GENERATION AND PARAMETER ESTIMATION OF MARKOV RANDOM FIELD TEXTURES AND A PARALLEL NETWORK FOR TEXTURE GENERATION

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Mehmet İzzet GÜRELLİ
February, 1990

GENERATION AND PARAMETER ESTIMATION OF MARKOV RANDOM FIELD TEXTURES AND A PARALLEL NETWORK FOR TEXTURE GENERATION

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Mehmet İzzet Gürelli

February, 1990

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Levent Onural (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Erdal Arıkan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. A. Enis Çetin

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

To my family

ABSTRACT

GENERATION AND PARAMETER ESTIMATION OF MARKOV RANDOM FIELD TEXTURES AND A PARALLEL NETWORK FOR TEXTURE GENERATION

Mehmet İzzet Gürelli
M.S. in Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Levent Onural
February, 1990

In this thesis, a special class of Markov random fields (MRF), which is defined on two dimensional pixel arrays and represented by a few numbers called the MRF parameters, is studied as a texture model. Specifically, the generation of sample MRF textures and estimation of MRF texture parameters are considered. For the generation of sample MRF textures, an algorithm that can be implemented in a parallel manner is developed together with a parallel network which implements the algorithm. A mathematical description of the algorithm, based on finite state Markov chains is given and the structure of the network is explained. For the estimation of MRF texture parameters, a method based on histogramming of a sample MRF texture is studied and a mathematical justification of the method is given. Generation and parameter estimation methods studied in this thesis are tested by some computer programs and the results are observed to be satisfactory for many purposes.

Keywords: Markov random fields, Gibbs random fields, texture modeling, image modeling, parallel networks.

ÖZET

MARKOV RASTGELE ALANI DOKULARININ ÜRETİMİ, PARAMETRELERİNİN KESTİRİMİ VE DOKU ÜRETİMİ İÇİN PARALEL, AĞ YAPILI BİR DEVRE

Mehmet İzzet Gürelli
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Levent Onural
Şubat, 1990

Bu tezde, Markov rastgele alanlarının (MRA) iki boyutlu piksel dizileri üzerinde tanımlanan ve MRA parametreleri adı verilen birkaç sayı ile ifade edilebilen özel bir grubu, bir doku modeli olarak ele alınıp incelenmiştir. Özellikle, örnek MRA dokularının üretilmeleri ve MRA doku parametrelerinin kestirimi üzerinde durulmuştur. Örnek MRA dokularının üretilmeleri için, paralel biçimde gerçekleştirilebilen bir algoritma, bu algoritmayı gerçekleştiren paralel, ağ yapılı bir devre ile birlikte geliştirilmiştir. Algoritmanın matematiksel temeli, sonlu durumları olan bir Markov zinciri olarak verilmiş ve paralel devrenin yapısı anlatılmıştır. MRA doku parametrelerinin kestirimi için, örnek bir MRA dokusunun histogramlanması temeline dayalı bir metod incelenmiştir. Bu tezde ele alınan doku üretimi ve parametre kestirimi metodları bazı bilgisayar programları ile denenmiş ve sonuçların pekçok amaç için yeterli olduğu gözlenmiştir.

Anahtar sözcükler. Markov rastgele alanları, Gibbs rastgele alanları, doku modelleme, görüntü modelleme, paralel ağlar.

ACKNOWLEDGEMENT

I am indebted to Assoc. Prof. Dr. Levent Onural for his encouragement, guidance, and invaluable suggestions during my study.

I would also like to gratefully acknowledge the other members of my M.S. thesis committee: Assoc. Prof. Dr. Erdal Arıkan and Assist. Prof. Dr. A. Enis Çetin.

Finally, it is my pleasure to express my thanks to Mr. Semih Kolukısa for his help in preparing some of the drawings, and to my friends, particularly Seyfullah Halit Oğuz, Mehmet Tankut Özgen, and Mustafa Karaman for their continuous encouragement and some valuable discussions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis deals with the Markov random field (MRF) texture model. More specifically, generation of sample MRF textures and estimation of MRF texture parameters are studied. Also, a parallel network is developed for sample MRF texture generation.

MRFs provide an important mathematical tool for texture modeling [1]–[3]. Texture modeling, being a branch of image modeling, deals with the development of models or methods for the characterization of textured images.

Texture modeling is an important problem in image analysis and processing in general. This is partly because the analysis and processing of textured images generally require much different methods than those of non-textured images. Therefore some tools for the characterization of textured images become essential in many cases. Some of the basic areas of image analysis and processing where texture models play an important role are the classification, data compression and segmentation of textured images.

Classification of textured images may be required in applications where one should decide on the type of a texture which appears in some image. Data compression is required both for the storage and transmission of images. In such a case, a texture model which associates a few parameters to the textured image may result in a high data compression rate. These parameters may be

used to regenerate the textured image through the same texture model at a later time. Another use of texture models may be in applications which require the segmentation or boundary detection of textured images. Segmentation or boundary detection of textured images generally require different methods than those of non-textured images [3]–[5]. This is because the boundaries of different textured regions are generally determined by the changes of some textural features instead of changes in the average brightness level.

Before developing or choosing a texture model for a specific application, it may be a good starting point to define what a texture is or with what kinds of textures that specific application will deal. Unfortunately, it is very difficult to give a precise definition of a texture. This difficulty is mainly due to the existence of an extremely large variety of features that a texture may possess. In this case, one may start by choosing a rather special group of textures to work with.

In the literature, there has been a variety of approaches to the problem of texture modeling [1],[2],[6]–[8]. An approach has been the use of random mosaic models [6]. A random mosaic model, basically, consists of tessellating an image region into cells and then assigning gray levels to each of these cells. The rules for tessellating the image region and assigning gray levels to these cells may be quite versatile. Brief descriptions of the methods in the literature that have been proposed to generate random mosaics may be found in [6].

Another approach to texture modeling is to consider the textured image as a realization of a random field. The probability assignment rule on the random field may be given in the form of conditional probabilities or in the form of joint probabilities as will be discussed in Chapter 2 of this thesis. An important texture model of this group is the MRF texture model [1]–[3]. The MRF texture model is studied in this thesis.

An MRF is basically a spatial interaction scheme in which the conditional probability assignment of a pixel value given the values of the other pixels in the textured image depends only on the values of the pixels lying in the neighborhood of the corresponding pixel. From this point of view, the model

model has been applied to the modeling of some real textures [1].

In this thesis, we limited our attention to binary MRF textures which are defined on finite rectangular pixel arrays. However, most of the work may be generalized to non-binary (but discrete valued) MRF textures defined on a finite set of spatial points. The work is based on a specific type of conditional probability distribution which will be described in Chapter 2.

In this thesis, mainly two aspects of MRF texture modeling, the generation of sample MRF textures having the specified parameters and the estimation of MRF texture parameters from a given sample texture, are considered. Also a parallel network is developed for MRF texture generation. The organization of the thesis is described below.

In Chapter 2, basic mathematical background is reviewed and some graphical concepts are introduced. Then, the definitions of MRFs and Gibbs random fields (GRF) are given. Following a theorem on the MRF-GRF equivalence, the chapter is continued with the descriptions of specific MRF models which will be considered in this thesis.

In Chapter 3, two algorithms for the generation of sample MRF textures are given. The first one is a common sequential algorithm, whereas the second one is suitable for parallel implementation. We designed the second algorithm to be used in the development of a parallel network for the generation of sample MRF textures as described in Chapter 5. A detailed mathematical analysis and a proof of this algorithm is also included. Some examples of MRF textures generated on a computer by these algorithms are presented in this chapter.

In Chapter 4, a method for the estimation of MRF texture parameters from a given sample MRF texture is described and a mathematical justification of this parameter estimation method is given. This chapter also contains several experimental results.

In Chapter 5, a parallel network for the generation of sample MRF textures having a specified set of parameters is proposed. The network runs on discrete

time steps and implements the parallel MRF texture generation algorithm described in Chapter 3. The main structure of the network is described and a specific realization is explained.

In Chapter 6, some conclusions and comments about the material of this thesis are given.

Finally, some programs that are used for the generation and parameter estimation of sample MRF textures are included in the appendices. All of these programs are written in C programming language.

The reliability of the experimental results are highly dependent on the quality of the pseudo-random number generators used in the sample MRF texture generation programs. In order to achieve reliable observations, we used several different methods of pseudo-random number generation. In all cases, the sample textures generated with the same sets of parameters were not only visually similar but the accuracies of the estimated parameters obtained from sufficiently large image regions (such as $128 * 128$ or larger pixel arrays) were also similar.

Chapter 2

Mathematical Background

2.1 Introduction

The aim of this chapter is to introduce the basic mathematical background that is used in this thesis. More specifically, a general information on random fields and the definitions of some related graphical concepts (such as neighborhood structure, clique, etc.) are given. Following the definition of a Markov random field (MRF), the definition of a Gibbs random field (GRF) is given. The chapter is continued with a theorem on MRF-GRF equivalence and the descriptions of some specific MRF models that will be used in this thesis. Some of the notational conventions that are used in this thesis are also given in this chapter.

2.2 Some Notes on Random Fields and Graphical Concepts

A random field may be roughly defined as a set of random variables assigned to the elements of a set of spatial points. The structure of the set of spatial points and the nature of the random variables assigned to these points may be quite versatile. For example, consider the case where the spatial points are placed on a plane. The placement may be regular or irregular, discrete or continuous

or any combination of them. Furthermore, the random variables assigned to these spatial points may be discrete or continuous or any combination of them. More mathematical descriptions and deeper analysis of random fields may be found in [9]–[11].

In this thesis, we will assume that the spatial points are the pixels of an image which are placed regularly on a rectangular grid and we will denote this set of pixels by $\mathcal{D}$. Furthermore, we will assume that the number of pixels in $\mathcal{D}$ is finite and much larger than one. Typical images of our concern may contain 64*64 pixels or more. Also, the random variables assigned to the pixels of $\mathcal{D}$ will be assumed to be binary such that they may take on values only from the set $\{0, 1\}$.

A useful concept in defining the interactions between the random variables assigned to the points in a random field is the concept of a neighborhood. A neighborhood system on a given set $\mathcal{D}$ associated with the concept of a neighborhood η_i of a point i in $\mathcal{D}$ is defined as follows [3].

Definition: A collection of subsets of $\mathcal{D}$ given by

$$\underline{n} = \{\eta_i : i \in \mathcal{D},\ \ \eta_i \subset \mathcal{D}\}$$

is a *neighborhood system* on the set $\mathcal{D}$ if and only if the *neighborhood*, η_i, of a point i is such that

$$i \notin \eta_i\ , \quad \text{and}$$

$$j \in \eta_i \Longrightarrow i \in \eta_j \quad \forall i \in \mathcal{D}\ .$$

In this thesis, the shape of the neighborhood of an interior point of $\mathcal{D}$ will be assumed to be independent from the position of the point in the image region, and therefore, unless the neighborhood of a specific point in $\mathcal{D}$ is referred to, the subscript i of η_i will be omitted and η will be called a *neighborhood structure* on $\mathcal{D}$.

Some possible neighborhood structures for the rectangular array of pixels $\mathcal{D}$ are shown in Figure 2.1. Generally, the neighborhood structure shown in Figure 2.1.a is referred to as the *first order neighborhood structure* and the one

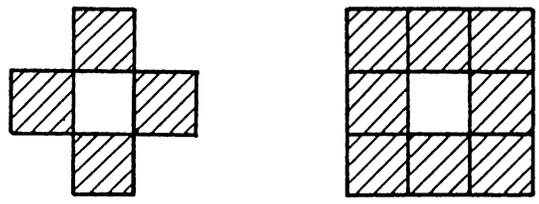


Figure 2.1. Valid neighborhood structures, (a) first order, (b) second order.

shown in Figure 2.1.b is referred to as the *second order neighborhood structure.* In Figures 2.1.a and 2.1.b, the shaded pixels are the neighbors of the center pixels.

Another basic definition commonly used in defining MRFs and GRFs is the definition of a clique as given below [3].

Definition: A clique of the pair $(\mathcal{D},\eta)$, denoted by c, is a subset of $\mathcal{D}$ such that:

1) c consists of a single pixel, or
2) for $i \neq j$, $i \in c$ and $j \in c$ implies that $i \in \eta_j$.

Note that the definition of a clique on $\mathcal{D}$ is dependent on how the neighborhood structure η is defined. Therefore, a subset of $\mathcal{D}$ may be a clique on $\mathcal{D}$ for some neighborhood structure but it may not be a clique for another neighborhood structure.

All individual pixels and all horizontally and vertically adjacent pixel pairs form cliques both for the first and for the second order neighborhood structures. However, although diagonally adjacent pixel pairs form cliques for the second order neighborhood structure, they do not form cliques for the first order neighborhood structure.

Cliques defined on $(\mathcal{D},\eta)$ may be classified according to their shapes as they

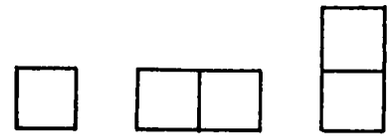
appear on the image region $\mathcal{D}$. Figure 2.2.a shows the set of all possible clique types for the two dimensional array of pixels associated with the first order neighborhood structure. In Figure 2.2.b, the set of all possible clique types for the second order neighborhood structure is shown.

The neighborhood structure defined on $\mathcal{D}$ should be updated at the boundary pixels of a finite rectangular image region. One way is to assume that the boundary pixels have smaller neighborhoods than the interior points. Another choice is to keep the shape of the neighborhood structure of the interior points the same also at the boundaries of the image region by assuming a periodic interaction which will be illustrated in the following paragraphs. In this latter case, the set $\mathcal{D}$ is called a *toroidal* image region. In this thesis, the second approach is adopted and the set $\mathcal{D}$ will be assumed to be toroidal.

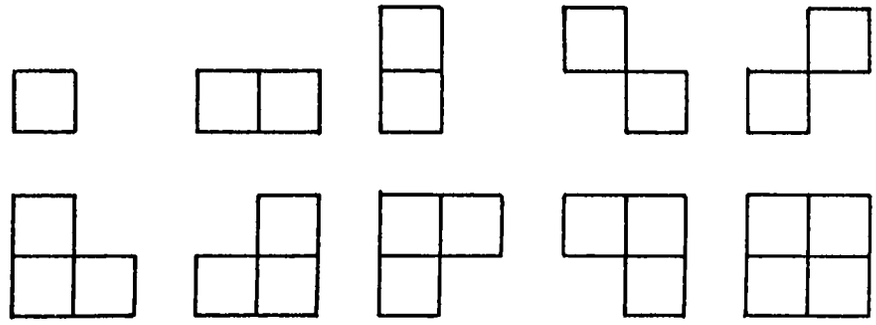
By assuming a toroidal image region, we got rid of some effects of the boundaries of the rectangular image region since now every pixel has adjacent pixels in all directions (horizontal, vertical, diagonal). The shape of the neighborhood structure will not change at the edges of the image region and it will have a continuation at some other edges. Figure 2.3.a illustrates the situation for a second order neighborhood of a pixel located at an edge of a rectangular image region. In Figure 2.3.b, a similar situation is illustrated for a corner pixel of the image region. In Figure 2.3, the shaded regions indicate the neighborhoods of the points i and j, respectively, and the black dots indicate the pixel locations.

In accordance with the neighborhood structure on toroidal arrays, the cliques at the boundaries of a rectangular array of pixels will be treated in a similar way. In Figure 2.4, the shaded area corresponds to a horizontal clique at the boundary of a rectangular array of pixels when the region is assumed to be toroidal.

Roughly speaking, a random field may be defined on a set $\mathcal{D}$ by assigning a random variable to each point of the set $\mathcal{D}$. Let $\tilde{y}(i)$ be a random variable assigned to a point $i \in \mathcal{D}$. The set of random variables defined by $\tilde{\mathcal{Y}} = \{\tilde{y}(i) : i \in \mathcal{D}\}$ is a random field on $\mathcal{D}$.

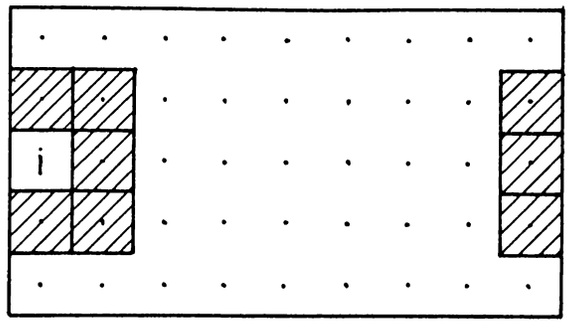


(a)

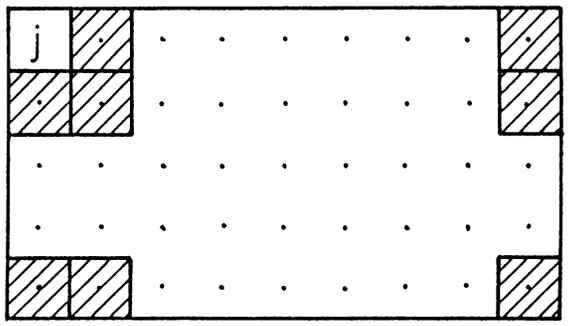


(b)

Figure 2.2. Clique types, (a) for the first order neighborhood structure, (b) for the second order neighborhood structure.



(a)



(b)

Figure 2.3. Appearance of the neighborhood, (a) for an edge pixel, (b) for a corner pixel.

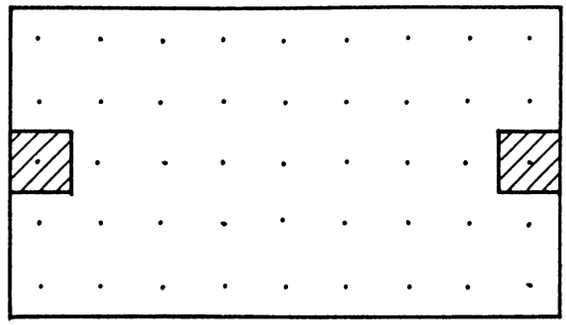


Figure 2.4. Appearance of a horizontal clique at the boundary of a toroidal array of pixels.

We will denote a realization of the random field $\tilde{\mathcal{Y}}$ by $\mathcal{Y}$ and the set of all realizations by Y. Also, $y(i)$ will denote a realization of $\tilde{y}(i)$.

The following notation will be used throughout this thesis. Let $\mathcal{R}$ be an arbitrary subset of $\mathcal{D}$. Then, $\tilde{\mathcal{Y}}(\mathcal{R})$ will denote the set of random variables assigned to the points in $\mathcal{R}$ and $\mathcal{Y}(\mathcal{R})$ will denote a numerical realization of $\tilde{\mathcal{Y}}(\mathcal{R})$. With this notation, we have $\tilde{\mathcal{Y}} = \tilde{\mathcal{Y}}(\mathcal{D})$ and $\mathcal{Y} = \mathcal{Y}(\mathcal{D})$. Also $P(.)$ will denote the probability and $P(.|.)$ will denote the conditional probability assignments.

2.3 Markov Random Fields

A definition of a Markov random field is given in [1]. In the definition given below, we omitted the *positivity* and *homogeneity* properties described in [1]. However, we will consider these properties as assumptions in this thesis.

Definition: A Markov random field (MRF) is a probability assignment on the elements $\mathcal{Y}$ of the set Y subject to the following condition (Markovianity

property):

$$P\left(y(i)|\mathcal{Y}(\mathcal{D}\setminus\{i\})\right) = P\left(y(i)|\mathcal{Y}(\eta_i)\right) \qquad \forall i \in \mathcal{D} \ . \tag{2.1}$$

Therefore, an MRF is characterized by a conditional probability distribution at each point of the set $\mathcal{D}$ conditioned on the numerical values of the corresponding neighborhood points. However, there are some limitations on the general form of the conditional probability distributions. These limitations are imposed in order to achieve consistent conditional probability distributions which are valid at each point of the set $\mathcal{D}$. A deeper analysis of MRFs may be found in [10],[12].

In the following paragraphs, the general formulation of valid conditional probability distributions will be described without proof. A more detailed discussion may be found in [13].

In this formulation, each numerical realization $\mathcal{Y}$ of $\tilde{\mathcal{Y}}$ is assumed to have a non-zero probability, that is

$$P(\mathcal{Y}) > 0 \qquad \forall \mathcal{Y} \in Y \ .$$

This is called the *positivity property.* As an example, consider the case of a binary MRF defined on an image region containing N pixels. Then, there may be a total number of 2^N realizations of the random field. Under positivity assumption, each of these 2^N realizations will have non-zero probabilities.

The positivity assumption enables us to define the following function

$$Q(\mathcal{Y}) = \ln\left(\frac{P(\mathcal{Y})}{P(\tilde{\mathcal{Y}} = \underline{0})}\right) \ . \tag{2.2}$$

In (2.2), $P(\tilde{\mathcal{Y}} = \underline{0})$ denotes the probability of a realization of the random field composed of all zeros. Here, it is assumed that the value 0 may be taken by any point of the set $\mathcal{D}$ with non-zero probability. Note that finding the general form of $Q(\mathcal{Y})$ means finding the general form of the conditional probability distributions. Also, define $\mathcal{Y}_i$ for any given $\mathcal{Y}$ as follows :

$$\mathcal{Y}_i = (y(1), ..., y(i-1), 0, y(i+1), ..., y(N)) \tag{2.3}$$

where N is the total number of pixels in $\mathcal{D}$.

Since we have

$$\exp\{Q(\mathcal{Y}) - Q(\mathcal{Y}_i)\} = \frac{P(\mathcal{Y})}{P(\mathcal{Y}_i)} = \frac{P(\tilde{y}(i) = y(i)|y(1), ..., y(i-1), y(i+1), ..., y(N))}{P(\tilde{y}(i) = 0|y(1), ..., y(i-1), y(i+1), ..., y(N))} , \tag{2.4}$$

the solution to this problem gives the most general form which may be taken by the conditional probability distribution at each point of $\mathcal{D}$. In [13], the most general form of $Q(\mathcal{Y})$ is given as

$$\begin{aligned} Q(\mathcal{Y}) &= \sum_{1 \le i \le N} y(i)\mathcal{G}_i(y(i)) + \sum_{1 \le i < j \le N} \sum y(i)y(j)\mathcal{G}_{i,j}(y(i), y(j)) \\ &+ \sum_{1 \le i < j < k \le N} \sum \sum y(i)y(j)y(k)\mathcal{G}_{i,j,k}(y(i), y(j), y(k)) + ... \\ &+ y(1)y(2)...y(N)\mathcal{G}_{1,2,...,N}(y(1), ..., y(N)) \quad . \end{aligned} \tag{2.5}$$

In the above formulation, for any $1 \le i < j < ... < s \le N$, the function $\mathcal{G}_{i,j,...,s}$ may be non-zero if and only if the points $i, j, ..., s$ form a clique. Subject to this restriction, the $\mathcal{G}$-functions may be chosen arbitrarily. Therefore, using (2.2) and (2.5), the most general form for the conditional probabilities may be found.

Before going onto the Gibbs random fields, the definition of an independent subset with some related concepts and properties will be introduced.

Definition: Consider a set $\mathcal{D}$ and a neighborhood structure η defined on it. Let $C \subset \mathcal{D}$ and $\bar{C} \subset \mathcal{D}$ be defined so that $C \bigcup \bar{C} = \mathcal{D}$ and $C \bigcap \bar{C} = \emptyset$. If for every $i \in C$ we have $\eta_i \subset \bar{C}$ then the set C will be called an *independent subset* of $\mathcal{D}$ for the assumed neighborhood structure η.

Note that a subset of $\mathcal{D}$ may be an independent subset for some neighborhood structure η and it may not be an independent subset for some other neighborhood structure. In this thesis, when an MRF is defined on the pair ($\mathcal{D}$,η), an independent subset of $\mathcal{D}$ will be assumed to be with respect to the neighborhood structure η used in defining the MRF.

For any MRF defined on a set $\mathcal{D}$ with a neighborhood structure η, the random variables assigned to the points of an independent subset C of $\mathcal{D}$ become statistically independent when conditioned on the numerical values assigned to the points of $\bar{C}$. Therefore we have

$$P(\mathcal{Y}(C)|\mathcal{Y}(\bar{C})) = \prod_{i \in C} P(y(i)|\mathcal{Y}(\bar{C})) \ . \tag{2.6}$$

Using the Markovianity property, (2.6) may be written as

$$P(\mathcal{Y}(C)|\mathcal{Y}(\bar{C})) = \prod_{i \in C} P(y(i)|\mathcal{Y}(\eta_i)) \ . \tag{2.7}$$

Let C_1 and C_2 be two independent subsets of $\mathcal{D}$. Then C_1 and C_2 will be called *disjoint independent subsets* of $\mathcal{D}$ if each of them are individually independent subsets of $\mathcal{D}$ and if they satisfy the condition $C_1 \bigcap C_2 = \emptyset$.

Furthermore any number of independent subsets $C_k, k = 1, ..., K$ of $\mathcal{D}$ will be called *disjoint independent subsets* of $\mathcal{D}$ if they are pairwise disjoint independent subsets of $\mathcal{D}$.

As an example, for the first order neighborhood structure, the set $\mathcal{D}$ may be partitioned into two disjoint independent subsets as shown in Figure 2.5.a. For the second order neighborhood case, the set $\mathcal{D}$ may be partitioned into four disjoint independent subsets as shown in Figure 2.5.b. In Figures 2.5.a and 2.5.b, the small squares indicate pixels and the numbers in them indicate the independent subsets to which the corresponding pixels belong. Such partitions of $\mathcal{D}$ are called as *codings* in [13].

2.4 Gibbs Random Fields

A closely related random field to the MRFs is the Gibbs random field. A Gibbs random field is defined on a set $\mathcal{D}$ as follows [3],[14]:

Definition: Let η be a neighborhood structure defined on the set $\mathcal{D}$. A random field $\tilde{\mathcal{Y}}$ defined on $\mathcal{D}$ has Gibbsian distribution (GD) or equivalently

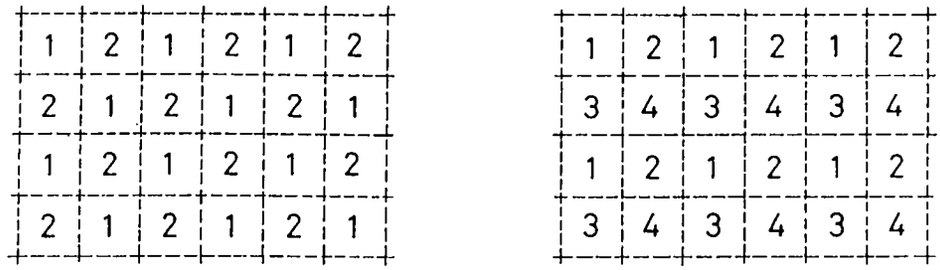


Figure 2.5. Codings, (a) for the first order neighborhood, and (b) for the second order neighborhood structures.

is a Gibbs random field (GRF) with respect to η if and only if its distribution is of the form

$$P(\mathcal{Y}) = \frac{1}{Z}\exp\{-\mathcal{U}(\mathcal{Y})\} \tag{2.8}$$

where

$$\mathcal{U}(\mathcal{Y}) = \sum_{c\in\mathcal{C}} V_c(\mathcal{Y}) \tag{2.9}$$

is the energy function and $V_c(\mathcal{Y})$ is the potential associated with clique c and Z is the partition function which can be expressed as

$$Z = \sum_{\mathcal{Y}\in\mathcal{Y}} \exp\{-\mathcal{U}(\mathcal{Y})\} \ . \tag{2.10}$$

The summation in (2.9) is over the set $\mathcal{C}$ of all cliques of $(\mathcal{D},\eta)$. The partition function Z is a normalizing constant to make the sum of the probabilities of all numerical realizations of the random field equal to one. The only condition imposed on the clique potentials $V_c(\mathcal{Y})$ is that they depend only on the values assigned to the pixels in the corresponding clique c.

The GD is basically an exponential distribution. However, by choosing the clique potentials $V_c(\mathcal{Y})$ properly, a wide variety of distributions can be formulated as GD. In contrast to MRF formulation, the GD naturally solves the consistency problems. Roughly speaking, the GRF formulation supplies to each outcome $\mathcal{Y}$ of the random field $\tilde{\mathcal{Y}}$ a probability assignment directly,

whereas the MRF formulation does this probability assignment in an indirect manner by imposing conditional probabilities to each point of $\mathcal{D}$.

2.5 MRF-GRF Equivalence Principle

In this section, an important and useful theorem on MRF-GRF equivalence will be given. A statement and a proof of the MRF-GRF equivalence principle may be found in [15].

Theorem: Under positivity assumption every MRF on a set $\mathcal{D}$ is a GRF on $\mathcal{D}$ and vice versa.

In this thesis, we limited our attention to those MRFs having the positivity property and we will consider the GRF equivalents of the MRFs where necessary keeping in mind the above theorem on MRF-GRF equivalence. Note that for a given binary Gibbsian distribution $P(\mathcal{Y})$ defined on the ensemble Y, the conditional probability distribution of the equivalent binary MRF may be found by

$$P(\tilde{y}(i) = s|\mathcal{Y}(\eta_i)) = P(\tilde{y}(i) = s|\mathcal{Y}(\mathcal{D} \setminus \{i\})) = \frac{P(\mathcal{Y}_{i,s})}{P(\mathcal{Y}_{i,0}) + P(\mathcal{Y}_{i,1})} \tag{2.11}$$

where $\mathcal{Y}_{i,s}$, $s \in \{0,1\}$, denotes the realization of the random field obtained by letting $\tilde{y}(i)$ equal to s and keeping the realization of $\tilde{\mathcal{Y}}(\mathcal{D} \setminus \{i\})$ constant.

2.6 Some MRF Probability Distributions

In this section, some specific MRF probability distributions will be described. These specific distributions are important not only because they illustrate the valid MRF distributions having the positivity property but also because they will be used in many parts of this thesis. More specifically, two models will be described which are equivalent to each other.

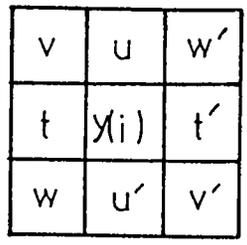


Figure 2.6. Numerical values of the neighborhood pixels.

2.6.1 Model 1

Consider a toroidal set of pixels $\mathcal{D}$. A binary MRF may be defined on $\mathcal{D}$ by the conditional probability distribution given below [1],

$$P(\tilde{y}(i) = s|\mathcal{Y}(\eta_i)) = \frac{\exp(sT)}{1+\exp(T)} \ , \quad s \in \{0,1\} \tag{2.12}$$

where T is a function of the numerical values of the pixels in η_i. For the first order neighborhood structure, T may be chosen as

$$T = \alpha + \beta_h(t+t') + \beta_v(u+u') \ . \tag{2.13}$$

For the case of second order neighborhood structure T may be chosen as

$$T = \alpha + \beta_h(t+t') + \beta_v(u+u') + \beta_m(v+v') + \beta_r(w+w') \tag{2.14}$$

where $t, t', u, u', v, v', w, w' \in \{0,1\}$ are the pixel values in a neighborhood of the point i as shown in Figure 2.6.

The parameters $\alpha, \beta_h, \beta_v, \beta_m, \beta_r$ appearing in (2.13) and (2.14) are called the *MRF parameters* or *MRF texture parameters* and roughly speaking they control:

α : the relative amount of 1's to 0's,

β_h : horizontal clustering of 1's,

β_v : vertical clustering of 1's,

β_m : clustering of 1's along the main diagonal,

β_r : clustering of 1's along the reverse diagonal.

Main diagonal of a rectangular image region is roughly defined as the diagonal from top left to bottom right corners of the image region $\mathcal{D}$. Similarly, reverse diagonal is roughly defined as the diagonal from top right to bottom left corners of the image region.

The conditional probability distribution in (2.12) may be obtained by choosing the $\mathcal{G}$-functions appearing in (2.5) as constant parameters for each clique type. For the first order neighborhood structure, the clique types are as shown in Figure 2.2.a. The form of T given by (2.13) is obtained by choosing the $\mathcal{G}$-functions as follows:

$$\mathcal{G}_i = \alpha \quad \text{for single pixel cliques,} \tag{2.15}$$

$$\mathcal{G}_{i,j} = \begin{cases} \beta_h & \text{if } i \text{ and } j \text{ are horizontally adjacent pixels,} \\ \beta_v & \text{if } i \text{ and } j \text{ are vertically adjacent pixels.} \end{cases} \tag{2.16}$$

For the second order neighborhood case, the clique types are as shown in Figure 2.2.b and the form of T given by (2.14) is obtained by choosing the $\mathcal{G}$-functions as follows:

$$\mathcal{G}_i = \alpha \quad \text{for single pixel cliques,} \tag{2.17}$$

$$\mathcal{G}_{i,j} = \begin{cases} \beta_h & \text{if } i \text{ and } j \text{ are horizontally adjacent pixels,} \\ \beta_v & \text{if } i \text{ and } j \text{ are vertically adjacent pixels,} \\ \beta_m & \text{if pixels } i \text{ and } j \text{ are adjacent along the main diagonal,} \\ \beta_r & \text{if pixels } i \text{ and } j \text{ are adjacent along the reverse diagonal,} \\ 0 & \text{for all other clique types for the second order neighborhood structure.} \end{cases} \tag{2.18}$$

Note that in (2.18), the $\mathcal{G}$-functions for clique types containing more than two pixels have been chosen to be zero. This is done for simplicity only. The conditional probability distribution in (2.12) will still be valid if non-zero parameters are assigned to the other possible clique types for the second order neighborhood structure.

For the second order neighborhood structure, the GRF equivalent of the conditional probability distribution given by (2.12) may be obtained by choosing the clique potentials $V_c(\mathcal{Y})$ appearing in (2.9) as:

$$V_c(\mathcal{Y}) = \begin{cases} -\alpha & \text{for single pixel cliques having value 1,} \\ -\beta_h & \text{for two horizontally adjacent pixels both having value 1,} \\ -\beta_v & \text{for two vertically adjacent pixels both having value 1,} \\ -\beta_m & \text{for two adjacent pixels along the main diagonal} \\ & \text{both having value 1,} \\ -\beta_r & \text{for two adjacent pixels along the reverse diagonal} \\ & \text{both having value 1,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.19}$$

For the first order neighborhood structure, the clique potentials may be obtained from (2.19) by letting β_m and β_r equal to zero.

In the MRF probability distribution described above, the MRF parameters are assigned to the clique types and they are independent from the position of the cliques in the image region $\mathcal{D}$. Therefore, the conditional probability distribution at any point in $\mathcal{D}$ is dependent on the numerical realizations of the neighboring points, but it is independent from the position of the point in the image region $\mathcal{D}$. This is called the *homogeneity* property as defined in [1]. In the GRF equivalent, this property corresponds to saying that a clique potential, $V_c(\mathcal{Y})$, is a function of the clique type and the pixel values in the clique but it is independent of the position of the clique in the image region $\mathcal{D}$. In this thesis, we will deal only with homogeneous MRFs.

2.6.2 Model 2

Again consider a toroidal set of pixels $\mathcal{D}$. A binary GRF may be defined on $\mathcal{D}$ by choosing the clique potentials as described below [3]. For the single pixel cliques, the clique potentials are chosen as

$$V_c(\mathcal{Y}) = \begin{cases} \alpha_0 & \text{if the pixel value in } c \text{ is 0,} \\ \alpha_1 & \text{if the pixel value in } c \text{ is 1.} \end{cases} \tag{2.20}$$

For cliques containing more than one pixel, a parameter is associated to each clique type. Then the clique potentials are chosen as:

$$V_c(\mathcal{Y}) = \begin{cases} -\psi_c & \text{if all pixel values in } c \text{ are equal,} \\ \psi_c & \text{otherwise.} \end{cases} \tag{2.21}$$

In (2.21), ψ_c denotes the parameter associated with the clique type to which the clique c belongs.

For the second order neighborhood structure and the assumption that only single and double pixel cliques may have non-zero parameters, the double pixel clique parameters may be chosen as:

$$\psi_c = \begin{cases} \beta'_h & \text{for horizontally adjacent pixel pairs,} \\ \beta'_v & \text{for vertically adjacent pixel pairs,} \\ \beta'_m & \text{for pixel pairs adjacent along the main diagonal,} \\ \beta'_r & \text{for pixel pairs adjacent along the reverse diagonal.} \end{cases} \tag{2.22}$$

From (2.11), the MRF equivalent of this type of a GRF probability distribution may be found as:

$$P(\tilde{y}(i) = s|\eta_i) = \begin{cases} \frac{\exp(T_0)}{\exp(T_0)+\exp(T_1)} & \text{for } s = 0 \\ \frac{\exp(T_1)}{\exp(T_0)+\exp(T_1)} & \text{for } s = 1 \end{cases} \tag{2.23}$$

where

$$\begin{aligned} T_0 &= \alpha_0 - 2\beta'_h(t + t' - 1) - 2\beta'_v(u + u' - 1) \\ &\quad -2\beta'_m(v + v' - 1) - 2\beta'_r(w + w' - 1) \ , \end{aligned} \tag{2.24}$$

and

$$\begin{aligned} T_1 &= \alpha_1 + 2\beta'_h(t + t' - 1) + 2\beta'_v(u + u' - 1) \\ &\quad +2\beta'_m(v + v' - 1) + 2\beta'_r(w + w' - 1) \ . \end{aligned} \tag{2.25}$$

In (2.24) and (2.25), the variables $t, t', u, u', v, v', w, w' \in \{0, 1\}$ are the values of the neighboring pixels of the point i as shown in Figure 2.6.

The conditional probability distribution given by (2.23) may be alternatively written as:

$$P(\tilde{y}(i) = s|\eta_i) = \begin{cases} \frac{1}{1+\exp(T')} & \text{for } s = 0 \\ \frac{\exp(T')}{1+\exp(T')} & \text{for } s = 1 \end{cases} \tag{2.27}$$

which is equivalent to

$$P(\tilde{y}(i) = s|\eta_i) = \frac{\exp(sT')}{1+\exp(T')} \qquad s \in \{0,1\} \tag{2.28}$$

where T' is given by

$$\begin{aligned} T' &= T_1 - T_0 \\ &= \alpha_1 - \alpha_0 \quad +4\beta'_h(t+t'-1) + 4\beta'_v(u+u'-1) \\ &\qquad\qquad +4\beta'_m(v+v') + 4\beta'_r(w+w') \ . \end{aligned} \tag{2.29}$$

In (2.28), the effect of the parameters α_0 and α_1 is equivalent to a single parameter α' defined as

$$\alpha' = \alpha_1 - \alpha_0 \tag{2.30}$$

With this definition of α', the single pixel clique potential may be written as:

$$V_c(\mathcal{Y}) = \begin{cases} 0 & \text{if the pixel value in } c \text{ is 0,} \\ \alpha' & \text{if the pixel value in } c \text{ is 1.} \end{cases} \tag{2.31}$$

Roughly speaking, the GRF parameters for this GRF distribution control:

α': the relative amount of 1's to 0's in the image,

β'_h: horizontal clustering of equal valued pixels,

β'_v: vertical clustering of equal valued pixels,

β'_m: clustering of equal valued pixels along the main diagonal,

β'_r: clustering of equal valued pixels along the reverse diagonal.

Now, we will show that Models 1 and 2 described above are equivalent MRF (GRF) distributions for binary random fields. Observing that (2.12) and (2.27) have similar forms and equating (2.14) and (2.28) we have

$$\alpha \ = \ \alpha' - 4\beta_h - 4\beta_v - 4\beta_m - 4\beta_r$$

$$\begin{aligned} \beta_h &= 4\beta'_h \\ \beta_v &= 4\beta'_v \\ \beta_m &= 4\beta'_m \\ \beta_r &= 4\beta'_r \end{aligned} \tag{2.31}$$

where α' is defined by (2.29).

Therefore, for the second order neighborhood structure, a binary MRF described by the conditional probability distribution given by (2.23) and parameters $\alpha', \beta'_h, \beta'_v, \beta'_m, \beta'_r$ is equivalent to a binary MRF described by the conditional probability distribution given by (2.12) and the parameters $\alpha, \beta_h, \beta_v, \beta_m, \beta_r$, if the parameters are chosen to satisfy (2.31).

Chapter 3

Generation of MRF Textures

3.1 Introduction

The aim of this chapter is to introduce some MRF texture generation algorithms. Such algorithms are also used in statistical mechanics for the simulations of some physical systems [16]. In this chapter two algorithms are described. The first one is a common sequential algorithm similar to the one given in [1]. The second one is quite suitable for parallel implementation and we designed it to develop a parallel network for MRF texture generation. The theory behind these algorithms is based on Markov chains and in the literature there exist a variety of such algorithms [1],[2],[14]. A mathematical analysis and a related proof of the second algorithm are also presented in this chapter. The chapter is continued with some computer simulation results both for the sequential and for the parallel algorithms.

3.2 A Sequential Algorithm for Sample MRF Texture Generation

The methods for the generation of sample MRF textures are generally based on finite state Markov chains. These methods consider all numerical realizations

of the random field as states of a Markov chain. The transition probabilities are chosen such that the Markov chain possesses a unique steady state probability distribution and in steady state this probability distribution is the equivalent Gibbsian distribution of the MRF under consideration. The following theorem is useful in order to set up such a Markov chain [1].

Theorem: Consider a finite state, symmetric, aperiodic, irreducible Markov chain with one step transition matrix P^* and with a total number of M states. Let $\underline{\pi} = \{\pi_k : k = 1, ..., M, \ \pi_k \in R^+, \ \sum_{k=1}^{M} \pi_k = 1\}$ be a set of positive numbers which sum up to one. Then the Markov chain with one step transition matrix P has limiting distribution $\underline{\pi}$, where P is defined by

$$p_{kl} = \begin{cases} p^*_{kl}\pi_l/\pi_k & \text{if } \pi_k > \pi_l \\ p^*_{kl} & \text{if } \pi_k \leq \pi_l \end{cases} \qquad \text{for} \quad k \neq l \ , \tag{3.1}$$

$$p_{kk} = 1 - \sum_{l, l \neq k} p_{kl} \ . \tag{3.2}$$

In (3.1) and (3.2), the one step transition probabilities p_{kl} and p^*_{kl} are the (k, l)'th entries of P and P^*, respectively, and k and l are the indices of any two states of the Markov chains.

To generate samples from an MRF in steady state, the steady state probability distribution $\underline{\pi}$ is chosen as the equivalent Gibbsian distribution of the MRF under consideration. In order to determine the one step transition probabilities, all we need to do is to determine the ratio of probabilities π_l/π_k. The following theorem may be used to determine the ratio of probabilities $P(\mathcal{Y}_1)$ and $P(\mathcal{Y}_2)$ for the Gibbsian distribution [1],[13].

Theorem: Let $\mathcal{Y}_1$ and $\mathcal{Y}_2$ be any two numerical realizations of an MRF. Then

$$\frac{P(\mathcal{Y}_2)}{P(\mathcal{Y}_1)} = \prod_{i=1}^{N} \frac{P(\tilde{y}_2(i) = y_2(i) | y_1(1), ..., y_1(i-1), y_2(i+1), ..., y_2(N))}{P(\tilde{y}_1(i) = y_1(i) | y_1(1), ..., y_1(i-1), y_2(i+1), ..., y_2(N)} \tag{3.3}$$

where N is the total number of points in $\mathcal{D}$.

A sequential algorithm that can be used to generate binary MRF textures is as follows:

- **Step 1**: Initially, assign each pixel of the image region $\mathcal{D}$ an arbitrary value taken from the set $\{0, 1\}$.

- **Step 2**: Randomly choose a pixel i of $\mathcal{D}$. This random choice must be such that each pixel of $\mathcal{D}$ has a fixed non-zero probability to be chosen. Let $\mathcal{Y}_p$ denote the present numerical realization of $\mathcal{D}$ and let $\mathcal{Y}_n$ denote the numerical realization of $\mathcal{D}$ which is obtained from $\mathcal{Y}_p$ just by changing the value of the chosen pixel. In other words, if $y_p(i) = 1$ then $y_n(i) = 0$ and if $y_p(i) = 0$ then $y_n(i) = 1$. Let

$$r = \frac{P(\mathcal{Y}_n)}{P(\mathcal{Y}_p)} \ . \tag{3.4}$$

  If $r \geq 1$ then strictly pass to $\mathcal{Y}_n$, if $r < 1$ then pass to $\mathcal{Y}_n$ with probability r.

- **Step 3**: Go to Step 2.

The sequential algorithm described above defines a Markov chain such that it has a unique steady state probability distribution and this distribution is Gibbsian. The algorithm never stops, however for practical purposes it may be stopped after a finite number of cycles between Steps 2 and 3 as stability is achieved. Here, the criteria for stability must be defined. As an example, stability may be defined as the condition that the estimated parameters of the generated sample textures differ from the specified parameters at most by some predefined error. Some other definitions for stability are also possible.

The number r in (3.4) may be calculated using (3.3). For the MRF probability distribution described by (2.12) where T may be given by (2.13) or (2.14) depending on the choice of the neighborhood structure, the number r is given by

$$r = \begin{cases} \exp(T) & \text{if } y_p(i) = 0 \ , \\ \exp(-T) & \text{if } y_p(i) = 1 \ . \end{cases} \tag{3.5}$$

In (3.5), i denotes the chosen pixel at the second step of the above algorithm.

3.3 A Parallel Algorithm for Sample MRF Texture Generation

In this section, another algorithm for binary MRF texture generation is described. Although the algorithm can be implemented in a highly parallel manner, it is also possible to implement it in a less parallel or completely sequential manner as well. The algorithm is used for the development of a parallel network for the generation of sample MRF textures as described in Chapter 5. In this section, we will again use the notational conventions introduced in Chapter 2. Also, the subscripts p and n will stand for the words *present* and *next*, respectively.

3.3.1 The Algorithm

For a given MRF probability distribution and an associated neighborhood structure η, the algorithm requires the set $\mathcal{D}$ to be partitioned into disjoint independent subsets as defined in Chapter 2. Let C_k, $k = 1, ..., K$ be disjoint independent subsets of $\mathcal{D}$ such that

$$\bigcup_{k=1}^{K} C_k = \mathcal{D} \ .$$

Then, the algorithm may be described as follows:

- **Step1:** Initially assign each pixel of the image region $\mathcal{D}$ an arbitrary value taken from the set $\{0, 1\}$.
- **Step2:** Randomly choose an independent subset C_k of $\mathcal{D}$ with a fixed non-zero probability P_k. So, we will have

$$P_k = P(C_k) \tag{3.6}$$

where

$$\sum_{k=1}^{K} P_k = 1 \quad \text{and} \quad P_k > 0 \quad \forall k \in \{1, .., K\} \ . \tag{3.7}$$

For each point $j \in C_k$, determine the probability that the point j takes on the value 1. Denote this probability by p_j. So we have

$$p_j = P(\tilde{y}_p(j) = 1 | \mathcal{Y}_p(\eta_j)) \ . \tag{3.8}$$

Also let

$$q_j = 1 - p_j = P(\tilde{y}_p(j) = 0 | \mathcal{Y}_p(\eta_j)) \ . \tag{3.9}$$

Then, let $y_n(j) = 1$ with probability p_j or equivalently, let $y_n(j) = 0$ with probability q_j. Repeat this updating procedure for all $j \in C_k$.

- **Step3:** Go to Step 2.

In this algorithm, all pixels of a chosen independent subset are updated at the same time. Therefore, to achieve the highly parallel potential of the algorithm it is desirable to keep the number of pixels in each independent subset as large as possible while keeping the number of the disjoint independent subsets of $\mathcal{D}$ as small as possible. As an example, for the first order neighborhood case, the disjoint independent subsets may be chosen as in Figure 2.5.a and for the second order neighborhood case they may be chosen as in Figure 2.5.b.

The above algorithm never stops because after being initialized in Step 1, it goes back and forth between Steps 2 and 3. However, for some practical purposes, it may be stopped after a finite number of cycles between Steps 2 and 3 when the sample textures generated by the algorithm become stable. Here, the criteria for stability may be defined as in the case of the sequential algorithm.

For the MRF model described in Chapter 2 as Model 1, the probabilities in (3.8) and (3.9) may be determined by letting s equal to 1 and 0, respectively, in the conditional probability distribution given by (2.12).

3.3.2 Mathematical Analysis of the Parallel Algorithm as a Markov Chain

A mathematical analysis of the parallel algorithm is as follows. Assume that all numerical realizations of the random field $\tilde{\mathcal{Y}}$ compose the states of an arbitrary Markov chain. Since the image region $\mathcal{D}$ is assumed to contain a finite number of pixels and since the random variables assigned to these pixels are binary, the number of states of the Markov chain will be finite. We will derive the one step transition probabilities of the Markov chain defined by the parallel algorithm. Also, we will show that this Markov chain is aperiodic and irreducible. Some basic definitions on Markov chains may be found in [17, Chapter 15].

Step 2 of the parallel algorithm defines the transition rule from one state (numerical realization of $\tilde{\mathcal{Y}}$) of the Markov chain to another one. So, this step defines the set of states directly accessible from the present state and it also determines the one step transition probabilities.

Let $\mathcal{Y}_p$ be the present state of the Markov chain. Then a next state, $\mathcal{Y}_n$, directly accessible from $\mathcal{Y}_p$ can differ from $\mathcal{Y}_p$ in at most one of the disjoint independent subsets of $\mathcal{D}$. That is, if i and j are any two points of $\mathcal{D}$ such that $i \in C_l$ and $j \in C_k$ where $l \neq k$ and if $y_p(i) \neq y_n(i)$ then we must have that $y_p(j) = y_n(j)$.

Let $G_k(\mathcal{Y}_p)$, $k = 1, ..., K$, denote the set of possible next states directly accessible from $\mathcal{Y}_p$ such that if $\mathcal{Y}_n \in G_k(\mathcal{Y}_p)$ then $\mathcal{Y}_n$ may differ from $\mathcal{Y}_p$ only in the k'th independent subset of $\mathcal{D}$. Note that $\mathcal{Y}_p$, itself, is an element of every $G_k(\mathcal{Y}_p), k = 1, ..., K$. Also let $G'_k(\mathcal{Y}_p) = G_k(\mathcal{Y}_p) \setminus \{\mathcal{Y}_p\}$. Therefore, for any arbitrary $\mathcal{Y}_p$, the sets $G'_k(\mathcal{Y}_p)$, $k = 1, ..., K$, are disjoint sets and the only common element in the sets $G_k(\mathcal{Y}_p)$, $k = 1, ..., K$, is the state $\mathcal{Y}_p$ itself.

Note that if a state $\mathcal{Y}_n$ is directly accessible from $\mathcal{Y}_p$ then $\mathcal{Y}_p$ is also directly accessible from $\mathcal{Y}_n$. Furthermore, if $\mathcal{Y}_n \in G_k(\mathcal{Y}_p)$ then $\mathcal{Y}_p \in G_k(\mathcal{Y}_n)$. This is obvious since if $\mathcal{Y}_n$ differs from $\mathcal{Y}_p$ only in k'th independent subset C_k then $\mathcal{Y}_p$ will also differ from $\mathcal{Y}_n$ only in k'th independent subset. Picture of the Markov

chain around an arbitrary present state $\mathcal{Y}_p$ is as shown in Figure 3.1.

The one step transition probabilities from an arbitrary state $\mathcal{Y}_p$ to the states that are directly accessible from $\mathcal{Y}_p$ are determined as follows. Let us denote the one step transition probability from state $\mathcal{Y}_p$ to $\mathcal{Y}_n$ by p_{pn}, that is,

$$p_{pn} = P(\tilde{\mathcal{Y}}_n = \mathcal{Y}_n | \tilde{\mathcal{Y}}_p = \mathcal{Y}_p) \ . \tag{3.10}$$

In (3.10), $\tilde{\mathcal{Y}}_p$ and $\tilde{\mathcal{Y}}_n$ denote the present and next states of the Markov chain. Then, we have

$$p_{pn} = \sum_{l=1}^{K} P(\mathcal{Y}_n | \mathcal{Y}_p, C_l) P(C_l | \mathcal{Y}_p) \ . \tag{3.11}$$

In (3.11), $P(C_l|\mathcal{Y}_p)$ denotes the probability of choosing the l'th independent subset C_l given that the present state is $\mathcal{Y}_p$ and $P(\mathcal{Y}_n|\mathcal{Y}_p, C_l)$ denotes the probability of passing to the state $\mathcal{Y}_n$ in one step if the present state is $\mathcal{Y}_p$ and the chosen independent subset is C_l.

Note that the probability of choosing C_l is independent from the present state, so we have

$$P(C_l|\mathcal{Y}_p) = P(C_l) = P_l \quad \forall l \in \{1, ..., K\} \ . \tag{3.12}$$

The probability $P(\mathcal{Y}_n|\mathcal{Y}_p, C_l)$ appearing in (3.11) is strictly zero if $\mathcal{Y}_n$ differs from $\mathcal{Y}_p$ in more than one of the independent subsets $C_l, l = 1, ..., K$. Assuming that $\mathcal{Y}_n \in G'_k(\mathcal{Y}_p)$, the probability $P(\mathcal{Y}_n|\mathcal{Y}_p, C_l)$ in (3.11) will be non-zero only for $l = k$. Therefore, (3.11) reduces to

$$p_{pn} = P_k P(\mathcal{Y}_n | \mathcal{Y}_p, C_k) \qquad \mathcal{Y}_n \in G'_k(\mathcal{Y}_p) \ . \tag{3.13}$$

If $\mathcal{Y}_n = \mathcal{Y}_p$ then the probability $P(\mathcal{Y}_n|\mathcal{Y}_p, C_l)$ appearing in (3.11) will be non-zero for all independent subsets, C_l, $l = 1, ..., K$,and the self loop probability will be

$$p_{pp} = \sum_{l=1}^{K} P_l P(\mathcal{Y}_n = \mathcal{Y}_p | \mathcal{Y}_p, C_l) \ . \tag{3.14}$$

Therefore, for the chosen independent subset C_k and the given present state $\mathcal{Y}_p$, we must determine the probability that the next state is $\mathcal{Y}_n$. Step 2 of the parallel algorithm suggests that $\mathcal{Y}_n$ is randomly chosen from the set $G_k(\mathcal{Y}_p)$.

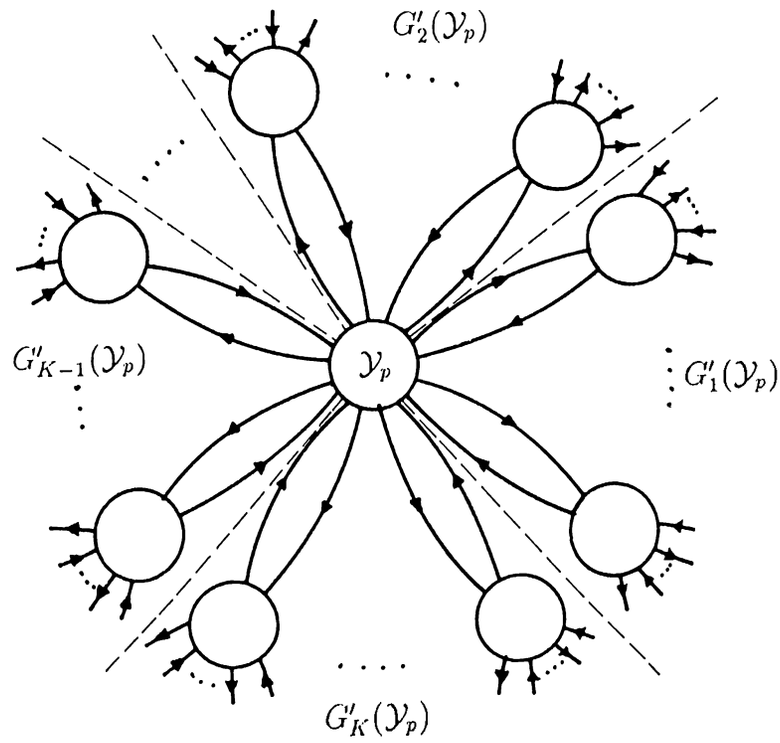


Figure 3.1. Structure of the Markov chain around an arbitrary present state $\mathcal{Y}_p$.

The random variables assigned to the pixels in C_k are statistically independent when conditioned on the fixed numerical realization of $\bar{C}_k$ which is common to $\mathcal{Y}_p$ and to all $\mathcal{Y}_n$'s where $\mathcal{Y}_n \in G_k(\mathcal{Y}_p)$, and they are updated independently from each other. Therefore, the probability of choosing a $\mathcal{Y}_n$ from the set $G_k(\mathcal{Y}_p)$ will be given by

$$P(\mathcal{Y}_n|\mathcal{Y}_p, C_k) = \prod_{i \in C_k, y_n(i)=1} p_i \prod_{j \in C_k, y_n(j)=0} q_j \tag{3.15}$$

where p_i and q_j are as defined by (3.8) and (3.9).

Using (3.13) and (3.15) we obtain the transition probability p_{pn} for any $\mathcal{Y}_n \in G'_k(\mathcal{Y}_p)$, $k = 1, ..., K$ as

$$p_{pn} = P_k \prod_{i \in C_k, y_n(i)=1} p_i \prod_{j \in C_k, y_n(j)=0} q_j \ . \tag{3.16}$$

For the case $\mathcal{Y}_n = \mathcal{Y}_p$ we have

$$p_{pp} = \sum_{l=1}^{K} \left\{ P_l \prod_{i \in C_l, y_p(i)=1} p_i \prod_{j \in C_l, y_p(j)=0} q_j \right\} \ . \tag{3.17}$$

Note that the one step transition probabilities p_{pn} have the property that,

$$0 < p_{pn} < 1 \qquad \forall \mathcal{Y}_n \in \bigcup_{l=1}^{K} G_l(\mathcal{Y}_p) \ . \tag{3.18}$$

In (3.18), p_{pn} can not be zero because the conditional probabilities p_i and q_j appearing in (3.16) and (3.17) are non-zero due to the positivity property of the MRF models under consideration and furthermore, the probability of choosing any independent subset of $\mathcal{D}$ is non-zero as required by the algorithm. In (3.18), p_{pn} is non-one because there are more than one possible next states for any present state $\mathcal{Y}_p$ and the one step transition probabilities, p_{pn}, should sum up to one over these next states. Therefore, if any p_{pn} had value one, this would require the other one step transition probabilities from the same present state to have the value zero which conflicts with the first inequality in (3.18). Also, for any state $\mathcal{Y}_s \notin \bigcup_{l=1}^{K} G_l(\mathcal{Y}_p)$ we have $p_{ps} = 0$.

Now, we will show that the Markov chain under consideration is aperiodic and irreducible. For any given two states in the Markov chain, there is a

possible transition from one of them to the other in at most K steps with non-zero probability. This is because any state can be obtained from any other state at most by modifying the pixel values in K of the disjoint independent subsets $C_k, k = 1, ..., K$, of $\mathcal{D}$ and any modification of a chosen independent subset has non-zero probability as implied by (3.18). Therefore the Markov chain is irreducible.

The Markov chain is aperiodic since for any given present state, the next state is a choice from a set containing at least two states which are the present state itself and some other state that can be obtained by modifying an independent subset of the image region.

3.3.3 Steady State Probability Distribution of the Markov Chain

Since the finite state Markov chain described above is aperiodic and irreducible, it has a unique steady state probability distribution independent from the initial state and this steady state distribution satisfies the balance equation given by

$$\sum_{\mathcal{Y}_n \in \mathcal{A}(\mathcal{Y}_p)} \pi_p p_{pn} = \sum_{\mathcal{Y}_n \in \mathcal{A}(\mathcal{Y}_p)} \pi_n p_{np} \tag{3.19}$$

for all states $\mathcal{Y}_p \in Y$. In (3.19), $\mathcal{A}(\mathcal{Y}_p)$ denotes the set of all states which are directly accessible from $\mathcal{Y}_p$ excluding $\mathcal{Y}_p$ itself, that is,

$$\mathcal{A}(\mathcal{Y}_p) = \bigcup_{l=1}^{K} G_l'(\mathcal{Y}_p) \tag{3.20}$$

and π_p and π_n are the steady state probabilities of an arbitrary present state $\mathcal{Y}_p$ and any state $\mathcal{Y}_n$ directly accessible from it, respectively. Furthermore any probability assignment which satisfies the balance equation given by (3.19) is the unique steady state probability distribution of the Markov chain. We will prove that the steady state probability distribution is Gibbsian by showing that the Gibbsian distribution satisfies the balance equation given by (3.19).

Note that the conditional probabilities p_i and q_j in (3.16) are determined

by an MRF conditional probability distribution. For example, for the MRF Model 1 described in Chapter 2, they are given by (2.12) by letting s equal to 1 and 0 respectively. Let $P_g(\mathcal{Y})$ denote the probability assignment on the elements of the ensemble Y determined by the Gibbsian distribution which is equivalent to the MRF conditional probability distribution used in the algorithm. Note that the ensemble Y is also the state space of the Markov chain implied by the MRF texture generation algorithm. With the Gibbsian probability distribution we have

$$P(\mathcal{Y}_n(\bar{C}_k)) = \sum_{\mathcal{Y}(C_k)} P_g(\mathcal{Y}_n) \ . \tag{3.21}$$

In the above equation, the joint probability distribution of the random variables assigned to the points in $\bar{C}_k$ is obtained by summing the joint probability distribution of all random variables assigned to the points in the set $\mathcal{D}$ over all possible numerical realizations of the portion of the random field $\tilde{\mathcal{Y}}(C_k)$. Using (3.21), we may write

$$\begin{aligned} \frac{P_g(\mathcal{Y}_n)}{\sum_{\mathcal{Y}(C_k)} P_g(\mathcal{Y}_n)} &= \frac{P(\mathcal{Y}_n(C_k \bigcup \bar{C}_k))}{P(\mathcal{Y}_n(\bar{C}_k))} \\ &= P(\mathcal{Y}_n(C_k)|\mathcal{Y}_n(\bar{C}_k)) \ . \end{aligned} \tag{3.22}$$

Since for any $i \in C_k$ we have $\eta_i \subset \bar{C}_k$, Markovianity property implies that the random variables $\tilde{y}(i), i \in C_k$ are statistically independent when conditioned on the numerical realization of $\tilde{\mathcal{Y}}(\bar{C}_k)$. Therefore we have

$$\begin{aligned} P(\mathcal{Y}_n(C_k)|\mathcal{Y}_n(\bar{C}_k)) &= \prod_{i \in C_k} P(y_n(i)|\mathcal{Y}_n(\bar{C}_k)) \\ &= \prod_{i \in C_k} P(y_n(i)|\mathcal{Y}_n(\eta_i)) \ . \end{aligned} \tag{3.23}$$

Note that the conditional probabilities appearing in (3.23) are derived from the Gibbsian equivalent of the MRF conditional probabilities used in the algorithm. Therefore, using the definitions of p_i and q_j given by (3.8) and (3.9) and noting that for any $\mathcal{Y}_n \in G_k(\mathcal{Y}_p)$ we have $\mathcal{Y}_n(\bar{C}_k) = \mathcal{Y}_p(\bar{C}_k)$, we may write

$$\prod_{i \in C_k} P(y_n(i)|\mathcal{Y}_n(\eta_i)) = \prod_{i \in C_k, y_n(i)=1} p_i \prod_{i \in C_k, y_n(i)=0} q_j \ . \tag{3.24}$$

Then, from (3.22), (3.23) and (3.24) we obtain

$$\prod_{i\in C_k, y_n(i)=1} p_i \prod_{i\in C_k, y_n(i)=0} q_j = \frac{P_g(\mathcal{Y}_n)}{\sum_{\mathcal{Y}(C_k)} P_g(\mathcal{Y}_n)} \quad . \tag{3.25}$$

Combining (3.16),(3.21) and (3.25), we may write the one step transition probabilities as

$$p_{pn} = \frac{P_k P_g(\mathcal{Y}_n)}{P(\mathcal{Y}_n(\bar{C}_k))} \qquad \forall \mathcal{Y}_n \in G'_k(\mathcal{Y}_p) \quad . \tag{3.26}$$

Similarly, since for any $\mathcal{Y}_n \in G_k(\mathcal{Y}_p)$ we have $\mathcal{Y}_p \in G_k(\mathcal{Y}_n)$, p_{np} may be written as

$$p_{np} = \frac{P_k P_g(\mathcal{Y}_p)}{P(\mathcal{Y}_p(\bar{C}_k))} \qquad \forall \mathcal{Y}_n \in G'_k(\mathcal{Y}_p) \quad . \tag{3.27}$$

Since for any $\mathcal{Y}_n \in G_k(\mathcal{Y}_p)$ we have $\mathcal{Y}_n(\bar{C}_k) = \mathcal{Y}_p(\bar{C}_k)$, we may write

$$P(\mathcal{Y}_n(\bar{C}_k)) = P(\mathcal{Y}_p(\bar{C}_k)) \quad . \tag{3.28}$$

Using (3.26), (3.27) and (3.28) we may write the below equality

$$P_g(\mathcal{Y}_p) p_{pn} = P_g(\mathcal{Y}_n) p_{np} \quad . \tag{3.29}$$

Summing both sides of the equation (3.29) over all $\mathcal{Y}_n \in \mathcal{A}(\mathcal{Y}_p)$ we have

$$\sum_{\mathcal{Y}_n \in \mathcal{A}(\mathcal{Y}_p)} P_g(\mathcal{Y}_p) p_{pn} = \sum_{\mathcal{Y}_n \in \mathcal{A}(\mathcal{Y}_p)} P_g(\mathcal{Y}_n) p_{np} \quad . \tag{3.30}$$

Note that (3.19) and (3.30) have similar forms and (3.30) is valid for all $\mathcal{Y}_p \in Y$. Therefore, if we let $\pi_p = P_g(\mathcal{Y}_p)$ and $\pi_n = P_g(\mathcal{Y}_n)$, then the balance equation in (3.19) will be satisfied. Since any probability distribution which satisfies the balance equation in (3.19) will be the unique steady state distribution of an aperiodic, irreducible Markov chain, we conclude that the Gibbsian distribution is the steady state probability distribution of the Markov chain.

Since the Markov chain is aperiodic and irreducible, its steady state distribution is independent from the initial state. As an arbitrary starting point, in Step 1 of the parallel algorithm, the Markov chain is initialized to a random state by setting the pixel values of $\mathcal{D}$ to arbitrary values taken from the set $\{0, 1\}$.

As stated before, this algorithm never stops. As time proceeds the probability of being in state $\mathcal{Y}_p$ converges to the probability assigned to it by the Gibbsian distribution which is equivalent to the MRF distribution. However, for some practical purposes, the algorithm may be stopped after a finite number of iterations.

This algorithm generates samples from an ensemble containing all possible numerical realizations of $\tilde{\mathcal{Y}}$ and whose probability distribution is determined by the MRF (GRF) formulation. At this step, we use the MRF-GRF equivalence to see that the textures generated at steady state has Markovianity property.

3.4 Experimental Results

Several experiments have been performed both for the sequential and for the parallel algorithms described in this chapter. Some of these experimental results are presented in Figures 3.2 to 3.19. In each figure, the texture shown in part (a) is generated by the sequential algorithm and the one shown in part (b) is generated by the parallel algorithm.

The MRF texture model used in generating these textures is the Model 1 described in Chapter 2 and defined on a 128*128 toroidal array of pixels. In this model, the conditional probability distribution is given by (2.12) and the form of T is given by (2.14). The random variables assigned to the pixels of the image are binary and in Figures 3.2 to 3.19, the pixel value 0 is represented by white and the pixel value 1 is represented by black.

The textures shown in Figures 3.2 to 3.19 have been generated by 100 iterations. For the sequential algorithm, one iteration corresponds to a number of cycles between Steps 2 and 3 of the algorithm which is equal to the total number of pixels in the image region $\mathcal{D}$, whereas, for the parallel algorithm, one iteration corresponds to a number of cycles between Steps 2 and 3 that is equal to the number of independent subsets of the image region under consideration. It has been observed that, both for the sequential and for the

parallel algorithms, stability is generally achieved in less than 20 iterations. Here, we define stability as the condition that the estimated parameters of the generated sample textures differ from the initially specified parameters at most by some predefined ratio of error. However, since the accuracy of parameter estimation may depend on the size of the textured image, the above definition for stability may not be a good one especially for small image regions.

One of the most important problems in implementing the algorithms is the generation of pseudo-random numbers. This is important for the reliability of the observations made on these sample textures. During our experimentation, we used three methods of pseudo-random number generation. First, we used the system supplied *rand()* function on Microsoft C compiler version 4.0 [23, p. 323]. As a second method, we used an improved pseudo-random number generator which uses the system supplied *rand()* function on the Microsoft C compiler version 4.0. This improved method is described in [20, pp. 207–208]. As a third method, we used the system supplied *drand48()* function available on the C compiler on Unix operating system directly [24, pp. 836–837]. The sample textures generated by using the random number generators described above by at least 20 iterations generally came out to be visually similar for the same sets of MRF texture parameters. The textured images shown in Figures 3.2 to 3.19 are generated by using the *drand48()* function. The programs listed in Appendix A may be used for the generation of sample MRF textures. Both programs are written in C programming language.

The programs listed in Appendix A use the system supplied *drand48()* function directly, however, it may be replaced by other pseudo-random number generators if desired. Some other pseudo-random number generators are described in [24] and some methods for random number generation may be found in [19, pp. 2–4], [20]. The first program listed in Appendix A implements the sequential algorithm and the second program implements the parallel algorithm which are described in this chapter.

Figures 3.2 to 3.19 show how the resulting MRF textures are affected by the MRF texture parameters. In Figure 3.2, by choosing the horizontal clustering

parameter β_h sufficiently larger than the other parameters, horizontal clustering is forced.

Similarly, in Figures 3.3 to 3.7 one of the β-parameters have been chosen sufficiently larger than the others so that a clustering in the corresponding direction is obtained.

In Figure 3.8, horizontal and vertical clustering parameters β_h and β_v are chosen positive and the diagonal clustering parameters β_m and β_r are chosen negative so that in the resulting texture horizontal and vertical clusterings are forced and clusterings along the diagonals are suppressed.

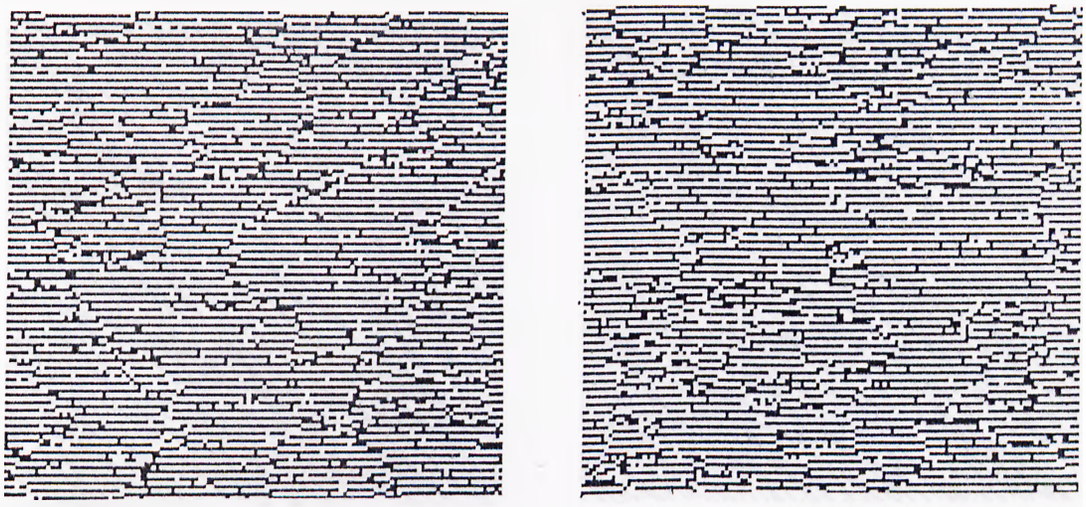
In Figures 3.9 to 3.11 the parameters are multiples of those of the textured image shown in Figure 3.8 so that the clusterings are emphasized.

In Figures 3.12 and 3.13, the horizontal and vertical clustering parameters, β_h and β_v, are chosen to be negative and the diagonal clustering parameters, β_m and β_r are chosen to be positive so that checkerboard looking textures are obtained.

In Figures 3.14 to 3.16, all β-parameters, except for the β_v parameter, are chosen to be equal and positive so that in the resulting texture there are both clusterings of black and white points into regions and also checkerboard looking regions. The MRF parameters are gradually increased from Figure 3.14 to 3.16 so that clustering effects are more emphasized in Figure 3.16.

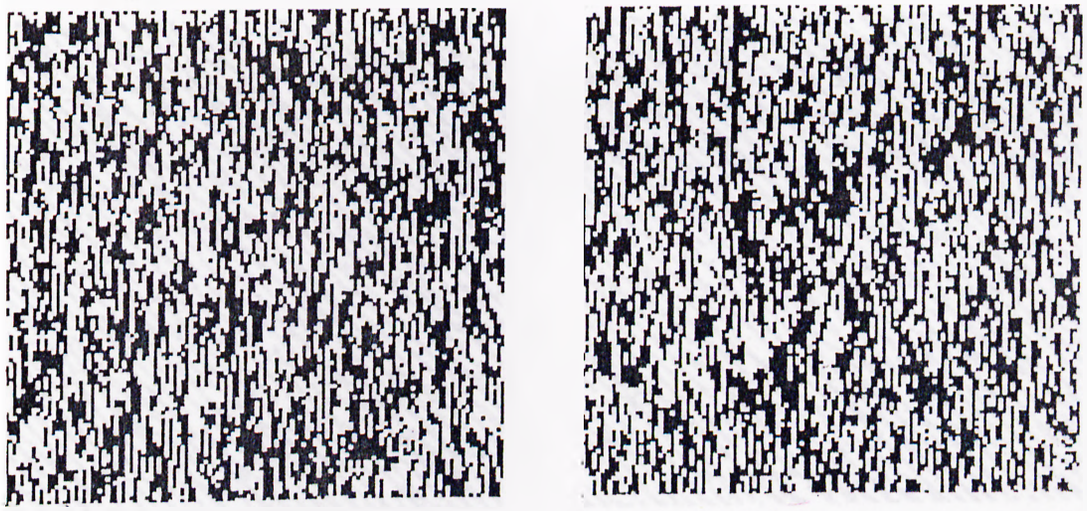
In Figure 3.17 all β-parameters are equal and positive so that the black and white points tend to cluster to form regions. The MRF parameters of the textures in Figures 3.18 and 3.19 are multiples of those in Figure 3.17 so that the clustering effects are more emphasized.

In all textures appearing in Figures 3.2 to 3.19, the α parameters are chosen to adjust the relative amounts of black and white pixels in the textures.



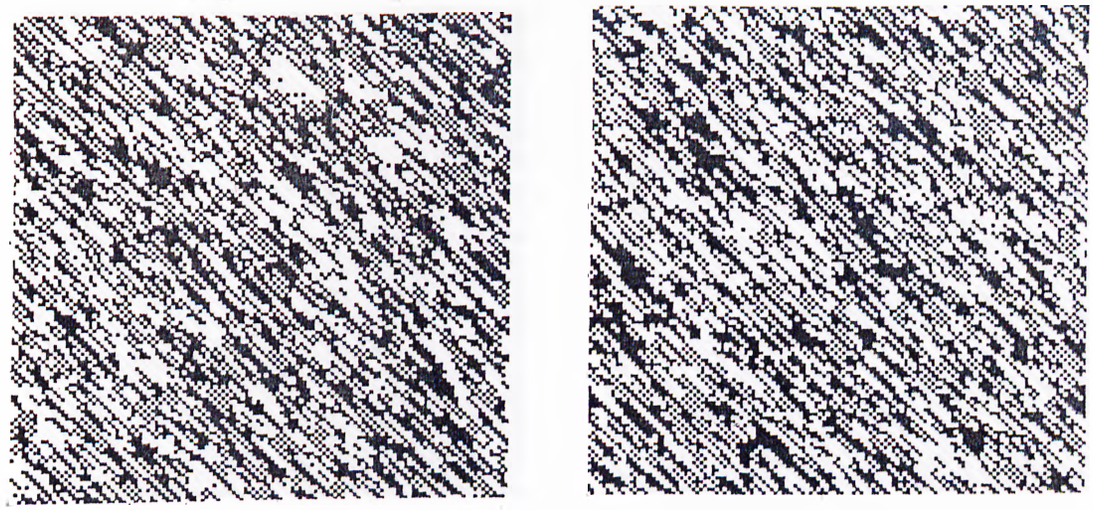
(a) (b)

Figure 3.2. MRF textures with parameters $\alpha = 0.5, \beta_h = 1.5, \beta_v = -0.7, \beta_m = -0.7, \beta_r = -0.7$ generated by (a) sequential, and (b) parallel algorithms.



(a) (b)

Figure 3.3. MRF textures with parameters $\alpha = -3.0, \beta_h = 0.0, \beta_v = 3.0, \beta_m = 0.0, \beta_r = 0.0$ generated by (a) sequential, and (b) parallel algorithms.

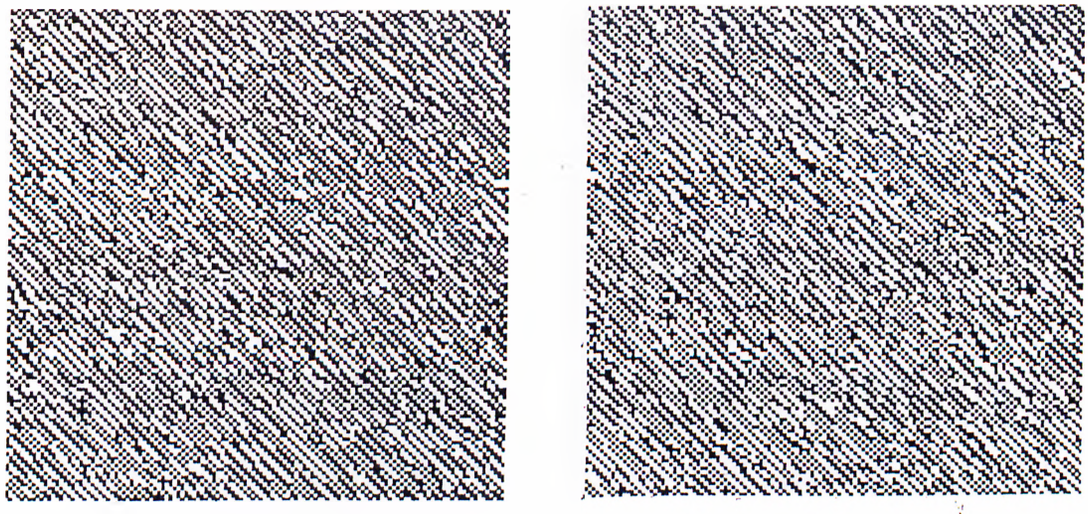


(a) (b)

Figure 3.4. MRF textures with parameters $\alpha = -3.0, \beta_h = 0.0, \beta_v = 0.0, \beta_m = 3.0, \beta_r = 0.0$ generated by (a) sequential, and (b) parallel algorithms.

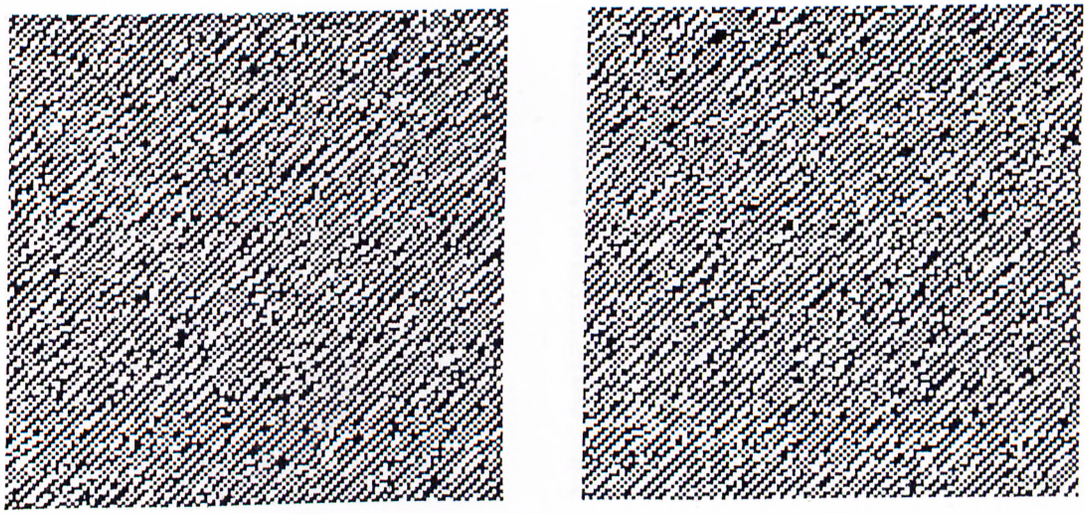
(a) (b)

Figure 3.5. MRF textures with parameters $\alpha = -3.0, \beta_h = 0.5, \beta_v = 0.5, \beta_m = 3.0, \beta_r = -1.0$ generated by (a) sequential, and (b) parallel algorithms.



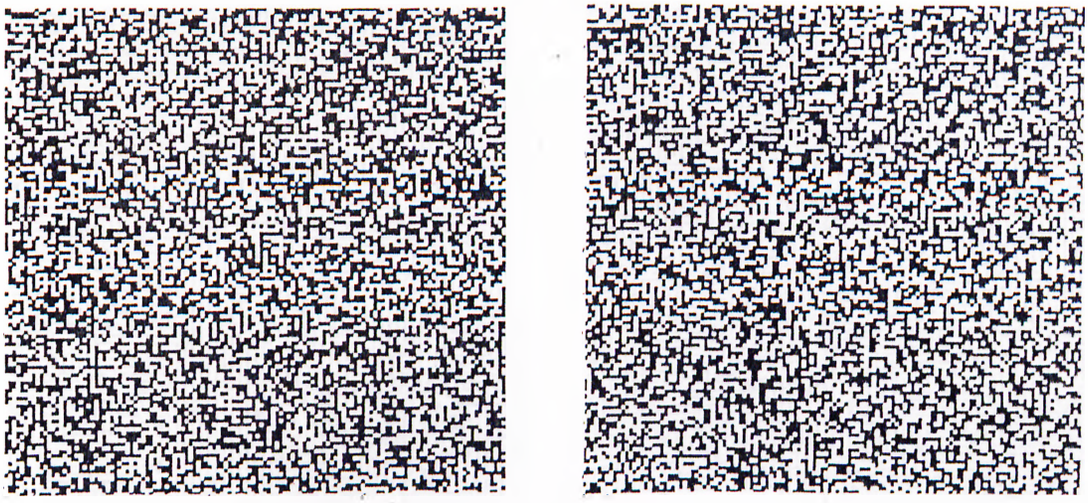
(a) (b)

Figure 3.6. MRF textures with parameters $\alpha = 0.0, \beta_h = -1.0, \beta_v = -1.0$, $\beta_m = 3.0, \beta_r = -1.0$ generated by (a) sequential, and (b) parallel algorithms.



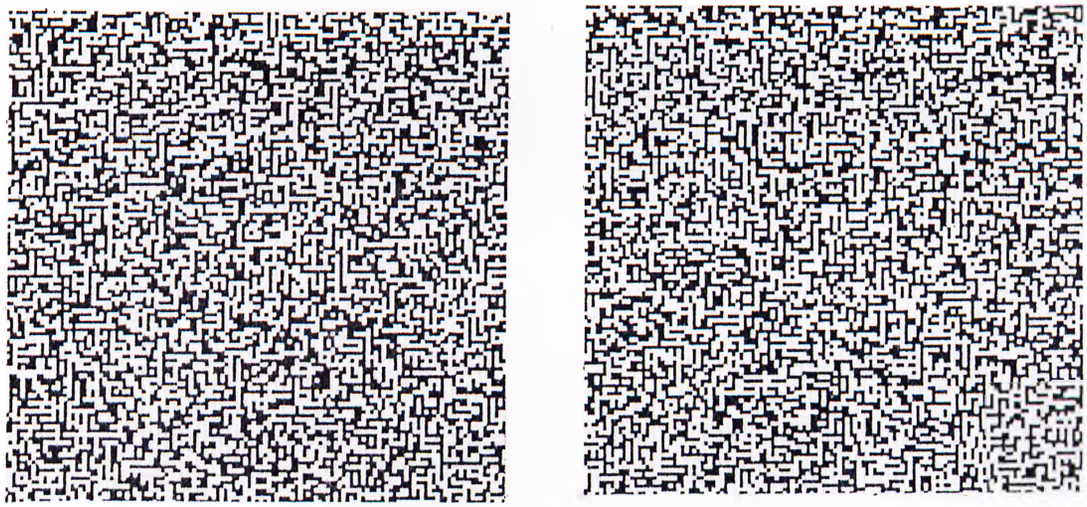
(a) (b)

Figure 3.7. MRF textures with parameters $\alpha = 1.0, \beta_h = -1.0, \beta_v = -1.0$, $\beta_m = -1.0, \beta_r = 2.5$ generated by (a) sequential, and (b) parallel algorithms.



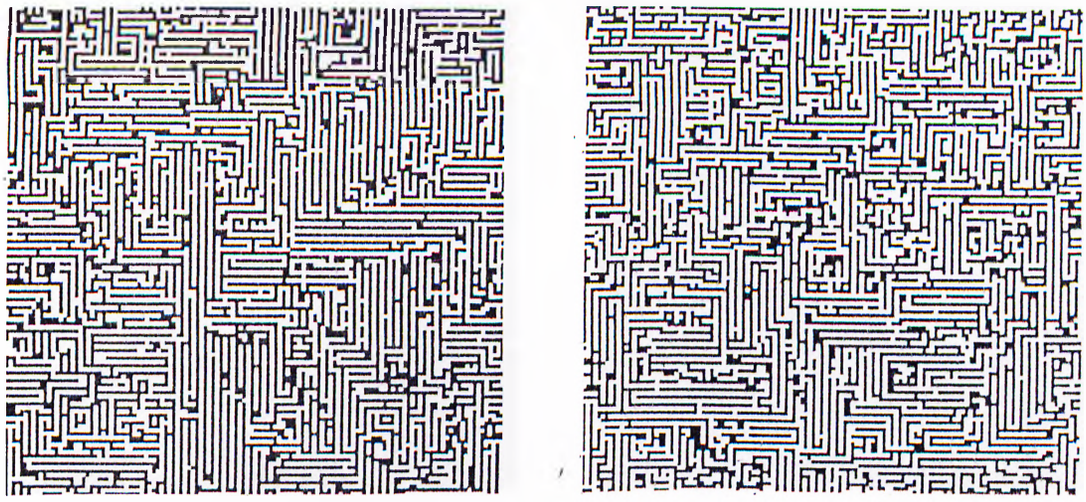
(a) (b)

Figure 3.8. MRF textures with parameters $\alpha = 0.0, \beta_h = 0.5, \beta_v = 0.5, \beta_m = -0.5, \beta_r = -0.5$ generated by (a) sequential, and (b) parallel algorithms.



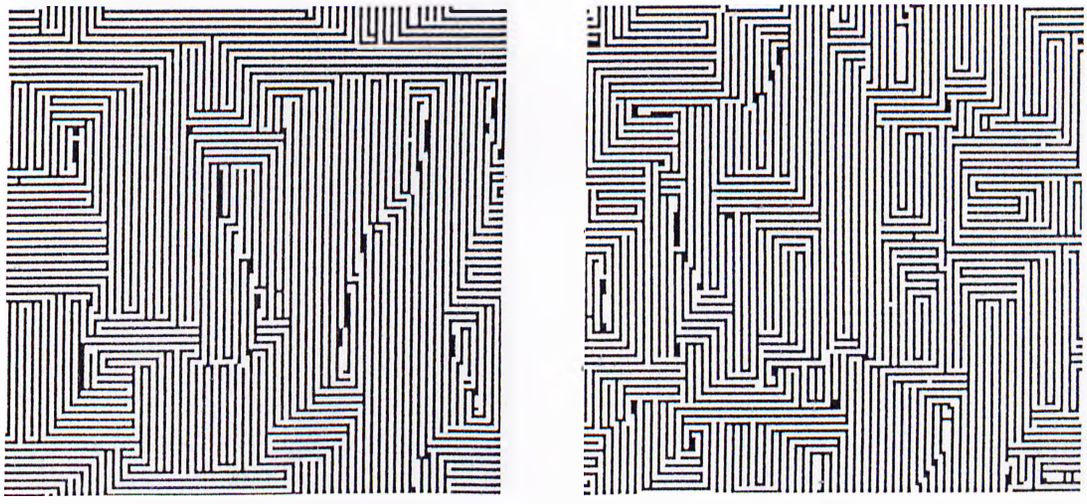
(a) (b)

Figure 3.9. MRF textures with parameters $\alpha = 0.0, \beta_h = 1.0, \beta_v = 1.0, \beta_m = -1.0, \beta_r = -1.0$ generated by (a) sequential, and (b) parallel algorithms.



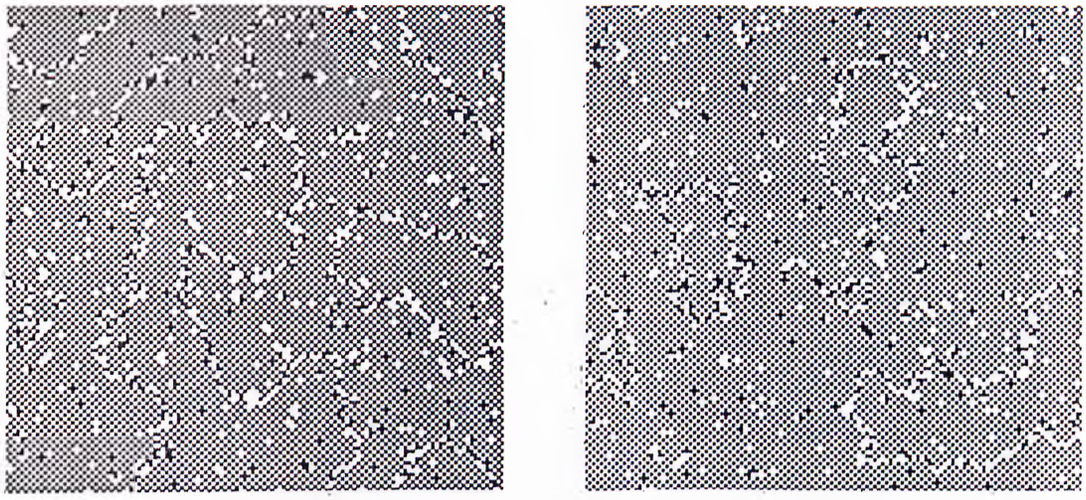
(a) (b)

Figure 3.10. MRF textures with parameters $\alpha = 0.0, \beta_h = 2.0, \beta_v = 2.0, \beta_m = -2.0, \beta_r = -2.0$ generated by (a) sequential, and (b) parallel algorithms.



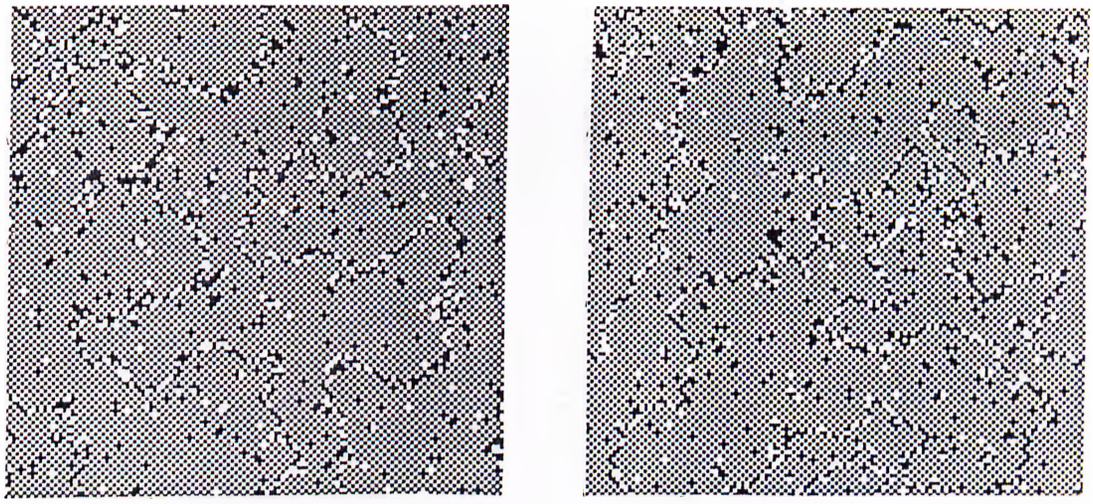
(a) (b)

Figure 3.11. MRF textures with parameters $\alpha = 0.0, \beta_h = 4.0, \beta_v = 4.0, \beta_m = -4.0, \beta_r = -4.0$ generated by (a) sequential, and (b) parallel algorithms.



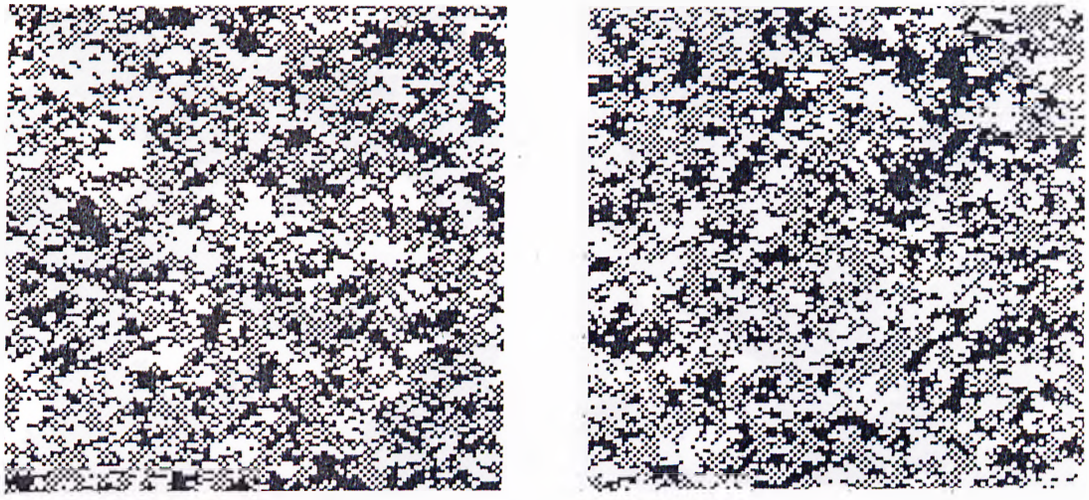
(a) (b)

Figure 3.12. MRF textures with parameters $\alpha = -0.5, \beta_h = -1.0, \beta_v = -1.0$, $\beta_m = 1.0, \beta_r = 1.0$ generated by (a) sequential, and (b) parallel algorithms.



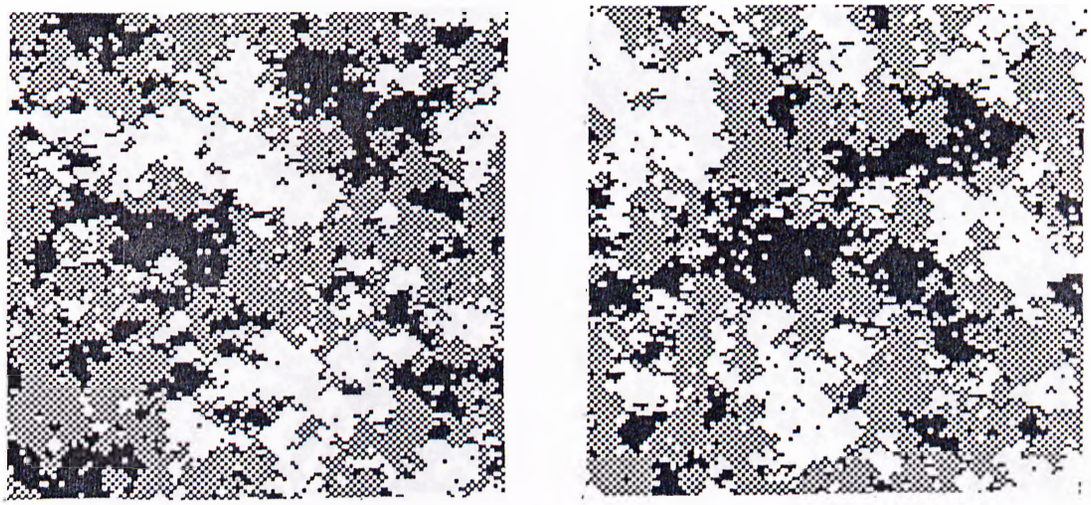
(a) (b)

Figure 3.13. MRF textures with parameters $\alpha = 0.5, \beta_h = -1.0, \beta_v = -1.0$, $\beta_m = 1.0, \beta_r = 1.0$ generated by (a) sequential, and (b) parallel algorithms.



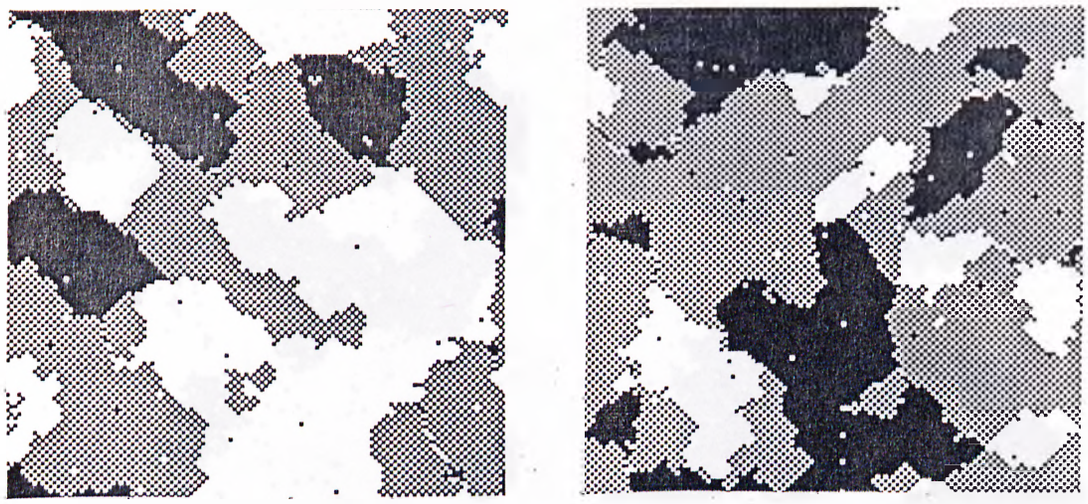
(a) (b)

Figure 3.14. MRF textures with parameters $\alpha = -3.0, \beta_h = 1.5, \beta_v = -1.5$, $\beta_m = 1.5, \beta_r = 1.5$ generated by (a) sequential, and (b) parallel algorithms.



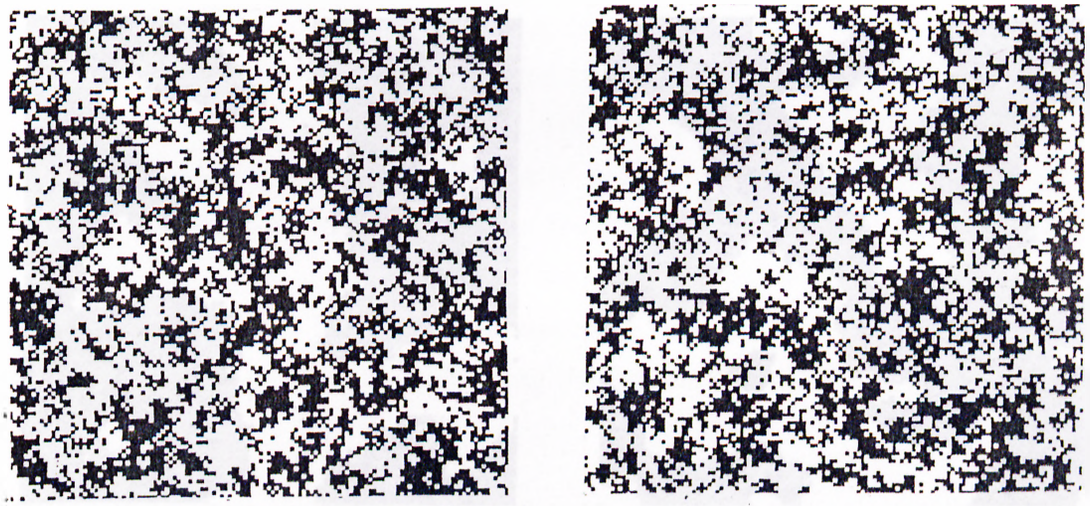
(a) (b)

Figure 3.15. MRF textures with parameters $\alpha = -4.0, \beta_h = 2.0, \beta_v = -2.0$, $\beta_m = 2.0, \beta_r = 2.0$ generated by (a) sequential, and (b) parallel algorithms.



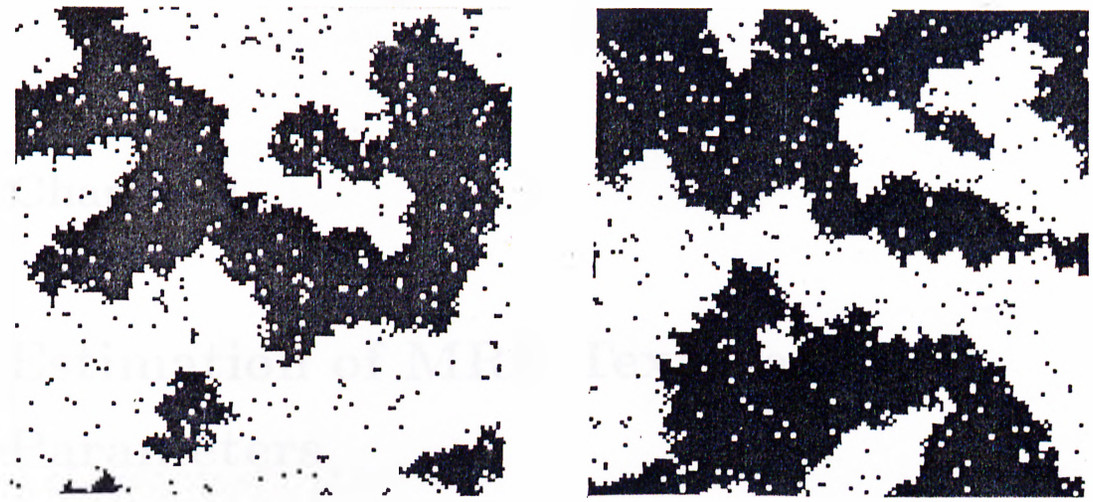
(a) (b)

Figure 3.16. MRF textures with parameters $\alpha = -6.0, \beta_h = 3.0, \beta_v = -3.0$, $\beta_m = 3.0, \beta_r = 3.0$ generated by (a) sequential, and (b) parallel algorithms.



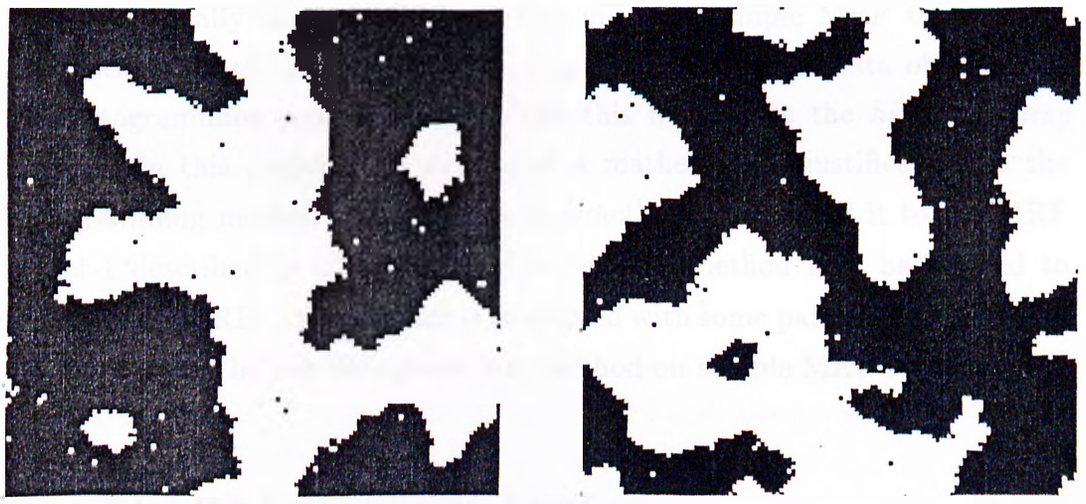
(a) (b)

Figure 3.17. MRF textures with parameters $\alpha = -2.4, \beta_h = 0.6, \beta_v = 0.6$, $\beta_m = 0.6, \beta_r = 0.6$ generated by (a) sequential, and (b) parallel algorithms.



(a) (b)

Figure 3.18. MRF textures with parameters $\alpha = -4.0, \beta_h = 1.0, \beta_v = 1.0$, $\beta_m = 1.0, \beta_r = 1.0$ generated by (a) sequential, and (b) parallel algorithms.



(a) (b)

Figure 3.19. MRF textures with parameters $\alpha = -6.0, \beta_h = 1.5, \beta_v = 1.5$, $\beta_m = 1.5, \beta_r = 1.5$ generated by (a) sequential, and (b) parallel algorithms.

Chapter 4

Estimation of MRF Texture Parameters

4.1 Introduction

The aim of this chapter is to introduce a parameter estimation method for the MRF textures. The method has been proposed by Derin and Elliott [3]. Since it basically consists of histogramming of a sample MRF texture and then estimating the parameters according to the statistical data obtained by the histogramming process, we will call this method as the *histogramming method.* In this chapter, we developed a mathematical justification to the histogramming method. To illustrate the method, we applied it to the MRF Model 1 described in Chapter 2. However, the method may be applied to more general MRFs. The chapter is continued with some parameter estimation results obtained by the histogramming method on sample MRF textures.

4.2 The Histogramming Method

In this section, the histogramming method will be described. To illustrate the method, we will apply it to the MRF Model 1 described in Chapter 2 with a second order neighborhood structure. We will assume that the form of T is

given by (2.14). However, the method may be extended to more general MRFs.

Consider a specific numerical realization of the neighborhood η_i of some point $i \in \mathcal{D}$. From (2.12) we have

$$\frac{P(\tilde{y}(i)=1|\mathcal{Y}(\eta_i))}{P(\tilde{y}(i)=0|\mathcal{Y}(\eta_i))} = e^T \Longrightarrow T = \ln\left(\frac{P(\tilde{y}(i)=1|\mathcal{Y}(\eta_i))}{P(\tilde{y}(i)=0|\mathcal{Y}(\eta_i))}\right) \tag{4.1}$$

For the five parameter expression of T given by (2.14) and using (4.1) we have

$$\alpha+(t+t')\beta_h+(u+u')\beta_v+(v+v')\beta_m+(w+w')\beta_r = \ln\left(\frac{P(\tilde{y}(i)=1|\mathcal{Y}(\eta_i))}{P(\tilde{y}(i)=0|\mathcal{Y}(\eta_i))}\right) . \tag{4.2}$$

Note that (4.2) is a linear equation in the unknown MRF parameters. For different neighborhood structures or by including three or four pixel cliques, the linear expression in (4.2) will contain other parameters, as well. For the model under consideration, the coefficient of α is always 1. The coefficients of the parameters $\beta_h, \beta_v, \beta_m$ and β_r may take on values from the set containing only 0,1 and 2. In some cases two or more different numerical realizations of the neighborhood structure may give the same equation in the form of (4.2). As an example consider the case where the left hand side of (4.2) is :

$$\alpha+\beta_h+\beta_v+2\beta_m+2\beta_r \ .$$

This linear expression may be obtained by four different numerical realizations of the neighborhood structure. These numerical realizations are shown in Figure 4.1.a. Similarly, the linear expression :

$$\alpha+\beta_h+\beta_v+\beta_m+\beta_r$$

is obtained for 16 different numerical realizations of the neighborhood structure which are shown in Figure 4.1.b. On the other hand, the linear expression:

$$\alpha+2\beta_h+2\beta_v+2\beta_m+2\beta_r$$

is obtained only by a specific numerical realization of η_i as shown in Figure 4.1.c.

Due to the positivity assumption on the MRF model under consideration, the right hand side of (4.2) is a finite number. For the MRF probability

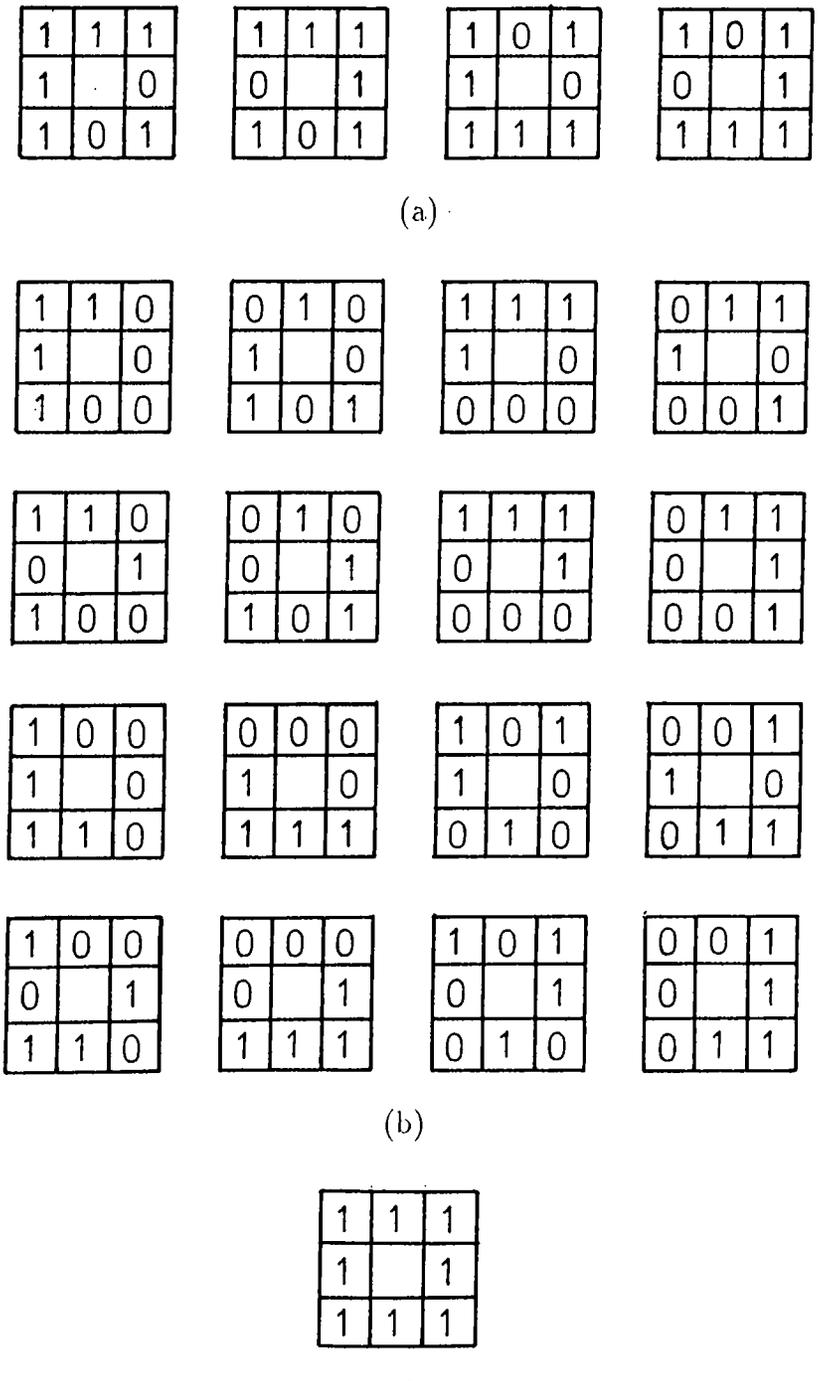


Figure 4.1. Examples of neighborhood realizations giving the same linear equations.

condition because, generally, the number of equations in (4.3) is much larger than the number of unknown MRF texture parameters.

The vector $\underline{d}$ in (4.3) is unknown and it will be estimated from a given sample MRF texture by histogramming. In the following parts, the idea behind histogramming will be developed step by step and at the same time the precise rule of histogramming of a sample MRF texture will become clear.

4.2.1 Histogramming Over an Independent Subset

Consider a specific group of numerical realizations $\mathcal{H}_l$ of the neighborhood structure η. Suppose we have L statistically independent sample observations for the random variable $\tilde{y}(i)$, all of which have neighborhood realizations taken from this group. Then, the probability that $\tilde{y}(i)$ takes on the value 1 and 0 will be fixed. According to the law of large numbers, for any positive real number ϵ, we have

$$\lim_{L\to\infty} P\left(\left|\frac{P(\tilde{y}(i)=1|\mathcal{Y}(\eta_i))}{P(\tilde{y}(i)=0|\mathcal{Y}(\eta_i))} - \frac{N_1^l}{N_0^l}\right| > \epsilon\right) = 0 \qquad L = N_0^l + N_1^l \text{ and } \mathcal{Y}(\eta_i) \in \mathcal{H}_l \tag{4.4}$$

where N_1^l and N_0^l are the numbers of 1 and 0 outcomes respectively. Therefore, it is reasonable make the following approximation,

$$\frac{P(\tilde{y}(i)=1|\mathcal{Y}(\eta_i))}{P(\tilde{y}(i)=0|\mathcal{Y}(\eta_i))} \approx \frac{N_1^l}{N_0^l} \quad . \tag{4.5}$$

The next question to be answered is how to find such independent sample observations for the random variable $\tilde{y}(i)$. The answer comes immediately from the definition of an independent subset of $\mathcal{D}$. For a given MRF and an associated neighborhood structure, the random variables assigned to the points of an independent subset of $\mathcal{D}$ are statistically independent when conditioned on the numerical realization of the remaining part of the MRF. So, choosing an arbitrary independent subset, C, and finding all points in C having neighborhood realizations taken from the same group $\mathcal{H}_l$, will give us the independent observations. Among these observations, let N_1^l and N_0^l be

the numbers of 1 and 0 outcomes, respectively. Then, by the weak law of large numbers, the approximation in (4.5) may be used. Note that we should omit those cases in which N_1^l or N_0^l comes out to be zero. Following this method and carrying out the histogramming over the chosen independent subset of $\mathcal{D}$ for all possible $\mathcal{H}_l$'s, one can find an estimate, $\underline{\hat{d}}$, of $\underline{d}$ and write down the following system of linear equations

$$X_r \underline{\omega} = \underline{\hat{d}} \tag{4.6}$$

where $\hat{d}_l$, the l'th entry of the vector $\underline{\hat{d}}$, is given by

$$\hat{d}_l = \ln\left(\frac{N_1^l}{N_0^l}\right) \tag{4.7}$$

The system of equations in (4.6) will in general contain less equations than the one in (4.3). This is simply because we have omitted some of the equations since either the number of 1 outcomes or 0 outcomes came out to be zero. Therefore, the matrix X_r in (4.6) is obtained from the matrix X in (4.3) by deleting some rows of X. The number of columns of the matrix X_r is equal to the number of parameters to be estimated. Also, it should be noted that since the entries of $\underline{\hat{d}}$ differ from the corresponding entries of $\underline{d}$ by an error term, (4.6) may not be a consistent system of equations in general. So, instead of searching for an exact solution for (4.6), we search for an approximate solution which minimizes a suitable cost function. In this thesis, we have chosen the cost function as the sum of squared errors given by :

$$\mathcal{E} = (X_r \underline{\omega} - \underline{\hat{d}})^T (X_r \underline{\omega} - \underline{\hat{d}}) \ . \tag{4.8}$$

Assuming that X_r has a full column rank, the solution becomes

$$\underline{\hat{\omega}} = (X_r^T X_r)^{-1} X_r^T \underline{\hat{d}} \tag{4.9}$$

where $\underline{\hat{\omega}}$ is an estimate of $\underline{\omega}^*$. So, (4.9) gives an estimate of the parameter set from a single independent subset of $\mathcal{D}$. Note that if the rank of X_r is less than m, then the matrix inversion in (4.9) can not be done and $\underline{\hat{\omega}}$ can not be found by this method.

4.2.2 Histogramming Over the Whole Texture

Now, the histogramming process will be generalized from one independent subset to the whole texture as described in the following paragraphs.

Note that if we were to do the histogramming over the whole texture, the weak law of large numbers would not in general hold because the sample outcomes collected from the texture would no more be statistically independent. However, histogramming over the whole texture is observed to give good estimates of the parameter set and the mathematical justification is as follows. We start by partitioning the whole set $\mathcal{D}$ into disjoint independent subsets $C_k, k = 1, ..., K$, such that

$$\bigcup_{k=1}^{K} C_k = \mathcal{D} \quad .$$

Let $N^l_{1,k}$ and $N^l_{0,k}$ denote the numbers of 1 and 0 outcomes, respectively, in the k'th independent subset obtained by histogramming for the neighborhood realizations in $\mathcal{H}_l$ as described above. Also, let $\mathcal{N}^l_1$ and $\mathcal{N}^l_0$ denote the numbers of 1 and 0 outcomes, respectively, corresponding to neighborhood realizations taken from the same group $\mathcal{H}_l$, obtained by histogramming over the whole MRF texture. Then we have

$$\mathcal{N}^l_1 = \sum_{k=1}^{K} N^l_{1,k} \quad \text{and} \quad \mathcal{N}^l_0 = \sum_{k=1}^{K} N^l_{0,k} \quad . \tag{4.10}$$

From (4.5) and (4.10) we obtain the following approximation:

$$\frac{P(\tilde{y}(i) = 1 | \mathcal{Y}(\eta_i))}{P(\tilde{y}(i) = 0 | \mathcal{Y}(\eta_i))} \approx \frac{\mathcal{N}^l_1}{\mathcal{N}^l_0} \quad . \tag{4.11}$$

The above approximation is a generalization of the simple identity that if for any set of numbers a, b, c, d we have $a/b = c/d = t$ then $(a + c)/(b + d) = t$. However, the accuracy of the approximation in (4.11) depends on several factors.

First of all, since we use weak law of large numbers, the larger the number of independent samples used in (4.5) or in (4.11), the less the mean absolute approximation error will be. So, histogramming over a larger region will

generally be adventageous as long as the samples are independent. On the other hand, if there is a statistical dependence between the newly introduced samples, the approximation in (4.11) may get worse.

Histogramming over the whole texture may introduce statistically dependent observations. However, if for any group of neighborhood realizations, the samples in the whole texture generally lie at sufficiently seperated places, then they will mostly be independent observations and the increase in the number of observations may result in a better parameter estimate.

4.2.3 Modified Histogramming Method

In the parameter estimation method described above, the histogramming has been performed either over the whole or over a fixed subset of $\mathcal{D}$. Another approach to histogramming may be to allow choosing different independent subsets of $\mathcal{D}$ for histogramming for each of the groups, $\mathcal{H}_l, l = 0, ..., n$, of neighborhood realizations.

Then the choice of the independent subset C_l for histogramming for each group $\mathcal{H}_l$ may be done in such a way that C_l contains as many independent samples as possible having the desired neighborhood realizations and the histogramming procedure may be optimized in this way for each $\mathcal{H}_l, l = 1, ..., n$, by optimally choosing an independent subset for each $\mathcal{H}_l$.

As an example consider Figure 4.2. Figure 4.2.a shows a piece of a textured image with its codings. The pixels in the region encircled by a bold line in Figure 4.2.a have value 1 and the other pixels have value 0. Suppose that histogramming is to be performed for the realization of the neighborhood structure containing all 1's. Histogramming over coding 1 will give the two points as shown in Figure 4.2.b. If the histogramming is done on coding 2, the two points shown in Figure 4.2.c will be counted. Histogramming over coding 4 will give only one point as shown in Figure 4.2.d whereas coding 3 will give no points in the histogramming process. Figure 4.2.e shows the 5 points which are counted if the histogramming is done on the whole region, however there

will be correlations between these 5 observations. In Figure 4.2.f, the result of the application of the modified histogramming method is shown. The points are placed so that they all fall in an independent subset and the independent subset is chosen so as to maximize the number of observations.

In this thesis, we will not use the modified histogramming method and no experimental results will be included.

4.3 Experimental Results

In order to test the performance of the histogramming method, several experiments have been performed on sample MRF textures. The reliability of the observations on these experimental results is dependent on the quality of the pseudo-random number generators used in generating the sample textures. To achieve reliable observations on this parameter estimation method, the same experiments have been performed on sample textures generated by using different pseudo-random number generators. More specifically, three different random number generators have been used which are described in Chapter 3. In all cases, the accuracy of the parameter estimates were similar especially for sufficiently large image regions such as 128*128 or larger pixel arrays.

Some of the experimental results are presented in Tables 4.1 to 4.3. The data appearing in Tables 4.1 to 4.3 are obtained on textures generated by the sequential algorithm described in Chapter 3. For the generation of pseudo-random numbers, the *drand48()* function supplied by the C compiler on Unix operating system has been used [24].

The program listed in Appendix B reads an image file corresponding to an image having 256*256 pixels. The image file is assumed to be obtained by scanning the rows of a rectangular image region of 256*256 pixels from left to right, beginning with the first row of the image until the end of the last row. Then, depending on the value of the *aoissz* parameter assigned inside the program, histogramming is done on a square shaped portion of the image

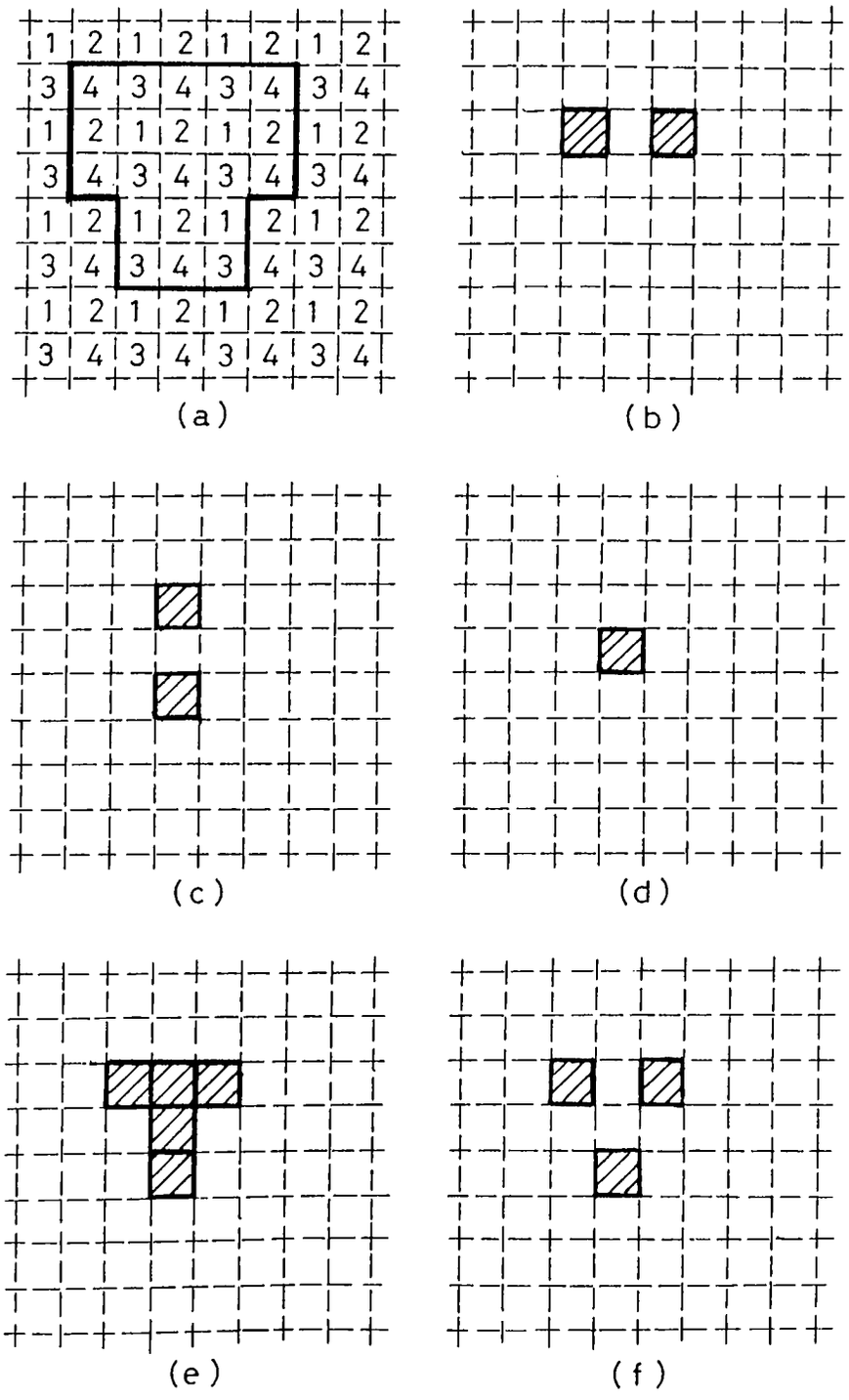


Figure 4.2. Examples of histogramming over several regions.

whose side length corresponds to 2*aoissz+2 pixels. More specifically, on this portion of the image the histogramming process is performed not only on the four codings of the image region as shown in Figure 2.5.b but on the whole of that portion of the image, as well. The program then generates an output file named *omtrx* which has the following format

$$\begin{array}{ll} A_0 & b_0 \\ A_1 & b_1 \\ A_2 & b_2 \\ A_3 & b_3 \\ A_4 & b_4 \end{array}$$

where A_i's are 5*5 matrices and b_i's are 5*1 vectors for $i = 0, ..., 4$. For $i = 0, ..., 3$, the matrix and vector pairs, (A_i, b_i) correspond to the four codings of the specified portion of the image. The pair (A_4, b_4) corresponds to the whole of the specified portion of the image.

Using these matrix and vector pairs, the estimate of the parameter set which is defined by (4.9) is found on the desired coding or on the whole of the specified portion of the image by the following formula:

$$\hat{\underline{\omega}} = A_i^{-1} * b_i \qquad i \in \{0, ..., 4\} \ . \tag{4.12}$$

In Tables 4.1 to 4.3, several experimental results are listed. The specified parameters correspond to textures similar to the ones shown in Figures 3.2 to 3.19. In all tables, the specified parameters, given in the first columns, are the parameters used in generating the sample textures on which the histogramming method have been applied. The estimated parameters are given in the other columns. In some cases, the A_i matrices appearing in (4.12) came out to be nearly or exactly singular and, therefore, parameter estimation could not be done. Those parts in Tables 4.1 to 4.3 which are left empty correspond to the cases in which the A_i matrices came out to be nearly or exactly singular.

In Table 4.1 parameter estimation results on the four codings and on the whole of 256*256 pixel arrays are given. In Table 4.2, results of parameter

estimation corresponding to 128*128 portions of the textured image are presented. Finally, in Table 4.3 parameter estimation results on smaller portions of the textures are presented. In Table 4.3, only the results corresponding to histogramming over the whole of the specified portions of the textured images are given and the results corresponding to the codings are not included.

Specified Parameters	Estimated Parameters				
	Coding 1	Coding 2	Coding 3	Coding 4	Whole Texture
α =0.5	0.4260	0.6400	0.5635	0.5490	0.5200
β_h =1.5	1.4982	1.3934	1.4018	1.4026	1.5081
β_v =-0.7	-0.7658	-0.9363	-0.7133	-0.9163	-0.8198
β_m =-0.7	-0.6192	-0.5775	-0.6820	-0.5440	-0.6615
β_r =-0.7	-0.7226	-0.6333	-0.6822	-0.6739	-0.7019
α =-3.0	-3.1060	-2.9607	-2.8215	-3.0198	-3.1505
β_h =0.0	-0.0397	-0.0266	-0.1676	-0.0833	-0.0672
β_v =3.0	3.0356	3.1129	2.9811	2.9834	3.1177
β_m =0.0	-0.0209	0.0232	0.0724	0.0453	0.0414
β_r =0.0	0.1096	-0.0689	0.0445	-0.0030	0.0338
α =-3.0	-3.0109	-3.0849	-3.0206	-2.9269	-3.0780
β_h =0.0	0.0594	0.0570	-0.0562	-0.0233	-0.0258
β_v =0.0	-0.0288	-0.0554	0.0499	-0.0076	0.0377
β_m =3.0	3.0210	3.0285	3.0634	3.0232	3.0569
β_r =0.0	-0.0195	0.0033	-0.0573	-0.0460	-0.0186
α =-3.0	-2.6814	-2.7810	-2.8112	-2.8500	-2.9266
β_h =0.5	0.5375	0.5801	0.5004	0.4625	0.5169
β_v =0.5	0.3318	0.5473	0.4764	0.6799	0.4541
β_m =3.0	2.9053	2.9066	2.8532	2.8726	3.0380
β_r =-1.0	-1.0904	-1.1272	-1.0467	-1.0772	-1.1013
α =0.0	0.9565	0.2978	0.9637	0.2100	0.7338
β_h =-1.0	-1.1345	-1.0552	-1.1782	-0.9159	-1.0036
β_v =-1.0	-1.2356	-1.0265	-1.2762	-1.1367	-1.2545
β_m =3.0	2.5406	2.7394	2.7991	2.8807	2.8261
β_r =-1.0	-1.0637	-0.9521	-1.2341	-1.0486	-1.2308
α =1.0	1.2732	1.0511	1.0037	1.2219	1.4885
β_h =-1.0	-1.0901	-1.0483	-0.9800	-1.0568	-1.0814
β_v =-1.0	-1.2064	-0.9835	-1.0431	-1.0990	-1.1686
β_m =-1.0	-0.9330	-0.9912	-0.8581	-1.0237	-1.0810
β_r =2.5	2.4411	2.4609	2.4327	2.5152	2.4607

Table 4.1. Parameter estimation results on 256*256 pixel arrays.

Table 4.1 (cont'd)

Specified Parameters	Estimated Parameters				
	Coding 1	Coding 2	Coding 3	Coding 4	Whole Texture
α =0.0	-0.0664	0.1056	-0.0620	0.0574	0.0076
β_h =0.5	0.4921	0.4594	0.5231	0.4974	0.4851
β_v =0.5	0.5489	0.4700	0.5411	0.4891	0.5058
β_m =-0.5	-0.4977	-0.5161	-0.4961	-0.5171	-0.5031
β_r =-0.5	-0.5171	-0.5059	-0.5019	-0.5295	-0.5113
α =0.0	-0.0397	-0.0470	-0.0288	0.0268	0.0006
β_h =1.0	0.9735	0.9806	0.9927	0.9687	0.9894
β_v =1.0	0.9848	0.9601	0.9730	0.9919	0.9879
β_m =-1.0	-1.0537	-0.9957	-0.9925	-0.9720	-1.0124
β_r =-1.0	-0.8809	-0.9458	-0.9700	-0.9759	-0.9681
α =0.0	0.1236	0.0949	0.4286	0.3042	-0.2103
β_h =2.0	1.9491	1.8110	1.6511	1.8645	2.0496
β_v =2.0	1.8348	1.7860	1.7447	1.9222	2.0702
β_m =-2.0	-1.8727	-1.9172	-1.9726	-2.1163	-2.1178
β_r =-2.0	-1.9800	-1.8412	-1.8505	-1.9992	-1.9036
α =0.0	0.0297	0.1999	0.0406	-0.2908	-0.0451
β_h =4.0	3.5825	4.0438	3.5926	3.7164	3.9882
β_v =4.0	3.3228	3.8859	3.4491	4.0482	3.8987
β_m =-4.0	-3.5813	-3.9639	-3.2214	-4.0580	-4.0878
β_r =-4.0	-3.3707	-4.0470	-3.7998	-3.4455	-3.8715
α =-0.5	-0.5111	-0.5207	0.0102	-0.0114	-0.2444
β_h =-1.0	-1.0013	-0.8393	-1.0450	-1.1076	-1.0081
β_v =-1.0	-1.0298	-1.0178	-1.1483	-1.0672	-1.1655
β_m =1.0	1.1537	0.9877	0.6948	0.8765	0.9399
β_r =1.0	0.9737	1.0008	0.9561	0.7676	0.9168
α =0.5	-0.0417	0.3892	-0.0918	0.3224	0.4716
β_h =-1.0	-0.9629	-0.9331	-0.8502	-0.7900	-0.9911
β_v =-1.0	-0.7647	-1.0346	-0.7961	-1.0879	-0.9899
β_m =1.0	0.9832	1.0704	1.0585	1.0213	1.0279
β_r =1.0	1.0951	0.9325	1.0491	0.8387	0.9911

Table 4.1 (cont'd)

Specified Parameters	Estimated Parameters				
	Coding 1	Coding 2	Coding 3	Coding 4	Whole Texture
α =-3.0	-2.9731	-3.0659	-2.9378	-3.0157	-2.9861
β_h =1.5	1.3857	1.3871	1.3270	1.5024	1.4089
β_v =-1.5	-1.2961	-1.4017	-1.3492	-1.5632	-1.4460
β_m =1.5	1.4943	1.4349	1.4494	1.6186	1.5051
β_r =1.5	1.3942	1.5913	1.5074	1.4381	1.5075
α =-4.0	-3.7551	-3.8148	-3.8595	-3.9229	-3.9024
β_h =2.0	1.6499	1.8105	1.8971	1.9410	2.0288
β_v =-2.0	-1.6527	-1.7714	-1.9621	-1.9235	-2.0029
β_m =2.0	1.9952	1.9337	1.9163	1.9406	2.0016
β_r =2.0	1.8879	1.8982	2.0161	1.9626	1.9571
α =-6.0	-5.2499	-5.6953	-5.6670	-6.0857	-5.3117
β_h =3.0	2.0367	2.8018	2.8149	3.3527	2.3597
β_v =-3.0	-2.1489	-2.8003	-2.9263	-3.3099	-2.3597
β_m =3.0	2.7627	2.8641	2.8347	3.2378	2.7393
β_r =3.0	2.7256	3.0399	2.7535	2.8657	2.7208
α =-2.4	-2.4053	-2.3941	-2.4407	-2.4505	-2.4027
β_h =0.6	0.6129	0.5435	0.5729	0.6286	0.5822
β_v =0.6	0.5648	0.6059	0.6209	0.6094	0.5966
β_m =0.6	0.5599	0.6140	0.6228	0.5769	0.5915
β_r =0.6	0.6738	0.6271	0.6141	0.6520	0.6323
α =-4.0	-3.9035	-3.4538	-3.6894	-3.8582	-3.9246
β_h =1.0	1.0483	1.0252	0.8994	0.9152	1.0051
β_v =1.0	0.8428	0.8664	0.8006	1.0465	0.8868
β_m =1.0	1.0185	0.8371	0.9729	0.8759	0.9850
β_r =1.0	0.9472	0.8506	1.0241	0.8556	0.9890
α =-6.0	-5.8180	-5.7137	-5.4105	-5.5431	-5.7803
β_h =1.5	1.4409	1.4588	1.7750	1.4929	1.8798
β_v =1.5	1.3758	1.8131	1.7419	1.7050	1.3830
β_m =1.5	1.4162	1.0041	0.9153	1.2557	1.2010
β_r =1.5	1.5111	1.4186	1.0520	1.1041	1.3004

Specified Parameters	Estimated Parameters				
	Coding 1	Coding 2	Coding 3	Coding 4	Whole Texture
$\alpha = 0.5$	0.5808	1.3293	0.7650	1.1865	0.9269
$\beta_h = 1.5$	1.4409	1.0318	1.3428	1.1077	1.2797
β_v =-0.7	-0.6602	-0.9430	-0.9425	-0.9171	-0.8441
β_m =-0.7	-0.6990	-0.8196	-0.7976	-0.5726	-0.6597
β_r =-0.7	-0.7656	-0.6033	-0.4500	-0.9016	-0.7746
α = -3.0	-2.9503	-3.0610	-2.9712	-2.9931	-3.1016
$\beta_h = 0.0$	0.0626	-0.1222	0.0137	-0.1028	-0.1072
$\beta_v = 3.0$	3.0487	2.8586	2.9513	2.9602	3.0855
$\beta_m = 0.0$	-0.1503	0.0754	0.0980	0.2359	0.0266
$\beta_r = 0.0$	0.0195	0.2140	-0.1035	-0.1396	0.0202
α = -3.0	-2.7613	-2.8934	-2.8539	-2.5211	-2.8865
$\beta_h = 0.0$	-0.0327	0.1138	-0.1723	-0.0089	-0.0322
$\beta_v = 0.0$	0.1181	-0.1034	0.0083	-0.0861	0.0507
$\beta_m = 3.0$	2.7607	2.9210	3.0804	2.7029	2.9962
$\beta_r = 0.0$	0.0328	-0.0663	-0.0438	-0.0990	-0.0647
α = -3.0	-2.5445	-2.5804	-2.3451	-3.1445	-2.7956
$\beta_h = 0.5$	0.5884	0.6745	0.5548	0.6880	0.7316
$\beta_v = 0.5$	0.0911	0.3492	0.2584	0.5186	0.2166
$\beta_m = 3.0$	2.7607	2.4803	2.5562	2.6222	2.8559
β_r =-1.0	-0.7884	-0.8177	-1.0412	-0.7659	-1.0238
$\alpha = 0.0$	0.5140	0.6412	0.9176	0.1221	0.4661
β_h =-1.0	-1.1435	-0.8886	-1.2753	-0.9898	-1.1374
β_v =-1.0	-1.0238	-1.1180	-1.0842	-1.0116	-1.0344
$\beta_m = 3.0$	2.4761	2.3642	2.3851	2.6853	2.7593
β_r =-1.0	-0.8279	-0.9517	-1.0495	-0.8663	-1.0980
$\alpha = 1.0$	1.8754	1.6250	1.2298	0.8427	1.2446
β_h =-1.0	-1.1909	-1.2304	-0.8352	-0.9023	-1.0805
β_v =-1.0	-1.2017	-1.0873	-1.2407	-1.0115	-1.1419
β_m =-1.0	-1.0275	-1.0800	-0.7860	-1.0014	-0.9821
$\beta_r = 2.5$	2.1474	2.2153	2.2760	2.4055	2.4817

Table 4.2. Parameter estimation results on 128*128 pixel arrays.

Table 4.2 (cont'd)

Specified Parameters	Estimated Parameters				
	Coding 1	Coding 2	Coding 3	Coding 4	Whole Texture
$\alpha = 0.0$	0.0349	0.1465	0.0644	0.0638	0.0490
$\beta_h = 0.5$	0.4541	0.4730	0.5025	0.5360	0.4946
$\beta_v = 0.5$	0.4847	0.4273	0.4721	0.4646	0.4712
β_m =-0.5	-0.5837	-0.4560	-0.4707	-0.5466	-0.5007
β_r =-0.5	-0.4562	-0.5443	-0.5173	-0.5915	-0.5430
$\alpha = 0.0$	-0.0196	0.1950	0.0257	0.2950	0.0405
$\beta_h = 1.0$	0.9062	0.9034	0.8607	0.9204	0.9424
$\beta_v = 1.0$	1.0078	0.8145	0.9961	0.8671	0.9660
β_m =-1.0	-1.0149	-0.9653	-0.9675	-1.1268	-1.0458
β_r =-1.0	-0.9043	-0.9481	-0.9308	-0.9756	-0.9407
$\alpha = 0.0$	0.0894	0.2911	0.3661	0.2595	0.3870
$\beta_h = 2.0$	1.7804	1.7126	1.7466	1.8395	1.8377
$\beta_v = 2.0$	1.6873	1.7788	1.7715	1.8386	1.8229
β_m =-2.0	-1.5365	-1.7832	-1.8287	-1.9865	-2.0738
β_r =-2.0	-1.9569	-2.0029	-1.9819	-1.8682	-1.9488
$\alpha = 0.0$	0.1164	-0.4258		-2.0040	0.0729
$\beta_h = 4.0$	3.1963	3.3895		3.9938	3.8237
$\beta_v = 4.0$	2.8825	3.5235		4.3723	3.8118
β_m =-4.0	-3.3127	-2.7886		-3.1768	-3.7710
β_r =-4.0	-2.9144	-3.7908		-3.4463	-4.0142
α = -0.5	0.0249	-0.6726	-0.1793	-0.2396	-0.2396
β_h =-1.0	-0.8386	-0.8254	-1.0244	-1.0469	-0.9677
β_v =-1.0	-1.2342	-1.0704	-1.1842	-1.0233	-1.0374
$\beta_m = 1.0$	0.8809	0.7577	0.8218	0.9101	0.8131
$\beta_r = 1.0$	0.7323	1.2729	1.1187	0.9265	1.0122
$\alpha = 0.5$	0.3751	0.2372	0.0198	0.5490	-0.0059
β_h =-1.0	-1.2752	-1.0202	-0.9501	-1.0722	-0.9803
β_v =-1.0	-0.7995	-0.8589	-0.7906	-0.9510	-0.7703
$\beta_m = 1.0$	0.8188	1.1225	1.0978	0.7753	1.0956
$\beta_r = 1.0$	1.2333	1.0100	1.0150	1.2039	1.0193

Table 4.2 (cont'd)

Specified Parameters	Estimated Parameters				
	Coding 1	Coding 2	Coding 3	Coding 4	Whole Texture
α = -3.0	-3.0946	-3.2513	-2.8858	-2.8367	-2.9500
β_h = 1.5	1.5510	1.4987	1.3267	1.3151	1.4949
β_v =-1.5	-1.5087	-1.4362	-1.4067	-1.3232	-1.5091
β_m = 1.5	1.6167	1.5417	1.3785	1.5142	1.5760
β_r = 1.5	1.4053	1.5920	1.6167	1.3638	1.4229
α = -4.0	-3.9794	-3.6430	-3.6397	-3.5602	-4.0167
β_h = 2.0	1.8190	1.6288	1.7140	1.7201	1.9056
β_v =-2.0	-1.7933	-1.5610	-1.8452	-1.9173	-1.9255
β_m = 2.0	2.0713	1.7639	1.8642	1.8443	2.0561
β_r = 2.0	2.0247	1.9608	1.8837	1.8601	1.9818
α = -6.0	-5.4556	-5.1426	-6.0202	-5.8579	-6.1392
β_h = 3.0	2.5534	2.4712	3.1564	3.3872	2.9428
β_v =-3.0	-2.7558	-2.6397	-3.0760	-3.2860	-2.7246
β_m = 3.0	2.6628	2.6712	2.7300	3.2659	3.1290
β_r = 3.0	2.8024	2.9390	3.0728	2.7373	2.8491
α =-2.4	-2.3587	-2.4097	-2.3598	-2.3333	-2.3445
β_h =0.6	0.5764	0.4970	0.5569	0.6227	0.5273
β_v =0.6	0.6426	0.6039	0.5852	0.5407	0.6019
β_m =0.6	0.6196	0.6965	0.6831	0.5458	0.6405
β_r =0.6	0.5598	0.6263	0.5897	0.6714	0.6014
α = -4.0	-4.1969	-3.8062	-4.0139	-4.2686	-3.7264
β_h = 1.0	1.1390	0.9167	1.1628	1.1139	1.0934
β_v = 1.0	0.8075	0.9579	0.8991	1.2331	0.8744
β_m = 1.0	0.9578	0.8622	1.2298	1.0091	0.8537
β_r = 1.0	1.0514	1.0751	0.8191	0.7450	0.7581
α = -6.0	-4.9216	-5.1062	-5.1441	-5.1215	-5.4431
β_h = 1.5	1.2262	1.5458	1.4187	1.2550	1.7626
β_v = 1.5	1.2027	1.8687	1.2639	1.8569	1.5971
β_m = 1.5	1.0344	0.9017	1.1070	0.9854	0.8905
β_r = 1.5	1.4442	0.8836	1.3578	1.1169	1.2122

Specified Parameters	Estimated Parameters 64*64 Subregion	40*40 Subregion	32*32 Subregion
α =0.5	0.4593	1.0576	-0.3505
β_h =1.5	1.5248	1.2352	1.8490
β_v =-0.7	-0.6413	-0.8000	-0.9490
β_m =-0.7	-0.9182	-0.8306	-0.5772
β_r =-0.7	-0.5835	-0.8151	0.0154
α =-3.0	-3.0509	-2.9378	-1.8317
β_h =0.0	0.0812	-0.0308	0.3616
β_v =3.0	2.9274	2.7800	2.4594
β_m =0.0	-0.0869	-0.1010	-0.2037
β_r =0.0	0.0218	0.0687	-0.5959
α =-3.0	-2.9726	-2.6542	-1.6968
β_h =0.0	-0.0349	-0.0388	-0.0523
β_v =0.0	-0.0488	-0.1306	-0.3220
β_m =3.0	2.9581	2.7117	2.1205
β_r =0.0	0.0886	0.1255	0.0919
α =-3.0	-2.5078	-2.1854	-2.4178
β_h =0.5	0.7400	0.5679	0.5109
β_v =0.5	0.0557	0.1731	0.0590
β_m =3.0	2.5573	2.2960	2.4017
β_r =-1.0	-0.8451	-0.8264	-0.7250
α =0.0	1.1854	1.4703	1.2158
β_h =-1.0	-1.2704	-1.3719	-1.3801
β_v =-1.0	-1.2346	-1.2406	-0.5585
β_m =3.0	2.3667	2.1683	1.8829
β_r =-1.0	-1.1556	-1.2397	-1.1766
α =1.0	1.6495	0.9747	0.6539
β_h =-1.0	-1.0787	-0.9261	-0.3586
β_v =-1.0	-1.1878	-0.9873	-0.8844
β_m =-1.0	-1.1318	-0.8794	-0.8180
β_r =2.5	2.2104	2.0327	1.4936

Table 4.3. Parameter estimation results on smaller pixel arrays.

Table 4.3 (cont'd)

Specified Parameters	Estimated Parameters		
	64*64 Subregion	40*40 Subregion	32*32 Subregion
α =0.0	0.4357	0.8486	0.7270
β_h =0.5	0.3199	0.1364	0.1798
β_v =0.5	0.3186	0.0882	0.0148
β_m =-0.5	-0.5527	-0.5318	-0.5260
β_r =-0.5	-0.5467	-0.4854	-0.4616
α =0.0	0.3693	0.4910	0.5169
β_h =1.0	0.8049	0.7695	0.7400
β_v =1.0	0.7870	0.6160	0.4536
β_m =-1.0	-1.0420	-1.0241	-0.8187
β_r =-1.0	-0.9270	-0.9116	-1.0316
α =0.0	0.1989	0.7380	1.4973
β_h =2.0	1.7996	1.5632	0.9543
β_v =2.0	1.9379	1.5813	0.8170
β_m =-2.0	-1.7857	-1.8854	-1.5323
β_r =-2.0	-2.1394	-2.0364	-1.8737
α =0.0	0.0317		
β_h =4.0	3.3007		
β_v =4.0	3.1962		
β_m =-4.0	-3.2766		
β_r =-4.0	-3.3323		
α =-0.5	-0.4977	-0.6308	-0.4579
β_h =-1.0	-0.7841	-0.5815	-0.4111
β_v =-1.0	-1.1178	-1.2725	-1.4973
β_m =1.0	0.7664	1.0086	1.0451
β_r =1.0	1.0925	0.9295	0.7324
α =0.5	-0.5940	-0.7604	-0.3954
β_h =-1.0	-0.9797	-0.6309	-0.5414
β_v =-1.0	-0.4597	-0.7584	-0.8028
β_m =1.0	1.0192	0.9616	1.0542
β_r =1.0	1.2551	1.4714	0.7915

Table 4.3 (cont'd)

Specified Parameters	Estimated Parameters		
	64*64 Subregion	40*40 Subregion	32*32 Subregion
α =-3.0	-3.0126	-2.8632	-2.6523
β_h =1.5	1.4497	1.3194	1.2943
β_v =-1.5	-1.4654	-1.3839	-1.3906
β_m =1.5	1.6027	1.6441	1.5354
β_r =1.5	1.4550	1.3921	1.2902
α =-4.0	-3.9185	-3.9442	-3.9812
β_h =2.0	2.0758	1.8548	1.9841
β_v =-2.0	-2.0196	-1.9575	-1.9780
β_m =2.0	2.0073	2.1421	2.2734
β_r =2.0	1.9943	1.9001	1.6920
α =-6.0	-5.7667	-5.1644	-5.1852
β_h =3.0	2.9826	3.1402	3.2162
β_v =-3.0	-2.9053	-3.3000	-3.0815
β_m =3.0	2.4750	2.2009	2.6411
β_r =3.0	3.2529	3.2945	2.4877
α =-2.4	-2.6855	-2.5196	-2.4219
β_h =0.6	0.6721	0.6962	0.5625
β_v =0.6	0.5856	0.4085	0.6043
β_m =0.6	0.6453	0.6772	0.5894
β_r =0.6	0.7611	0.7175	0.6397
α =-4.0	-3.9243	-3.7427	-3.3816
β_h =1.0	0.7063	0.9501	0.7250
β_v =1.0	1.2667	1.0120	0.5218
β_m =1.0	1.0153	0.8827	1.1820
β_r =1.0	0.9575	1.0199	1.0374
α =-6.0	-4.4840	-4.9380	
β_h =1.5	1.0066	1.2795	
β_v =1.5	2.1048	1.3535	
β_m =1.5	0.4985	1.6085	
β_r =1.5	0.7032	0.6604	

Chapter 5

A Parallel Network for MRF Texture Generation

5.1 Introduction

In this chapter, we propose a parallel network for the generation of sample MRF textures. The network implements the parallel algorithm described in Chapter 3. We specifically considered the MRF Model 1 described in Chapter 2 and we limited our attention to the form of T given by (2.14). Generalization of the network for larger neighborhood structures and for cliques containing more than two pixels requires some modifications and will not be mentioned in this thesis. The network is a discrete time system.

5.2 A Parallel Network for MRF Texture Generation

The network described in this chapter basically implements the parallel MRF texture generation algorithm described in Chapter 3. The structure of the network resembles the rectangular array of pixels and the interconnections of the network resemble the neighborhood interactions of the pixels. As will be described soon, the network requires a set of random valued signals as inputs to its nodes. A controlling mechanism controls the operation of the network

according to the parallel algorithm described in Chapter 3.

5.2.1 Basic Structure of the Network

In the following description of the network, we will consider a second order neighborhood structure and assume that T is of the form given by (2.14).

The network is a finite rectangular array of nodes with interconnections defined only between the neighboring nodes. Here the concept of a neighborhood is exactly as defined for the MRFs, however it is now defined for an array of nodes instead of an array of pixels. So, the basic structure of the network will be as shown in Figure 5.1. As in the case of pixel arrays, the array of nodes of the network may also be assumed to be toroidal.

Each node of the network is a processing unit with several inputs coming from the outputs of the nodes lying in its neighborhood, an external input through which random signals are applied to the node, a control input and a single output. The input-output diagram of a node is shown in Figure 5.2.

The outputs of all nodes are binary signals which may take on values from the set $\{0, 1\}$. Therefore, the feedback inputs coming from the neighboring nodes are also binary signals. The output signals of the nodes correspond to the binary pixel values of a texture and the nodes themselves resemble the pixels of an image. In other words, the output signal of an arbitrary node i corresponds to the random variable $\tilde{y}(i)$ in an MRF as defined in Chapter 2.

The internal structure of an arbitrary node is as shown in Figure 5.3. It mainly consists of three blocks. The first block is driven by the outputs of the nodes lying in its neighborhood and it generates a signal $v_p(i)$ which is numerically equal to the probability that the output of that node is 1 given the output values of the nodes lying in its neighborhood. In other words $v_p(i)$, the signal output of the first block of the i'th node, is numerically equal to p_i as defined by (3.8).

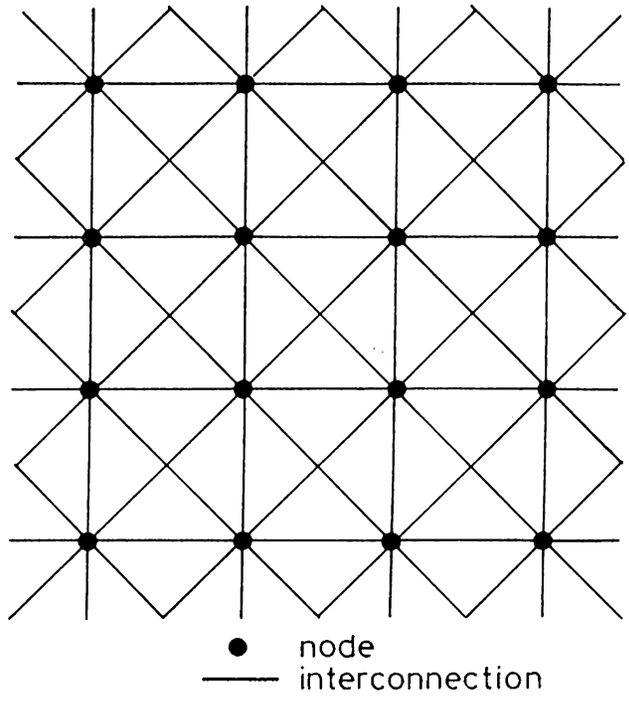


Figure 5.1. Basic structure of the network.

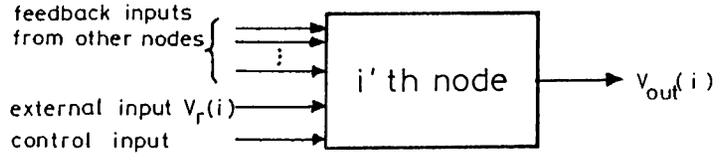


Figure 5.2. Input-output diagram for a single node.

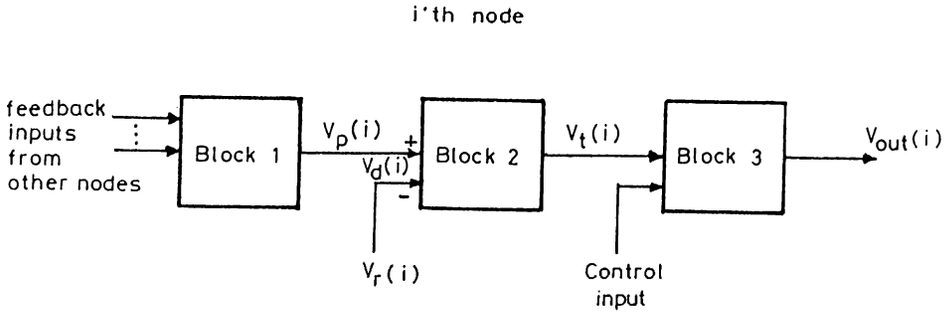


Figure 5.3. Block diagram of a typical node.

The second block seen in Figure 5.3 has two inputs which are the signal $v_p(i)$ and an external random valued input $v_r(i)$. This block is a comparator whose output is a signal corresponding to level 1 when $v_p(i) > v_r(i)$ and to level 0 otherwise. Defining $v_d(i) = v_p(i) - v_r(i)$, we have

$$v_t(i) = \begin{cases} 0 & \text{if} \quad v_d(i) < 0 \\ 1 & \text{if} \quad v_d(i) \geq 0 \end{cases} \tag{5.1}$$

where $v_t(i)$ is the output of the second block. Note that for a fixed input $v_p(i)$ and a uniformly distributed (on $[0, 1)$) input $v_r(i)$, the output $v_t(i)$ of this block becomes 1 with probability that is numerically equal to $v_p(i)$.

The third block of the node is a latch which may be enabled or disabled by the control input. When enabled, the latch becomes transparent and its output $v_{\text{out}}(i)$ becomes equal to $v_t(i)$. Otherwise, it keeps its output unchanged.

5.2.2 Operation of the Network

In order to implement the highly parallel algorithm described in Chapter 3, the set of nodes of the network are partitioned into a finite number of disjoint independent subsets $C_k, k = 1, ..., K$. These independent subsets may be the codings as shown in Figure 2.5. Here the concept of an independent subset is defined in a similar manner as it is defined for MRFs.

Each independent subset has a control line to which the control inputs of the nodes in the corresponding independent subset are connected. Therefore, the nodes of a given independent subset are allowed to change their outputs only when signalled through the corresponding control line.

The network operates according to the following algorithm.

- **Step(1)**: Initialize the latch outputs to arbitrary binary values taken from theset $\{0, 1\}$ and keep the latches disabled.
- **Step(2)**: Randomly choose an independent subset C_k with a fixed non-zero probability P_k as defined by (3.6) and (3.7). At this moment, all latches are disabled but the v_p's are generated inside each node. Present random statistically independent signals to the v_r inputs to all nodes of the chosen independent subset. These random inputs must be uniformly distributed in [0,1). For a short time enable the nodes in the chosen independent subset C_k through the corresponding control line until the outputs are all updated.Then disable all nodes.
- **Step(3)**: Go to Step 2.

Note that when an independent subset is enabled, the outputs of the nodes in that subset are updated as done in Step 2 of the highly parallel algorithm described in Chapter 3. The operation of the network may be stopped after the sample MRF textures defined by the output signals of the network become stable. Here, the concept of stability is as explained in Chapter 3 for the MRF texture generation algorithms.

5.2.3 A Specific Realization

Consider the MRF Model 1 described in Chapter 2 where the conditional probability distribution is given by (2.12). We will assume a second order neighborhood structure and the form of T will be taken as in (2.14). For this model, a node of the network may be realized as shown in Figure 5.4.

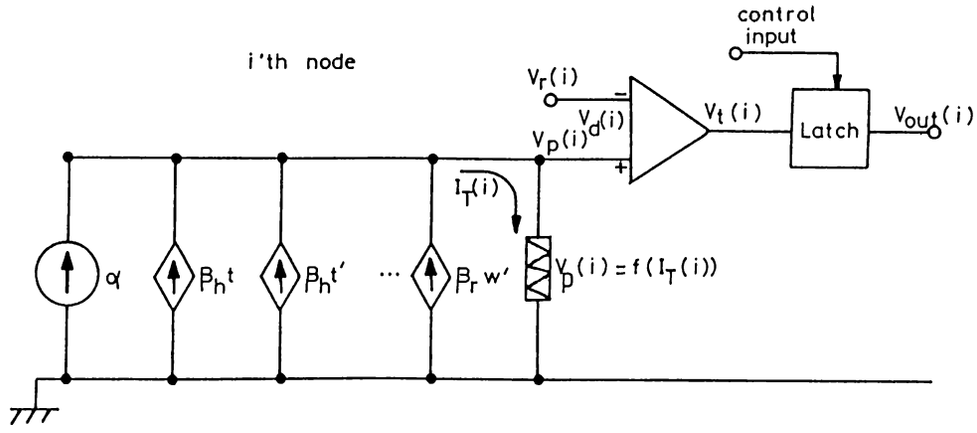


Figure 5.4. Internal structure of a node.

In the circuit shown in Figure 5.4, there is a constant current source of value α and some controlled current sources with transconductances $\beta_h, \beta_v, \beta_m, \beta_r$ which are controlled by the output voltages of the nodes in the neighborhood of the corresponding nodes. The output currents of the independent and the controlled sources sum up to a value $I_T(i)$ as shown in Figure 5.4. So, we have

$$I_T(i) = \alpha + \beta_h(t + t') + \beta_v(u + u') + \beta_m(v + v') + \beta_r(w + w') \qquad (5.2)$$

where $t, t', u, u', v, v', w, w'$ are the outputs of the neighboring nodes which are placed as shown in Figure 2.6. Note that the numerical value of $I_T(i)$ is of the form of T given by (2.14).

The total current $I_T(i)$ is, then, passed through a nonlinear device whose input-output characteristic is given by

$$f(I_T(i)) = \frac{\exp(I_T(i))}{1 + \exp(I_T(i))} \quad . \qquad (5.3)$$

This characteristic is shown in Figure 5.5. It is assumed that the comparator does not drive any current. Note that the characteristics of the nonlinear device is of the form of the conditional probability distribution given by (2.12). So, the output voltage, $v_p(i)$, of this part of the node is numerically equal to the

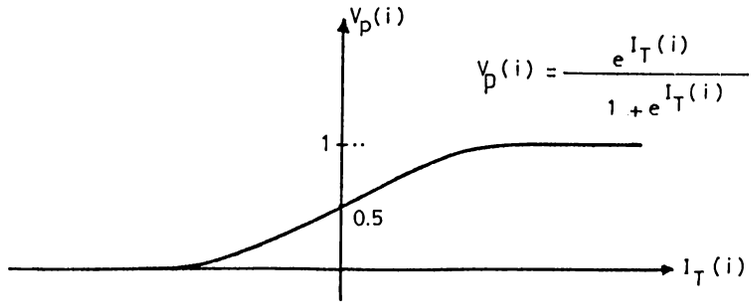


Figure 5.5. Nonlinear device characteristics.

probability that the output $v_{\text{out}}(i)$ of the node will be 1 if it were a random variable described by the conditional probability distribution given by (2.14) .

Finally, the comparator output, $v_t(i)$, described by (5.1), is applied to a latch which is enabled or disabled by the control input.

The complete network is an array of such nodes which are placed and interconnected as shown in Figure 5.1. The operation of the network is as described in Section 5.2.2.

Chapter 6

Conclusion

In this thesis, the MRF texture model is studied. More specifically, generation of sample MRF textures and estimation of MRF texture parameters from a given sample MRF texture are considered. Also a parallel network is developed for the generation of sample MRF textures.

The MRF texture model considered in this thesis is a binary valued MRF defined on a finite rectangular array of points (pixels). The conditional probability distribution has a specific form which depends on a set of parameters called the MRF parameters.

In the texture generation part of this thesis, two different methods for the generation of sample MRF textures are described. The theory behind these methods is generally based on finite state Markov chains in which each numerical realization of the random field is considered to be a state of the Markov chain. The Markov chain is designed to be aperiodic and irreducible so that it has a unique steady state probability distribution independent from the initial state. Then, by suitably defining the transition probabilities between the states of the Markov chain, the steady state distribution is made Gibbsian which is equivalent to the MRF distribution under consideration.

In this thesis, first, a common sequential algorithm is explained and then, another algorithm that can be implemented in a highly parallel manner

is described. A mathematical analysis and a related proof of the parallel algorithm are also included. Although there exist several such algorithms in the literature, we specifically designed the parallel algorithm to develop a parallel network for the generation of sample MRF textures. Computer simulations for both methods have been performed and some results are presented in this thesis.

In the MRF parameter estimation part of this thesis, we studied a method proposed by Derin and Elliott. The method is based on histogramming of a sample MRF texture and obtaining linear equations in the unknown MRF parameters. The set of linear equations generally come out to be inconsistent so instead of searching for an exact solution, a best fitting solution which minimizes some cost function is searched. This best fitting solution then becomes an estimate of the MRF texture parameters. We called this method as the *histogramming method* and gave a mathematical justification to it. Also the performance of this method has been tested on several sample MRF textures and the results have been observed to be satisfactory for many purposes.

Finally, a parallel network is developed for the generation of sample MRF textures. The network runs on discrete time steps and implements the parallel algorithm described in this thesis.

Bibliography

[1] George R. Cross and Anil K. Jain, "Markov random field texture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-5, no. 1, pp. 25-39, January 1983.

[2] Martin Hassner and Jack Sklansky, "The use of Markov random fields as models of texture," *Computer Graphics and Image Processing,* vol. 12, pp. 357-370, 1980.

[3] Haluk Derin and Howard Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-9, no. 1, pp. 39-55, January 1987.

[4] Ramakant Nevatia, "Locating object boundaries in textured environments," *IEEE Transactions on Computers*, pp. 1170-1175, November 1976.

[5] William B. Thompson, "Textural boundary analysis," *IEEE Transactions on Computers,* pp. 272-276, March 1977.

[6] B.Schacter, A. Rosenfeld, and L. S. Davis, "Random mosaic models for textures," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-8, pp.694-702, 1978.

[7] Ryuzo Yokoyama and Robert M. Haralick, "Texture pattern image generation by regular Markov chain," *Pattern Recognition,* vol. 11, pp. 225-233, 1979.

[8] Larry S. Davis, Steven A. Johns, and J.K.Aggarwal, "Texture analysis using generalized co-occurance matrices," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-1, no. 3, pp. 251-259, July 1979.

[9] Robert J. Adler, *The Geometry of Random Fields,* John Wiley & Sons Ltd.,1981.

[10] Yu. A. Rozanov, *Markov Random Fields,* Springer-Verlag, New York, 1982.

[11] P. L. Dobruschin, "The description of a random field by means of conditional probabilities and conditions of its regularity," *Theory of Probability and Its Applications,* vol. 13, no. 2, pp. 197-224, 1968.

[12] John W. Woods, "Two-dimensional discrete Markovian fields," *IEEE Transactions on Information Theory,* vol. IT-18, no. 2, pp. 232-240, March 1972.

[13] Julian Besag, "Spatial interaction and the statistical analysis of lattice systems," *J. Roy. Statist. Soc. B,* vol.36, pp. 192-236, 1974.

[14] Stuart Geman and Donald Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-6, no. 6, pp. 721-741, November 1984.

[15] Frank Spitzer, "Markov random fields and Gibbs ensembles," *Amer.Math.Mon.*, vol.78, pp.142-154, February 1971.

[16] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics,* vol. 21, No. 6, pp. 1087-1092, June 1953.

[17] William Feller, *An Introduction to Probability Theory and Its Applications,* vol. 1, third edition, John Wiley & Sons Inc., New York, 1968.

[18] Peter Whittle, *Systems in Stochastic Equilibrium,* John Wiley & Sons Ltd., 1986.

[19] *Applications of the Monte Carlo Method in Statistical Physics,* K. Binder (editor), second edition, Springer-Verlag, Berlin, 1987.

[20] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes in C,* Cambridge University Press, Cambridge, 1988.

[21] A. Hoogland, J. Spaa, B. Selman, and A. Compagner, "A special-purpose processor for the Monte Carlo simulation of Ising spin systems," *Journal of Computational Physics,* vol. 51, pp. 250-260, 1983.

[22] Robert B. Pearson, John L. Richardson, and Doug Toussaint, "A fast processor for Monte-Carlo simulation," *Journal of Computational Physics,* vol. 51, pp. 241-249, 1983.

[23] Microsoft C Compiler, Run-Time Library Referance Manual, Microsoft Corporation, 1986.

[24] SunOS Reference Manual, Sun release 4.0, Sun Microsystems Inc., 1988.

Appendix A

Programs for MRF Texture Generation

```
/****************************************/
/*                                      */
/*                                      */
/*   A PROGRAM THAT GENERATES           */
/*   SAMPLE MRF TEXTURES DEFINED        */
/*   ON 128*128 PIXEL ARRAYS BY         */
/*   USING THE SEQUENTIAL ALGORITHM.    */
/*   THE PROGRAM IS WRITTEN IN C        */
/*   PROGRAMMING LANGUAGE.              */
/*                                      */
/*                                      */
/****************************************/

#include <stdio.h>
#include <math.h>

char tex[128][128] ;  /* image array  */
```

```
double rparr[2][3][3][3][3] ;
int pval,ph,pv,pm,pr ;
double a,b_h,b_v,b_m,b_r ;
int sz=127 ;

double drand48() ;
void srand48() ;
void opt1() ;
void opt2() ;
void opt3() ;
void opt4() ;

/* start of function main() */
main()
{
int i,c ;
long sd ;
double rni ;

printf("enter seed:\n");
scanf("%ld",&sd);
srand48(sd) ; /* This is for initialization of
                 the random number generator.  */
for(i=0; i<1000; i++){rni=drand48();}

do { printf("MAIN MENU : \n") ;
     printf("press : \n") ;
     printf("1 ------>Input Parameters \n") ;
     printf("2 ------>Initialize \n") ;
     printf("3 ------>One Step Iteration \n") ;
     printf("4 ------>Save Texture \n") ;
     printf("5 ------>Quit \n") ;
     scanf("%d",&c);
```

```
            if(c==1){opt1() ;}
            if(c==2){opt2() ;}
            if(c==3){opt3() ;}
            if(c==4){opt4() ;}
    }
 while ( c != 5) ;
 }
 /* end of function main()  */

 /* start of function opt1()     */

 /* This function is used for    */
 /* entering MRF parameters      */
 /* and generating a look-up     */
 /* table for use in iterations. */
 void opt1()
 {
 double t;

 printf("Enter Texture Parameters: \n");
 printf("a: \n");
 scanf("%lf",&a);
 printf("b_h: \n");
 scanf("%lf",&b_h);
 printf("b_v: \n");
 scanf("%lf",&b_v);
 printf("b_m: \n");
 scanf("%lf",&b_m);
 printf("b_r: \n");
 scanf("%lf",&b_r);


   for(ph=0; ph<=2; ph++)
```

```
    for(pv=0; pv<=2; pv++)
     for(pm=0; pm<=2; pm++)
      for(pr=0; pr<=2; pr++)
       {
       t=a+ph*b_h+pv*b_v+pm*b_m+pr*b_r;
       rparr[0][ph][pv][pm][pr] = exp(t) ;
       rparr[1][ph][pv][pm][pr] = exp(-t) ;
       }
}
/* end of function opt1()  */

/* start of function opt2()   */

/* This function is used to   */
/* initialize the image.      */
void opt2()
{
int i,j ;
double rni ;

for (j=0; j<=sz; j++)
  for (i=0; i<=sz; i++)
    {
    tex[i][j] = 0;
    rni=drand48();
    if(rni<0.5){tex[i][j]=1;}
    }
}
/*  end of function opt2()     */

/*  start of function opt3()   */

/*  This function is used for  */
```

```
/*  one step iteration.        */
void opt3()
{
int x, y ;
int i, j ;
double rni,r ;

for(y=0; y<=sz; y++ )
  for(x=0; x<=sz; x++ )
    {
    rni=drand48();
    i=(int)(128*rni) ;
    rni=drand48();
    j=(int)(128*rni) ;

    pval=tex[i][j] ;
    ph=tex[(i+1)&sz][j]+tex[(i-1)&sz][j] ;
    pv=tex[i][(j+1)&sz]+tex[i][(j-1)&sz];
    pm=tex[(i-1)&sz][(j-1)&sz]+tex[(i+1)&sz][(j+1)&sz] ;
    pr=tex[(i+1)&sz][(j-1)&sz]+tex[(i-1)&sz][(j+1)&sz] ;
    r=rparr[pval][ph][pv][pm][pr] ;

    if(r>=1.0) {tex[i][j]=(pval+1)&1 ;}
    else {
         rni = drand48() ;
         if (rni<r) tex[i][j] = (pval+1)&1 ;
         }
    }
}
/* end of function opt3() */

/* start of function opt4()   */
```

```
/* This function is used to    */
/* save the image data into    */
/* a file. The data is saved   */
/* by scanning the rows of the */
/* image array starting with   */
/* the first row until the     */
/* end of the last row.        */
/* Each pixel of the image     */
/* is represented and saved as */
/* a character type variable.  */
void opt4()
{
int i,j ;
char b ;
char foname[30] ;
FILE *fo ;

 printf("file name: \n") ;
 scanf("%s",foname) ;
 fo = fopen(foname,"wb") ;
 for(j=0; j<=sz; j++)
   for(i=0; i<=sz; i++)
     {
     b = tex[i][j] ;
     fwrite(&b,1,1,fo) ;
     }
 fclose(fo) ;
}
/* end of function opt4()   */

/* end of program  */
```

```
/****************************************/
/*                                      */
/*                                      */
/*   A PROGRAM THAT GENERATES           */
/*   SAMPLE MRF TEXTURES DEFINED        */
/*   ON 128*128 PIXEL ARRAYS BY         */
/*   USING THE PARALLEL ALGORITHM.      */
/*   THE PROGRAM IS WRITTEN IN C        */
/*   PROGRAMMING LANGUAGE.              */
/*                                      */
/*                                      */
/****************************************/

#include <stdio.h>
#include <math.h>

char tex[128][128] ;  /* image array  */
double rparr[3][3][3][3] ;
int ph,pv,pm,pr ;
double a,b_h,b_v,b_m,b_r ;
int sz=127 ;

double drand48() ;
void srand48() ;
void opt1() ;
void opt2() ;
void opt3() ;
void opt4() ;

/* start of function main() */
main()
```

```
{
int i,c ;
long sd ;
double rni ;

printf("enter seed:\n");
scanf("%ld",&sd);
srand48(sd) ; /* This is for initialization of
                 the random number generator.  */
for(i=0; i<1000; i++){rni=drand48();}

do { printf("MAIN MENU : \n") ;
     printf("press : \n") ;
     printf("1 ------>Input Parameters \n") ;
     printf("2 ------>Initialize \n") ;
     printf("3 ------>One Step Iteration \n") ;
     printf("4 ------>Save Texture \n") ;
     printf("5 ------>Quit \n") ;
     scanf("%d",&c);
           if(c==1){opt1() ;}
           if(c==2){opt2() ;}
           if(c==3){opt3() ;}
           if(c==4){opt4() ;}
   }
while ( c != 5 ) ;
}
/* end of function main()  */

/* start of function opt1()     */

/* This function is used for       */
/* entering MRF parameters         */
/* and generating a look-up        */
```

```
/* table to be used in iterations. */
void opt1()
{
double t,ef ;

printf("Enter Texture Parameters: \n");
printf("a: \n");
scanf("%lf",&a);
printf("b_h: \n");
scanf("%lf",&b_h);
printf("b_v: \n");
scanf("%lf",&b_v);
printf("b_m: \n");
scanf("%lf",&b_m);
printf("b_r: \n");
scanf("%lf",&b_r);


   for(ph=0; ph<=2; ph++)
    for(pv=0; pv<=2; pv++)
     for(pm=0; pm<=2; pm++)
      for(pr=0; pr<=2; pr++)
       {
       t=a+ph*b_h+pv*b_v+pm*b_m+pr*b_r;
       ef = exp(t) ;
       rparr[ph][pv][pm][pr] = ef/(1+ef) ;
       }
}
/* end of function opt1()  */

/* start of function opt2()    */

/* This function is used to    */
```

```
/* initialize the image array. */
void opt2()
{
int i,j ;
double rni ;

for (j=0; j<=sz; j++)
  for (i=0; i<=sz; i++)
    {
    tex[i][j] = 0;
    rni=drand48();
    if(rni<0.5){tex[i][j]=1;}
    }
}
/*  end of function opt2()     */

/*  start of function opt3()    */

/* This function is used for    */
/* one step iteration.          */
void opt3()
{
int ci,cj,cd ;
int x, y ;
int i, j ;
int rnc ;
double rni, r ;

for(cd=0; cd<=3; cd++ )
  {
    rni=drand48();
    rnc=(int)(rni*4);
    ci=(rnc)&1 ;
```

```
      cj=(rnc>>1)&1 ;

      for(y=0; y<=63; y++)
        for(x=0; x<=63; x++)
          {
      i=2*x+ci ;
      j=2*y+cj ;
      ph=tex[(i+1)&sz][j]+tex[(i-1)&sz][j] ;
      pv=tex[i][(j+1)&sz]+tex[i][(j-1)&sz];
      pm=tex[(i-1)&sz][(j-1)&sz]+tex[(i+1)&sz][(j+1)&sz] ;
      pr=tex[(i+1)&sz][(j-1)&sz]+tex[(i-1)&sz][(j+1)&sz] ;
      r=rparr[ph][pv][pm][pr] ;

           rni = drand48() ;
           tex[i][j]=0;
           if (rni<r) tex[i][j] = 1 ;
          }
    }
}
/* end of function opt3() */

/* start of function opt4()   */

/* This function is used to     */
/* save the image data into     */
/* a file. The data is saved    */
/* by scanning the rows of the  */
/* image array starting with    */
/* the first row until the      */
/* end of the last row.         */
/* Each pixel of the image      */
/* is represented and saved as  */
/* a character type variable.   */
```

```
void opt4()
{
int i,j ;
char b ;
char foname[30] ;
FILE *fo ;

 printf("file name: \n") ;
 scanf("%s",foname) ;
 fo = fopen(foname,"wb") ;
 for(j=0; j<=sz; j++)
   for(i=0; i<=sz; i++)
     {
     b = tex[i][j] ;
     fwrite(&b,1,1,fo) ;
     }
 fclose(fo) ;
}
/* end of function opt4()   */

/* end of program  */
```

Appendix B

A Program Used for MRF Parameter Estimation

```
/*************************************/
/*                                   */
/*    A PROGRAM USED FOR THE         */
/*    ESTIMATION OF MRF TEXTURE      */
/*    PARAMETERS BY THE              */
/*    HISTOGRAMMING METHOD.          */
/*    THE PROGRAM IS WRITTEN IN      */
/*    C PROGRAMMING LANGUAGE.        */
/*                                   */
/*************************************/


#include <stdio.h>
#include <math.h>

char tex[256][256] ;  /* image array */
int hist[5][3][3][3][3][2] ;
```

```
double dvec[5][3][3][3][3] ;
double bvec[5][5] ;
int xmat[5][5][5] ;
char namei[30];
int cod,ph,pv,pm,pr,pval ;

int sz=255 ;
int aoissz=127 ; /* This parameter is used
     to define a region for histogramming.
     The length of the sides of the region
     corresponds to 2(aoissz+1) pixels.  */
int trshld=1 ;

void ini() ;
void eqns() ;
void mtrx() ;

main()
{
printf("Enter file name: \n");
scanf("%s",namei);
ini() ;
eqns() ;
mtrx() ;
}

/* start of function ini()  */
void ini()
{
int i,j ;

for(cod=0; cod<5; cod++)
  {
```

```
   for(ph=0; ph<3; ph++)
    for(pv=0; pv<3; pv++)
     for(pm=0; pm<3; pm++)
      for(pr=0; pr<3; pr++)
       for(pval=0; pval<2; pval++)
        {hist[cod][ph][pv][pm][pr][pval]=0;}
   for(i=0; i<5; i++)
     {
     for(j=0; j<5; j++)
       {xmat[cod][i][j]=0 ; }
     bvec[cod][i]=0 ;
     }
   }
}
/* end of function ini() */

/* start of function eqns() */
void eqns()
{
FILE *fi;
int n0, n1;
double t, dk, eq1, eq0;
int i,j,ci,cj ;
char b;

fi=fopen(namei,"rb");
for(j=0; j<=sz; j++)
 for(i=0; i<=sz; i++)
  {
  fread(&b,1,1,fi);
  tex[i][j]=b&1;
  }
fclose(fi);
```

```
for(j=0; j<=aoissz; j++)
 {
 for(i=0; i<=aoissz; i++)
  {
  ci=2*i; cj=2*j;
  pval=tex[ci][cj];
  ph=tex[(ci-1)&sz][cj]+tex[(ci+1)&sz][cj];
  pv=tex[ci][(cj-1)&sz]+tex[ci][(cj+1)&sz];
  pm=tex[(ci-1)&sz][(cj-1)&sz]
    +tex[(ci+1)&sz][(cj+1)&sz];
  pr=tex[(ci-1)&sz][(cj+1)&sz]
    +tex[(ci+1)&sz][(cj-1)&sz];
  hist[0][ph][pv][pm][pr][pval]=
    1+hist[0][ph][pv][pm][pr][pval] ;

  ci=2*i+1; cj=2*j;
  pval=tex[ci][cj];
  ph=tex[(ci-1)&sz][cj]+tex[(ci+1)&sz][cj];
  pv=tex[ci][(cj-1)&sz]+tex[ci][(cj+1)&sz];
  pm=tex[(ci-1)&sz][(cj-1)&sz]
    +tex[(ci+1)&sz][(cj+1)&sz];
  pr=tex[(ci-1)&sz][(cj+1)&sz]
    +tex[(ci+1)&sz][(cj-1)&sz];
  hist[1][ph][pv][pm][pr][pval]=
    1+hist[1][ph][pv][pm][pr][pval] ;

 ci=2*i; cj=2*j+1;
 pval=tex[ci][cj];
 ph=tex[(ci-1)&sz][cj]+tex[(ci+1)&sz][cj];
 pv=tex[ci][(cj-1)&sz]+tex[ci][(cj+1)&sz];
 pm=tex[(ci-1)&sz][(cj-1)&sz]
   +tex[(ci+1)&sz][(cj+1)&sz];
```

```
    pr=tex[(ci-1)&sz][(cj+1)&sz]
      +tex[(ci+1)&sz][(cj-1)&sz];
    hist[2][ph][pv][pm][pr][pval]=
      1+hist[2][ph][pv][pm][pr][pval] ;

    ci=2*i+1; cj=2*j+1;
    pval=tex[ci][cj];
    ph=tex[(ci-1)&sz][cj]+tex[(ci+1)&sz][cj];
    pv=tex[ci][(cj-1)&sz]+tex[ci][(cj+1)&sz];
    pm=tex[(ci-1)&sz][(cj-1)&sz]
      +tex[(ci+1)&sz][(cj+1)&sz];
    pr=tex[(ci-1)&sz][(cj+1)&sz]
      +tex[(ci+1)&sz][(cj-1)&sz];
    hist[3][ph][pv][pm][pr][pval]=
      1+hist[3][ph][pv][pm][pr][pval] ;
      }
    }

 for(ph=0; ph<3; ph++)
  for(pv=0; pv<3; pv++)
   for(pm=0; pm<3; pm++)
    for(pr=0; pr<3; pr++)
     for(pval=0; pval<2; pval++)
      {
      hist[4][ph][pv][pm][pr][pval]=
          hist[0][ph][pv][pm][pr][pval]
         +hist[1][ph][pv][pm][pr][pval]
         +hist[2][ph][pv][pm][pr][pval]
         +hist[3][ph][pv][pm][pr][pval] ;
      }

for(cod=0; cod<=4; cod++)
 for(ph=0; ph<=2; ph++)
```

```
   for(pv=0; pv<=2; pv++)
    for(pm=0; pm<=2; pm++)
     for(pr=0; pr<=2; pr++)
      {
      n1=hist[cod][ph][pv][pm][pr][1];
      n0=hist[cod][ph][pv][pm][pr][0];
      eq1=(double)(n1);
      eq0=(double)(n0);
      dk=10000.00;
      if(n0>=trshld && n1>=trshld)
                        { t=eq1/eq0 ;
                          dk=log(t) ; }
      dvec[cod][ph][pv][pm][pr]=dk ;
      }
}
/* end of function eqns()  */

/* start of function mtrx()  */
void mtrx()
{
int i;
double eq;
int x0, x1, x2, x3, x4;
double b;
FILE *fo;

for(cod=0; cod<5; cod++)
 for(ph=0; ph<3; ph++)
  for(pv=0; pv<3; pv++)
   for(pm=0; pm<3; pm++)
    for(pr=0; pr<3; pr++)
     {
     eq=dvec[cod][ph][pv][pm][pr];
```

```
if(eq<1000.0)
{     xmat[cod][0][0]=xmat[cod][0][0]+1;
     xmat[cod][0][1]=xmat[cod][0][1]+ph;
     xmat[cod][0][2]=xmat[cod][0][2]+pv;
     xmat[cod][0][3]=xmat[cod][0][3]+pm;
     xmat[cod][0][4]=xmat[cod][0][4]+pr;

           xmat[cod][1][0]=xmat[cod][0][1];
    xmat[cod][1][1]=xmat[cod][1][1]+ph*ph;
    xmat[cod][1][2]=xmat[cod][1][2]+ph*pv;
    xmat[cod][1][3]=xmat[cod][1][3]+ph*pm;
    xmat[cod][1][4]=xmat[cod][1][4]+ph*pr;

           xmat[cod][2][0]=xmat[cod][0][2];
           xmat[cod][2][1]=xmat[cod][1][2];
   xmat[cod][2][2]=xmat[cod][2][2]+pv*pv;
   xmat[cod][2][3]=xmat[cod][2][3]+pv*pm;
   xmat[cod][2][4]=xmat[cod][2][4]+pv*pr;

           xmat[cod][3][0]=xmat[cod][0][3];
           xmat[cod][3][1]=xmat[cod][1][3];
           xmat[cod][3][2]=xmat[cod][2][3];
  xmat[cod][3][3]=xmat[cod][3][3]+pm*pm;
  xmat[cod][3][4]=xmat[cod][3][4]+pm*pr;

           xmat[cod][4][0]=xmat[cod][0][4];
           xmat[cod][4][1]=xmat[cod][1][4];
           xmat[cod][4][2]=xmat[cod][2][4];
           xmat[cod][4][3]=xmat[cod][3][4];
  xmat[cod][4][4]=xmat[cod][4][4]+pr*pr;

  bvec[cod][0]=bvec[cod][0]+eq;
```

```
      bvec[cod][1]=bvec[cod][1]+ph*eq;
      bvec[cod][2]=bvec[cod][2]+pv*eq;
      bvec[cod][3]=bvec[cod][3]+pm*eq;
      bvec[cod][4]=bvec[cod][4]+pr*eq;      }

        }

 fo=fopen("omtrx","wt");

 for(cod=0; cod<5; cod++)
  {
  fprintf(fo,"A_%d  b_%d \n",cod,cod) ;
  for(i=0; i<5; i++)
   {
   x0=xmat[cod][i][0];
   x1=xmat[cod][i][1];
   x2=xmat[cod][i][2];
   x3=xmat[cod][i][3];
   x4=xmat[cod][i][4];
   b=bvec[cod][i];
   fprintf(fo,"%d %d %d %d %d   ",x0,x1,x2,x3,x4);
   fprintf(fo,"%.4lf \n",b);
   }
}
fclose(fo);

}
/* end of function mtrx()  */

/* end of program */
```