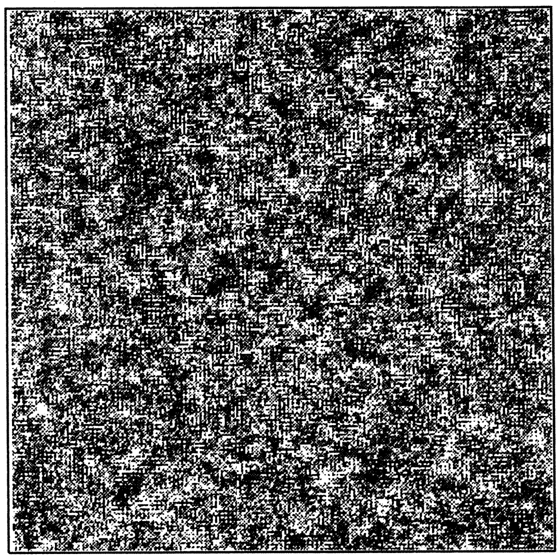
Fig.10.

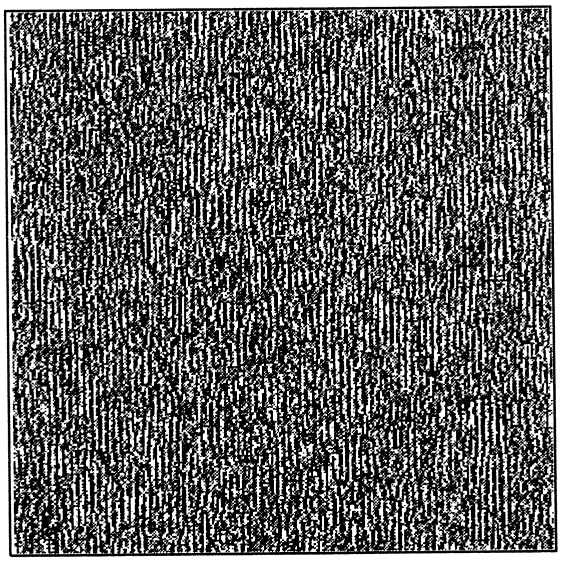
Fig.11.

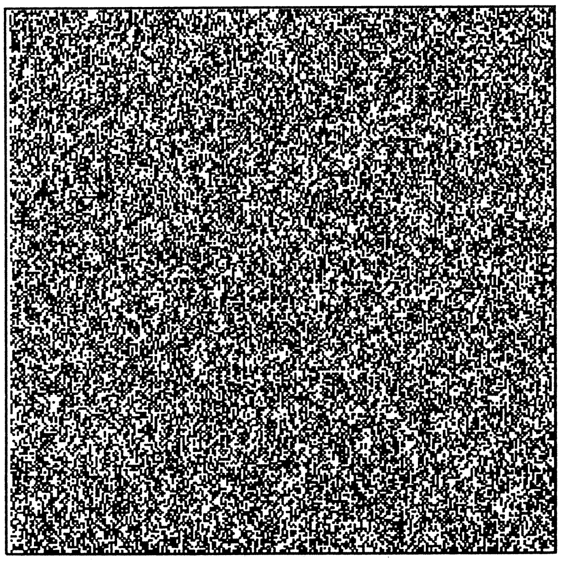
Fig.12.

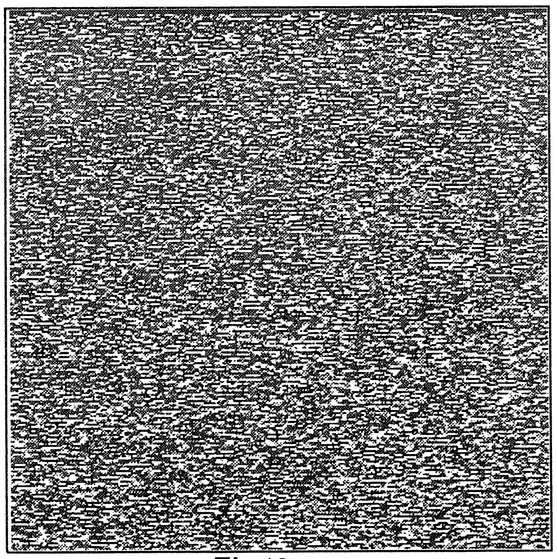
Fig.13.

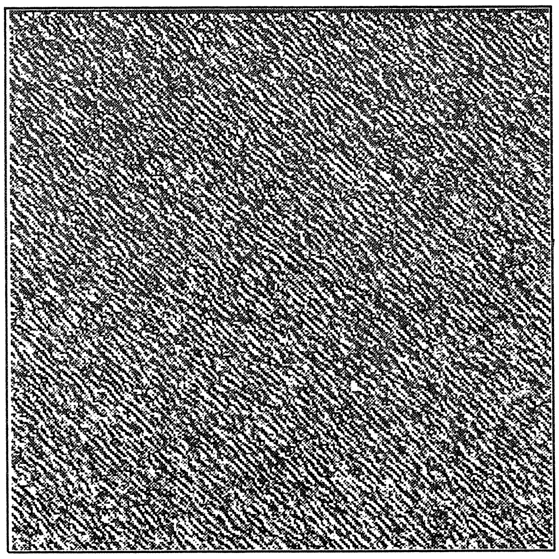
Fig.14.

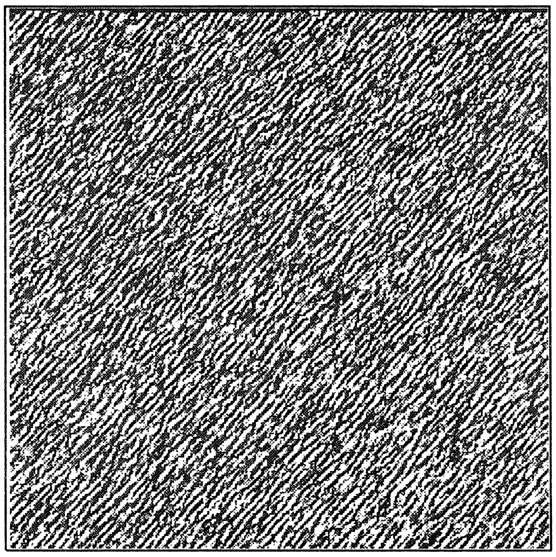
Fig.15.

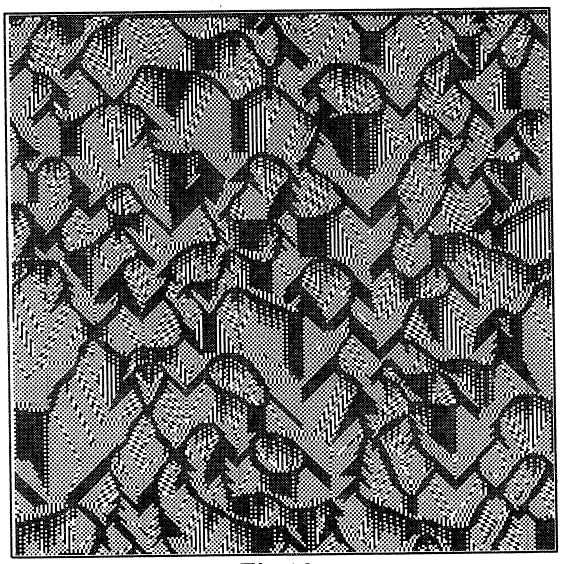
Fig.16.

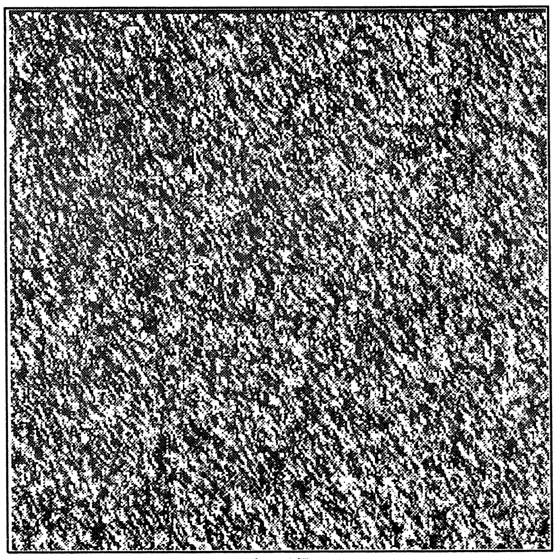
Fig.17.

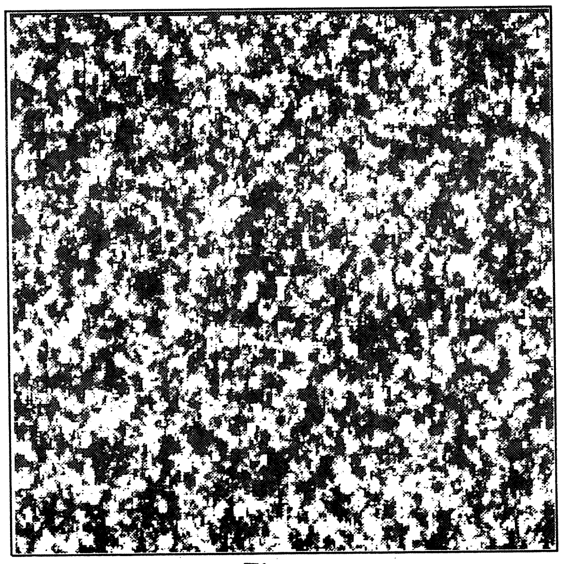
Fig.18.

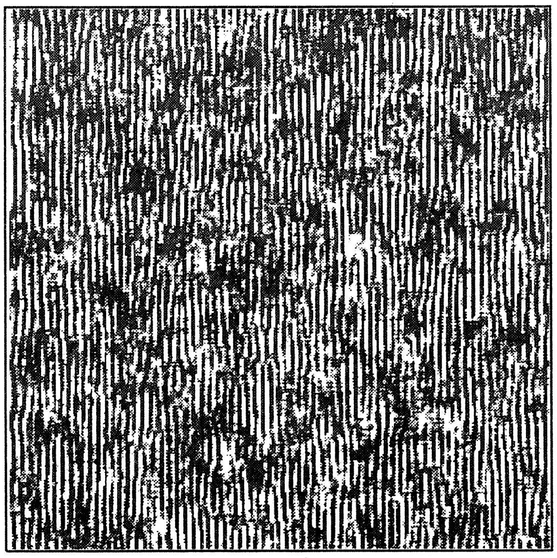
Fig.19.

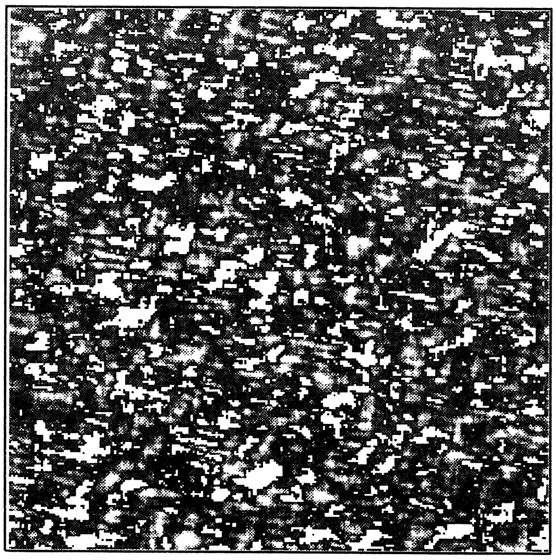
Fig.20.

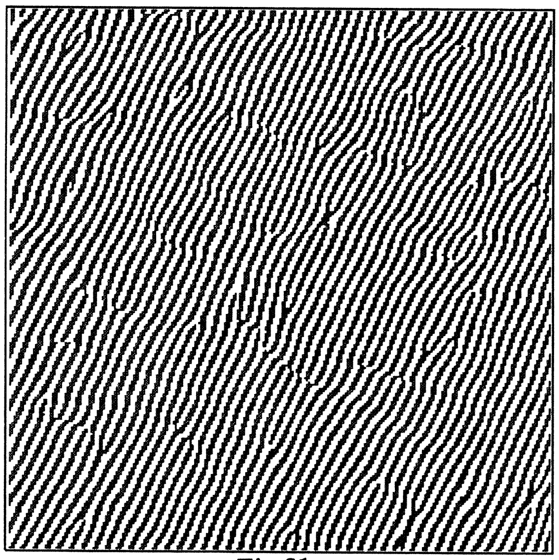
Fig.21.

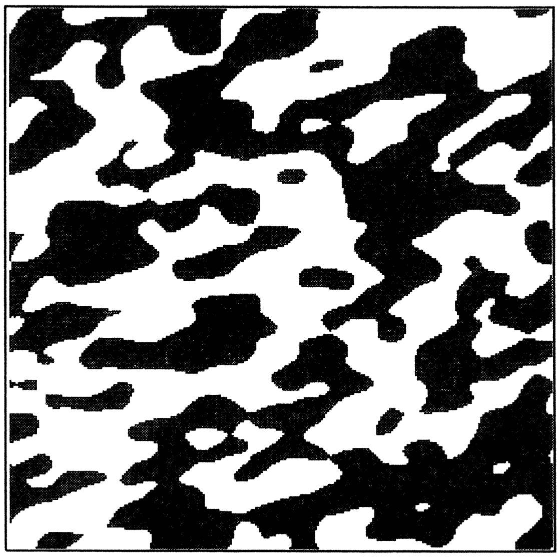
Fig.22.

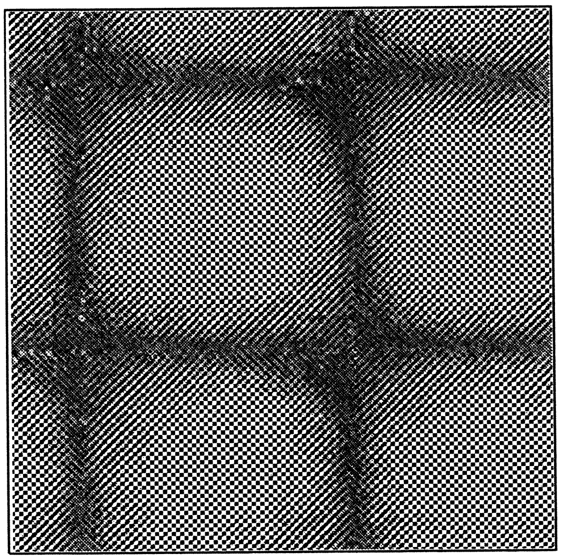
Fig.23.

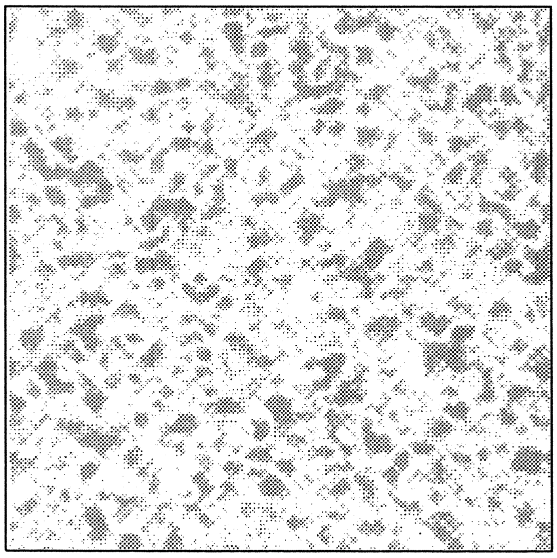
Fig.24.

7. References

1 Ahuja, N.,A. Rosenfeld " Mosaic models for textures ," *IEEE Trans. Pattern Anal. Mach. Int.* **PAMI–3**, pp.1–11, 1981.

2 Barnsley, M., " Fractals Everywhere , " Academic Press , 1988.

3 Bennis, C., A.Gagalowicz " Hierarchical texture synthesis on 3-D surfaces," in *Eurographics '89* , ed. W.Hansmann F.R.A. Hopgood and W.Strasser , pp.257–269, 1989.

4 Bennis, C., A.Gagalowicz " 2-D Macroscopic Texture Synthesis ," *Computer Graphics Forum* , **8**, pp.291–300, 1989.

5 Besag, J., " Spatial Interaction and the Statistical Analysis of Lattice Systems ," *J. Royal Stat. Soc.*, **B–36**, pp.192–236, 1974.

6 Besag, J., " On the Statistical Analysis of Dirty Pictures ," *J. Royal Stat. Soc.*, **B–48**, pp.259–302, 1986.

7 Bouman, C.,B. Liu " Multiple Resolution Segmentation of Textured Images," *IEEE Trans. Pattern Anal. Mach. Int.* **PAMI–13**, pp.99–113, 1991.

8 Cano, D., T.H. Minh " Texture synthesis using hierarchical linear transformation," *Signal Processing*, **SP–15**, pp.131–148, 1988.

9 Chellappa, R., R.L. Kashyap " Texture synthesis using spatial interaction models ," *Proc. of IEEE Comp. Soc. Conf. on Pattern Rec.,Inf. Proc.*, Las Vegas, pp.226–230, 1982.

10 Chellappa, R., "Two-dimensional discrete Gaussian Markov random field models for image processing " in *Progress in Pattern Recognition 2*, ed. L.N. Kanal A.Rosenfeld, Elsevier, North-Holland, 1985.

11 Cohen, F.S., "Markov random fields for image modelling and analysis" in *Modeling and Application of Stochastic Processes*, ed. U.B. Desai, Kluwer Academic Publishers, Boston, 1986.

12 Cross, G.R., A.K. Jain "Markov Random Field Texture Models," *IEEE Trans. Pattern Anal. Mach. Int.*,**PAMI–5**, pp.25–39, 1983.

13 Deguchi, K., I. Morishita " Two-dimensional auto-regressive model for the representation of random image fields ," *Proc. 6-th Int.Conf. on Pattern Recognition*, Munich, pp.90–93, 1982.

14 Demco, S., L.Hodges, B.Naylor " Construction of fractal objects with iterated function systems," *Computer Graphics* **19**, pp.271–278, 1985.

15 Derin, H., A.W. Cole " Segmentation of textured images using Gibbs random fields ," *Comp. Graphics Image Proc.* **CGIP–35**, pp.72–97, 1986.

16 Derin, H., H. Elliot "Modeling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Trans. Pattern Anal. Mach. Int.*,**PAMI–9**, pp.39–55, 1987.

17 Dodd, N. , " Multispectral Texture Synthesis Using Fractal Concepts," *IEEE Trans. Pattern Anal. Mach. Int.*, **PAMI–9**, pp.703–707, 1987.

18 Englert, G., G. Sakas " A Model for Description and Synthesis of Heterogeneous Textures," in *Eurographics '89* , ed. W.Hansmann F.R.A. Hopgood and W.Strasser , pp.245–255, 1989.

19 Fan, Z.,F.S.Cohen,M.A. Patel, "Rotated and Scaled Textured Images using Markov Random Field Models," *IEEE Trans. Pattern Anal. Mach. Int.*,**PAMI–13**, pp.191–202, 1991.

20 Faugeras, O.D., "Autoregressive modeling with conditional expectations for texture synthesis," *Proc. 5-th Int.Conf. on Pattern Recognition*, Miami Beach, pp.792–782, 1980.

21 Faugeras, O.D., O.D. Garber "Algebraic reconstruction techniques for texture synthesis," *Proc. 5-th Int.Conf. on Pattern Recognition*, Miami Beach, pp.792–782, 1980.

22 Francos, J.M., A.Z. Meiri " A Unified Structural-Stochastic Model for Texture Analysis and Synthesis ," *Proc. 9-th Int.Conf. on Pattern Recognition*, Washington, pp.41–46, 1988.

23 Fu,K.S. , S.Y. Lu " Computer generation of texture using a syntactic approach," *Computer Graphic*, **12**, pp.147–152, 1978.

24 Fu, K., " Syntactic Image Modelling Using Stochastic Tree Grammars ," *Comp. Graphics Image Proc.* **CGIP–12**, pp.136–152, 1980.

25 Gagalowicz, A., S. Ma " Synthesis of natural textures ," *Proc. 6-th Int.Conf. on Pattern Recognition*, Munich, pp.1081–1086, 1982.

26 Gagalowicz, A., S. Ma " Sequential synthesis of natural textures ," *Comp. Graphics Image Proc.* **CGIP–30**, pp.289–315, 1985.

27 Gagalowicz,A. , " A New Method for Texture Fields Synthesis: Some Applications to the Study of Human Vision ," *IEEE Trans. Pattern Anal. Mach. Int.*, **PAMI-3** , pp.520–533, 1981.

28 Geman,S. , D. Geman " Stochastic Relaxation , Gibbs Distributions and Bayesian Restoration of Images," *IEEE Trans. Pattern Anal. Mach. Int.*, **PAMI-6** , pp.721–741, 1984.

29 Gidas,B., " A renormalization Group Approach to Image Processing problems," *IEEE Trans. Pattern Anal. Mach. Int.*, **PAMI–11**, pp.164–180, 1989.

30 Goutsias, J., "Mutually compactible Gibbs images: properties, simulation and identification," *Proc. ICASSP 88*, New York, vol. 2, pp.1020–1023, 1988.

31 Hammersley, J.M., D.C.Handscomb " Monte Carlo Methods , " Methuen , London, 1964.

32 Hassner, M., J. Sklansky " The Use of Markov Random Fields as Models of Texture ," *Comp. Graphics Image Proc.* **CGIP–12**, pp.357–370, 1980.

33 Heckbert, P.S. , " Survey of texture mapping ," *IEEE Computer Graphics and Applications*, **11**, pp.56–67, 1986.

34 Jain, A. , " Advances in mathematical models for image processing ," *Proc. IEEE* , **69**, pp.502–528, 1981.

35 Julesz, B. , " Textons , the elements of texture perception and their interactions ," *Nature* **290**, pp.91–97, 1981. vskip -3truemm

36 Kaneko, H., E. Yodogawa " A Markov Random Field Application to Texture Classification," *Proc. IEEE Conf. Pattern Recognition and Image Processing*, Las Vegas, 1982.

37 Kashyap, R.L., R. Chellappa "Estimation and Choice of Neighbors in Spatial-Interaction Models of Images," *IEEE Trans. Inf. Theory*,**IT–29**, pp.60–72, 1983.

38 Kashyap, R.L., "Analysis and Synthesis of Image Patterns by Spatial Interaction Models " in *Progress in Pattern Recognition 1*, ed. L.N. Kanal A.Rosenfeld, Elsevier, North-Holland, 1981.

39 Kaufman, A., " TSL - a Texture Synthesis Language," *Visual Computer* **4**, pp.148–158, 1988.

40 Kaufman, A., S.Azaria " Texture synthesis techniques for computer graphics," *Computer and Graphics* **9**, pp.139–145, 1985.

41 Kindermann, R., J.L.Snell " Markov Random Fields and their Applications, " American Mathematical Society , Providence, 1980.

42 Lu, S.Y., K.S. Fu " A syntactic approach to texture analysis," *Comp. Graphics Image Proc.* **CGIP–7**, pp.303–330, 1978.

43 Magnenat-Thalmann, N., D.Thalmann" Image Synthesis, " Springer , Tokyo, 1987.

44 Mandelbrot, B.B., " The Fractal Geometry of Nature , " Freeman , San Francisco, 1982.

45 Peachey, D.R., " Solid Texturing of Complex Surfaces," *Computer Graphics* **19**, pp.279–286, 1985.

46 Peleg,S.,J.Naor,R.Hartley and D.Avnir " Multiple resolution texture analysis and classification ," *IEEE Trans. Pattern Anal. Mach. Int.*, **PAMI–6**, pp.518–523, 1984.

47 Pentland,A.P., " Fractal-based description of natural scenes," *IEEE Trans. Pattern Anal. Mach. Int.*, **PAMI–6**, pp.661–674, 1984.

48 Perlin, K., " An image synthetizer," *Computer Graphics* **19**, pp.287–296, 1985.

49 Pickard, D.K., " A curious binary lattice process," *J. Appl.Probability* **14**, pp.717–731, 1977.

50 Pratt, W.K. ,O.D.Faugeras, A. Gagalowicz " Application of Stochastic Texture Field Models to Image Processing ," *Proc. IEEE* , **69**, pp.542–551, 1981.

51 Press,W.H.,B.P.Flannery, S.A.Teukolsky and W.T.Vetterling, " Numerical recipes in C, " Cambridge Univ. Press, New York, 1989.

52 Preston, Ch.J., " Gibbs states on countable sets, " Cambridge Univ. Press, London, 1974.

53 Qian, W. and D.M.Titterington " Multidimensional Markov Chain Models for Image Textures ," *J. Royal Stat. Soc.*, **B–53**, pp.661–674, 1991.

54 Schachter, B.J., A. Rosenfeld and L.S. Davis, "Random Mosaic Models for Textures," *IEEE Trans. Syst. Man Cyb.*,**SMC–8**, pp.694–702, 1978.

55 Schachter, B.J., N. Ahuja " Random pattern generation processing," *Comp. Graphics Image Proc.* **CGIP–10**, pp.95–114, 1979.

56 Voss R.F., "Random fractal forgeries" in *Fundamental Algorithms for Computer Graphics*, ed. R. A. Earnshaw , NATO ASI Series F, vol.17, Springer, Hiedelberg, 1985.

57 van Wijk, J.J., " Spot noise texture synthesis for data visualization," *Computer Graphics* **25**, 1991.

58 Woods, J.W., "Two-Dimensional Discrete Markovian Fields ," *IEEE Trans. Inf. Theory*, **IT–18**, pp.232–240, 1972.