NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

UNITED STATES NAVY
NAVAL POSTGRADUATE SCHOOL

THESIS

MARKOV RANDOM FIELD TEXTURES AND APPLICATIONS IN IMAGE PROCESSING

by

Christopher A. Korn

March 1997

Thesis Advisors: Carlos Borges
Hal Fredricksen

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE March 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE MARKOV RANDOM FIELD TEXTURES AND APPLICATIONS IN IMAGE PROCESSING	5. FUNDING NUMBERS
6. AUTHOR(S) Korn, Christopher A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

In the field of image compression, transmission and reproduction, the foremost objective is to reduce the amount of information which must be transmitted. Currently the methods used to limit the amount of data which must be transmitted are compression algorithms using either lossless or lossy compression. Both of these methods start with the entire initial image and compress it using different techniques. This paper will address the use of Markov Random Field Textures in image processing. If there is a texture region in the initial image, the concept is to identify that region and match it to a suitable texture which can then be represented by a Markov random field. Then the region boundaries and the identifying parameters for the Markov texture can be transmitted in place of the initial or compressed image for that region.

14. SUBJECT TERMS Markov Random Fields, Image Compression, Textures, cliques, neighborhoods, Gibbs Distributions	15. NUMBER OF PAGES 78
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

MARKOV RANDOM FIELD TEXTURES AND APPLICATIONS IN IMAGE PROCESSING

Christopher A. Korn
Lieutenant, United States Navy
B.S., United States Naval Academy, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
March 1997

Author: [illegible]
Christopher A. Korn

Approved by: [illegible]
Carlos Borges, Thesis Advisor

[illegible]
Hal Fredricksen, Thesis Advisor

W. M. Woods
W. M. Woods, Chair, Department of Mathematics

ABSTRACT

In the field of image compression, transmission and reproduction, the foremost objective is to reduce the amount of information which must be transmitted. Currently the methods used to limit the amount of data which must be transmitted are compression algorithms using either lossless or lossy compression. Both of these methods start with the entire initial image and compress it using different techniques. This paper will address the use of Markov Random Field Textures in image processing. If there is a texture region in the initial image, the concept is to identify that region and match it to a suitable texture which can then be represented by a Markov random field. Then the region boundaries and the identifying parameters for the Markov texture can be transmitted in place of the initial or compressed image for that region.

TABLE OF CONTENTS

ACKNOWLEDGMENT

I would like to express my sincere appreciation for the guidance and teaching by Professor Carlos Borges and Professor Hal Fredricksen. Their enthusiasm and expertise in various field of mathematics made collaboration on this Thesis very rewarding and educational.

I. INTRODUCTION

A. GENERAL

One of the current technologies which affects almost everyone in the military services everyday is image transmission and reproduction. The task of taking an initial visual image and transforming it into an electrical signal and then reproducing it at a final destination is a technically complex undertaking. The amount of information which is contained in a still image is enormous in terms of colors and their locations on the image.

If one were to transmit an image pixel for pixel, the amount of information which would have to be sent would be formidable. It only makes economic sense to try to find a way to transmit and reproduce such images both faster and cheaper. Image compression and reproduction are designed to produce reasonable solutions to these problems.

B. PROBLEM DESCRIPTION

More efficient ways of transmitting images would benefit all armed forces. Tactical information changes rapidly and updating the forces quickly is of paramount importance. Most tactical strategies depend greatly on battlefield imagery, which can change greatly from the time of mission assignment until implementation. The more efficient the transmission of updated images, the quicker they can be sent to field positions. How can better image compression and reproduction be accomplished for the tactical commander to allow him the most timely battlefield information?

C. SCOPE OF THESIS

This paper explores the possibility of using Markov Random Field Textures in image compression and reproduction. The basic idea is to detect a "texture" region in an image, and determine the corresponding Markov parameters for that region. Then the region location and Markov parameters are coded and transmitted to the receiving station.

At the receiving station the Markov parameters are used to produce a representative texture which is displayed in the region. The final region will not look exactly the same as the original region but will have the same approximate texture. The overall result will be a substantial savings in transmission time and amount of data transmitted, with a product image comparable to the initial image.

D. SUMMARY OF CONTENTS

This thesis consists of five chapters with the intent of giving the reader a basic understanding of how Markov Random Field (MRF) Textures can be utilized in images for both compression and reproduction. Some current lossy and lossless image compression techniques as well as the Markov Random Field model and the Gibbs distribution are discussed in Chapter II. In Chapter III the mathematics behind MRF Textures are discussed in detail as well as a representative algorithm for texture generation. Chapter III also shows some representative textures for well-known Markov parameters. Chapter IV discusses the process of determining Markov parameters for an image, and presents an algorithm for parameter estimation for a binary image. Both initial and final images produced with this algorithm are shown. Chapter V examines the effectiveness of using the MRF textures in image compression and suggests further areas of study.

II. IMAGE COMPRESSION

A. LOSSY COMPRESSION

There are two major categories of image compression, lossy and lossless. In lossy compression some of the information in the initial image is lost, while in lossless compression the information is compressed but none is lost. Two well known examples of these types of compression are JPEG(lossy) and JPEG(lossless). These two specific image compression techniques will be explained for illustrative purposes.

1. JPEG Compression

JPEG can be used on either a gray-scale image or a color image. First the image is transformed into a suitable color space. This does not apply to the gray-scale image but for the color image we want to transform the red/green/blue code into a luminance/chrominance color space. The luminance component is gray-scale, and the other two axes are color information. This transformation is performed because substantially more redundant information can appear in the chrominance components since the human eye is not as sensitive to high frequency chroma information. The entire color space does not have to be transformed. Maximum compression can not be reached since you will have to code all of the components at a high luminance quality.

The first step in the compression is to group the component values for each pixel into 8X8 blocks. Each of these blocks is transformed using a discrete cosine transform (DCT). The DCT is related to the Fourier transform and produces a frequency map with 8X8 components. The resulting numbers represent the average value in each block along with successively higher frequency changes within each block. Typically, the high-frequency information can be discarded without affecting the low-frequency information.

The next operation in each block is to divide each of the 64 frequency components by a "quantization coefficient" and round the result to the nearest integer. This is the step where most of the information is lost. The larger the quantization coefficients, the more data is lost. Even the smallest quantization coefficient, one, loses some information since the DCT outputs are usually not integers. Higher frequencies are always quantized with less accuracy than lower frequencies since they are less visible to the human eye. The luminance data is typically quantized more accurately than the chrominance data by using separate 64-element quantization tables since the human eye is more sensitive to it. Most of the today's encoders use a simple linear scaling of the JPEG standard tables with a single "quality" setting which determines the scaling multiplier. Compression ratios achieved are a function of the quality setting desired. Tuning the quantization tables for best results is an active research area.

Finally the reduced coefficients are encoded using either Huffman or arithmetic coding and the appropriate headers are added. In a normal JPEG file all of the compression parameters are included in the headers so that the decompressor can reverse the process. These parameters include the Huffman or arithmetic coding tables and the quantization tables. The tables can be omitted if using a closed system in which the coding tables are already known to both the transmitting and receiving stations.

2. Decompression of Baseline JPEG

The decompression algorithm reverses the process of the compression algorithm. The decompressor multiplies the reduced coefficients by the quantization table entries to produce approximate DCT coefficients. Since these coefficients are only approximate, the reconstructed pixel values are also estimates. If the design has done what it was intended to, the resulting errors will not be highly visible. Typically a high quality decompressor will include some smoothing steps to reduce any pixel-to-pixel discontinuities.

B. LOSSLESS COMPRESSION

1. Lossless JPEG Compression

The quantization step in the DCT introduced error into the compressed and subsequently the restored image. In some cases, such as satellite imagery, it is desirable to restore the exact image. To make the algorithm independent of the encoder and decoder, a predictive coding method is used for the lossless version of JPEG. In the lossless algorithm, the pixel information is transformed by predicting the pixel value based on the values of neighboring pixels. The lossless algorithm is only concerned with the pixel to be predicted and the neighboring pixels; a neighborhood technique is also used in Markov Random Fields. Only pixels that have been previously coded may be used as predictors since their values are available to both the encoder and the decoder.

Once the prediction value is determined, the difference between the predictor and the original pixel value is computed. These differences are then losslessly entropy-coded using either Huffman or arithmetic coding. Since there is no quantization step involved in this process, and the coding step is lossless, the algorithm is therefore lossless.

2. Decompression of Lossless JPEG

In the decoder, the data stream is first entropy decoded using the same Huffman or arithmetic coding used in compression. The difference is then added to the prediction to determine the original pixel value. One can generally expect a compression ratio of 2:1 with the lossless JPEG algorithm.

C. RADIANT TIN COMPRESSION ALGORITHM

The Radiant Tin algorithm is an innovative way of approaching image compression [Ref.1]. Normally, lossy image compression is accomplished by transformation, quantization and coding. In Radiant Tin, instead of the typical transformation step, the original image in its pixel space representation is converted into a symbol space representation.

This conversion extracts features (edges, arcs, and textures) and converts them to symbols in vector space representing the features. These symbolic representations consist of a starting point, ending point, texture information, statistical information defining shape, color standard deviation, relational information, and geometric information defining direction and length.

The conversion process used by Radiant Tin is a two step process. In the first step the edges of the image are determined using a modified Sobel filter which has been constructed to maximize the ability to extract edge information. The filter is a type of gradient operator which consists of four 3X3 masks. These masks are convolved with the image producing a resultant vector which gives the magnitude of the gradient as well as its direction. Gradients which exceed a predetermined threshold level are termed edges.

When the starting point of an edge has been determined, the edge can proceed in only one of three directions (left, forward, or right). Each edge is then traced pixel by pixel with the starting point, ending point, starting direction, directional changes , and length being recorded in tabular format. The second step deals with the texture information on either side of the edge. The average textures on both sides of the edge are determined and recorded. These textures are used to provide the gradient information for the edge determination as well as for image reconstruction.

Once the edges have been determined, the algorithm provides two methods as options to transform and code the image. The first uses the symbolic image. Only the areas of the image where symbols exist are transformed using a modified Mallat wavelet transform. The resulting coefficients are subtracted from the wavelet transform of the entire original image. This leaves only the texture information of the original image which is then coded using a different wavelet transform.

The second method performs transforms in the spatial domain using a residual error coding approach. The symbolic information from the edge determination is compressed and then restored. This information is then subtracted from the original image. The remaining data is then transformed using a modified wavelet transform.

Various levels of compression are achieved by modifying the quality setting (Q from 1 to 100) upon calling the compression routine. The Q-value determines the amount of texture information (method one) or residual data (method two) that is coded. The higher the Q-value the lower the compression ratio. Additionally, the Q-value is used to select the amount of segment data that is detected by the edge detection algorithm.

D. THE MARKOV RANDOM FIELD MODEL AND THE GIBBS DISTRIBUTION

1. The Gibbs Distribution

A Markov Random Field (MRF) is a special case of a Gibbs Distribution (GD). The GD was originally considered in statistical physics, and has subsequently been used for many applications in visual processing. We begin with a description of a basic GD. In this section our attention will be focused on a finite two dimensional lattice. [Ref.2]

First a neighborhood system must be defined on the lattice (**M**). The lattice will have n rows and m columns. In the system, a collection of subsets of **M** described by:

$$\eta = \{ \eta_{ij} : (i,j) \in \mathbf{M} , \eta_{ij} \subseteq \mathbf{M} \}$$

is a *neighborhood system* on **M** if and only if η_{ij}, the neighborhood of the lattice point (i,j), is such that:

1) $(i,j) \notin \eta_{ij}$, and

2) if $(k,l) \in \eta_{ij}$, then $(i,j) \in \eta_{kl}$ for any $(i,j) \in \mathbf{M}$.

One specific hierarchically ordered sequence of neighborhood systems that is commonly used in image modeling is known as the nearest-neighbor model [Ref. 3]. We will designate the first-order neighborhood structure consisting of the four closest neighbors as η^1, while η^2 represents the second-order neighborhood structure which uses the eight closest neighbors. Likewise, η^m would represent the mth order neighborhood system. The neighborhood structures are shown in Figure 2.1. The blocks with 1's are the first order neighborhood of the (i,j) entry designated η^1, while the set of all blocks with 1's and 2's is the second-order neighborhood designated as η^2, and so on.

5	4	3	4	5
4	2	1	2	4
3	1	(i,j)	1	3
4	2	1	2	4
5	4	3	4	5

Figure 2.1 Hierarchical Neighborhood Table.

Since a finite lattice is used, the neighborhoods for entries (or pixels) on the boundaries of the lattice are smaller unless a toroidal lattice structure (one in which the lattice wraps around on itself) is assumed. In order to create this periodic lattice structure a torus is formed from the lattice, basically all of the entries on the left boundary are placed adjacent to all of the associated right boundary entries and all of the upper boundary entries are placed adjacent to their associated lower boundary entries. It is not necessary that the defined neighborhood be hierarchically ordered (established by pixels a defined distance from the objective pixel), symmetric, or isotropic. A neighborhood can be designated in other ways, but only the hierarchically ordered neighborhoods will be addressed in this thesis for ease of explanation and computation.

The *cliques* associated with a neighborhood pair (**M**, η) are defined as follows:

A clique of (**M**, η), denoted by c is a subset of **M** such that:

1) c consists of a single pixel entry, or

2) for $(i,j) \neq (k,l)$, $(i,j) \in c$ and $(k,l) \in c$ implies that $(i,j) \in \eta_{kl}$

The collection of all cliques of (M, η) is denoted by $C(M, \eta)$.

Now we can define the Gibbs Distribution(GD). Let η be a neighborhood system defined over the matrix M. A random field $X = \{X_{ij}\}$ defined on M has a GD or is a Gibbs Random Field (GRF) with respect to η if and only if its joint distribution is of the form:

$$P(X=x) = \frac{1}{Z} e^{-U(x)} \tag{2.1}$$

where: $U(x) = \sum_{c \in C} V_c(x)$, is the energy function which is the summation of all of the individual clique potentials for the entire lattice M

$V_c(x)$ = potential associated with clique c

$Z = \sum_{x} e^{-U(x)}$ partition function

The partition function Z is a normalizing constant. The only condition on the clique potential $V_c(x)$ is that it depends solely on the values of the lattice elements in clique c. In all of the cases presented above, capital letters are used to represent random variables while lower case letters are used to represent specific values.

The joint distribution shown in equation (2.1) can be interpreted physically. The smaller U(x) is, the more likely the realization is. Frequently the exponent can be expressed as $-(\frac{1}{T})* U'(x)$, where T is called the temperature and U'(x) is the energy function. When T is at a high value, the magnitude of $\frac{1}{T}$ is smaller and the system is called 'hot' since many realizations are highly probable. In this paper, T(x) and U′(x) are combined and only U(x) is used.

The Gibbs distribution is basically an exponential distribution. If the clique potential function $V_c(x)$ is chosen properly then a wide variety of distributions for continuous and discrete random fields can be formed as Gibbs Distributions. For example: Poisson, Gaussian, binomial, and binary are a few.

A revived interest in GD has come about recently, particularly with respect to image modeling and processing due to the Hammersley-Clifford Theorem. This theorem established a one-to-one correspondence between GRF's and MRF's and is discussed in the Appendix. Since it is known that the GD provides the joint distribution of the random field, many consistency problems in the MRF model can be eliminated and a more workable model can result.

2. A Useful Class of Gibbs Distributions

This section addresses a particular class of GD which is used to model textures and regions. We start with a random field (X) whose individual elements (X_{ij}) are able to take on N different values. Let the set $Q=\{q_1, q_2, q_3, \ldots, q_N\}$ be the different pixel values that X_{ij} can take on. Three things are required to uniquely describe a Gibbs Distribution, the neighborhood system in use (η), the clique potentials ($V_c(x)$), and the associated cliques.

For our GD we assume that the random field is homogeneous. The clique potentials only depend on the clique type and the pixel values in the clique. A second-order neighborhood system (η^2) is used. For the second order neighborhood in use, the clique potentials are defined by assigning a specific parameter to each clique type as shown below.

$$[*,\alpha],\ [**,\beta_1],\ \begin{bmatrix} * & \\ *, & \beta_2 \end{bmatrix},\ \begin{bmatrix} & * & \\ * & & ,\beta_3 \end{bmatrix},\ \begin{bmatrix} * & & \\ & *, & \beta_4 \end{bmatrix},\ \begin{bmatrix} * & * & \\ * & & ,\gamma_1 \end{bmatrix},\ \begin{bmatrix} * & & \\ * & *, & \gamma_2 \end{bmatrix},$$

$$\begin{bmatrix} * & * & \\ & *, & \gamma_3 \end{bmatrix},\ \begin{bmatrix} & * & \\ * & *, & \gamma_4 \end{bmatrix},\ \begin{bmatrix} * & * & \\ * & *, & \xi_1 \end{bmatrix}$$

There are a total of ten clique types, whose associated clique potentials are:

$$V_c(x)[\text{for } \beta\text{'s}, \gamma\text{'s}, \xi\text{'s}] = \begin{cases} -\delta & \text{if all x in c are equal} \\ \delta & \text{otherwise} \end{cases} \qquad \begin{array}{l} \text{where } \delta \text{ is the parameter} \\ \text{for clique type c} \end{array}$$

V_c (x)[for α's] = q_k for x_η

The different parameters (α_K, β_K, γ_K, ξ_k) control different aspects of the GD. The α_k parameters control the percentage of pixels in each region type. The other parameters control the size, and direction of clustering. This class of GD is used extensively for modeling images as texture models, and will be referred to in the sections which follow.

It is much more convenient and efficient to use the conditional probability distribution of this model. The Hammersley-Clifford Theorem allows use of conditional probabilities of the Markov distribution in place of the joint probabilities in the Gibbs distribution since the two are equivalent. The conditional probabilities utilized are shown below:

$$P\{(i,j) \text{ entry}=1 \mid \text{values of its neighbors } \eta_{ij}\} = \frac{e^{\eta_t}}{1 + e^{\eta_t}} = P_1$$

$$P\{(i,j) \text{ entry}=0 \mid \text{values of its neighbors } \eta_{ij}\} = \frac{1}{1 + e^{\eta_t}} = P_2$$

In the case of the second-order neighborhood shown in Figure (2.2). The term η_t is defined as:

$$\eta_t = \alpha + \beta_h (v + v') + \beta_v (u + u') + \beta_m (m + m') + \beta_r (w + w')$$

m	u	w
v	(i,j)	v'
w'	u'	m'

Figure 2.2 Second-order Neighborhood

III. MARKOV TEXTURES

A. MATHEMATICAL ASPECTS

Many images contain regions of similar features like ocean, desert, fields or some other easily recognizable pattern. These regions are termed “textures”. A texture can be thought of as a two-dimensional stochastic, and possibly periodic, image field. Textures give important information on the depth and orientation of an object. They can be used in computer graphics to model textures from real life (i.e. gravel, camouflage, clouds, etc.). In images, Markov Random Fields (MRF) can be used to generate textures similar to the original one but with greatly enhanced efficiency and compression.

Texture is an intrinsic feature of realistic images. There is no universal definition for a texture. In this paper a texture is considered to be a stochastic, low-dimensional image field. In attempting to classify textures, the following six attributes can be used: coarseness, contrast, directionality, line-likeness, regularity, and roughness. These attributes were determined in a study by Tamura [Ref.4].

Most texture research can be specified by the assumptions made about the process of creating the texture. There are two major approaches to the process. The first approach, the *placement rule viewpoint*, considers a texture to be composed of primitives. These primitives may be of varying or deterministic shape, such as geometric objects or patterns. Macrotextures have large primitives and microtextures have small primitives. The textured image is formed from the primitives by rules which specify how the primitives are oriented in the image as well as in relation to each other.

Many textures such as grass, sand, and oceans are not described well by the placement model, since the primitives in these images are very random in shape and cannot be easily described. These textures are handled better by a second viewpoint in which the texture generation processes involves a stochastic assumption. In the stochastic point of view, the texture is considered as a sample from a probability distribution on the image space. The space is usually an NXN grid and the value at each point is a random variable in the range.

The goal of the research in Markov random field textures is to produce a texture analysis and synthesis system which takes texture inputs, analyzes their parameters according to a Markov random field model, and then generates a textured image that is visually and statistically very close to the original image. A number of researchers including George Cross, Anil Jain, and Sateesha Nadabar have shown that Markov random fields can be used effectively and efficiently to model specific textures [Ref. 5].

There is much to be gained from the refinement of the MRF texture methods. An improvement in the method would allow extremely high compression of textured regions of an image. This would lead to substantial cost reduction in transmitting the image data. In order to have a functional system which uses Markov Random Field Textures only two major steps are involved. The first is basically to produce a recognition algorithm which samples the original image and then selects the texture region and assigns appropriate Markov parameters. Then these parameters are sent, along with others specifying the region boundaries, to the receiving station. At the receiving station the second major step is performed in which MRF parameters are used to generate the texture in the desired region.

B. ALGORITHM FOR TEXTURE GENERATION

The property that makes Markov Random Fields so attractive as a model for texture is that they are locally dependent (i.e. pixel values are only influenced by their neighbors). Since a pixel in a MRF model can be viewed as only depending on its nearby pixels, the model can be used on small portions of an image or a large texture area. This local dependence allows use of only a small number of pixels to determine the texture for an entire region.

The algorithm employed in this thesis to create textures is based on the principle that the texture is the time-evolution of a Markov chain which is aperiodic and irreducible. A Markov chain of this type has been shown to produce a unique steady state texture that is virtually independent from its initial state. Since the starting image will have a minimal effect on the final steady state texture, a random initial image can be chosen. Binary images (black and white only) will be used, but gray-scale images which use a finite number of different shades of gray can also be produced by using a more general model. Predetermined parameters will be used to transform the initial random matrix into one which can be identified as a texture.[Ref. 6]

The algorithm used is the Gibbs Sampler Algorithm, which can be used on gray-scale as well as binary images. The initial random matrix is transformed using the Gibbs Sampler Algorithm. The algorithm changes each individual pixel based on the values of its neighboring pixels. It is used on all of the cells in the matrix and iterated a number of times. Any neighborhood system can be used and the algorithm will determine the new value of each pixel based on its neighbors.

The specific algorithm appearing in this section uses a second-order neighborhood model based on a binary image. It determines the new value of y(i) based on its neighbors and the input Markov parameters as depicted in Table (3.1) and equation (3.1). The transformation formula is equation (3.1).

v	u	w'
t	y(i)	t'
w	u'	v'

Table 3.1 Second Order Neighborhood Grid

$$T = \alpha + \beta_h(t + t') + \beta_v(u+u') + \beta_m(v+v') + \beta_r(w+w') \quad (3.1)$$

In the above equation the α and the β's are called the texture parameters. These parameters determine what the final texture will look like. The algorithm is repeated until the image reaches a steady-state texture. The desired texture is usually reached in less than twenty iterations.

In order to increase the speed of the algorithm, a way of implementing it in a parallel manner is desired. This can be accomplished by partitioning the initial image M into K subsets C_k such that:

$$\bigcup_{k=1}^{K} C_k = M$$

In order to use partitioning it is necessary that if an individual pixel is in a subset C_k, then none of its neighbors (as defined by the neighborhood method selected) can be in the same subset. The pixel entries in a subset must be independent of each other, therefore they cannot have a neighbor relationship between them. To provide a clearer understanding of this coding, the necessary coding scheme for the first and second-order neighborhoods are shown in Figure (3.1) and Figure (3.2) respectfully. Notice that in the first-order system only two subsets are required in a normal checkerboard pattern, while four are required in the second-order neighborhood system in an interlaced double checkerboard pattern.

1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1

Figure 3.1 First-order Neighborhood Coding

1	2	1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4	3	4
1	2	1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4	3	4
1	2	1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4	3	4
1	2	1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4	3	4
1	2	1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4	3	4

Figure 3.2 Second-order Neighborhood Coding

The first step is to establish a random matrix of possible pixel values. In the binary case it is a random matrix of ones and zeroes. Secondly, one of the independent subsets C_k of M is chosen at random. Let P_k be the probability of choosing C_k.

When a particular C_k is chosen, we let p_j be the probability that each individual point j in C_k takes on a value of 1 and let q_j be the probability that the same point j takes on the value zero. These probabilities are known from the conditional probability distribution discussed earlier. Thus Y(j)=1 with probability p_j and Y(j) =0 with probability q_j. Now update the pixel value of Y(j) by setting Y(j)=1 with probability p_j and Y(j)=0 with probability q_j. This updating process is repeated for all j in the chosen C_k. This algorithm will continue to cycle through all points j in all subsets C_k after being initialized in the first step, but steady state is usually reached for each C_k when it has been iterated 20 times. Various stopping rules can be built into the program, but the algorithm used in this thesis will limit the number of iterations.

C. MATLAB SAMPLE ALGORITHM FOR TEXTURE GENERATION

A matlab algorithm to generate binary textures based on Markov Random Fields was written. This algorithm uses a sequential (not parallel) iteration process and treats the entire region D instead of subregions as discussed above. The following is the simplified algorithm for the mathematical program of MATLAB which uses a second order neighborhood approach as discussed in part A:

1. The Matlab algorithm

```
function [X] = gibbs(X)

%       [Y] = gibbs(X)

%     This function generates Markov Random Field textures ala Gurelli and Onural.

%     Copyright 1996 by Carlos F. Borges.  All rights reserved.

% Initialize

% p = [0 2 2 -2 -2];% maze

% p = [-4 2 -2 2 2];% clouds-e

% p= [-6 1.5 1.5 1.5 1.5];% clouds-f

p = [0 -2 -2 2 2];                    (1)

[n m] = size(X);

for q=1:20*n*m                        (2)

        % Choose a random site

        k= ceil (n*rand(1));

        j= ceil(m*rand(1));

        if k==1

                id=n;

        else

                id= k-1;
        end
        if k==n
                iu = 1;
        else
                iu= k+1;
        end
        if j==1
                jl=m;
        else
                jl=j-1;
        end
        if j==m
                jr=1;
        else
        jr=j+1;
        end
        T = p(1) + p(2)*(X(k,jl)+X(k,jr)) + p(3)*(X(iu,j) + X(id,j));      (3)
        T = T + p(4)*(X(iu,jl) + X(id,jr)) + p(5)*(X(id,jl) + X(iu,jr));
        rho = exp(T)/(1 + exp(T));             (4)
        if (rand(1) <= rho)
                X(k,j) = 1;
        else
                X(k,j) = 0;
        end
end
```

2. Matlab Code and Algorithm Explanation

The matlab code is divided into 4 sections, sections (1) - (4) as described below. Each section starts at the number in parenthesis and ends when the line of the next numbered section is reached.

a. Section(1)

In this section of code the Markov parameters are initialized to a set of values. The vector **p** has 5 entries and each corresponds to a value of a texture parameter in formula (3.1). The values of some typical texture parameters and the textures they produce are listed in the code. Next since the initial image can be treated as a matrix, n is set equal to the number of rows in the matrix while m is set equal to the number of columns.

b. Section (2)

This section basically identifies the appropriate neighboring row and column indices for the randomly targeted entry. In Section (2) a random choice of an individual site must be made. This is done by choosing a random matrix row and column. The random row number is obtained by first choosing a random number between zero and one and multiplying it by the total number of rows, n. The next integer value greater than or equal to the first number is used to specify the random row. The random column number is chosen in a similar manner. Each of these random numbers can take on values of any possible position in the matrix.

Next the neighbors of the targeted individual entry (element (k,j)) must be found. Since theoretically the matrix is considered a torus, the neighbors for the edge and corner cells must be defined. If the row designation of the target entry is a one then the row for the element's upper neighbors (id) is row n, otherwise the row is k-1; likewise, if the row of the target entry is n then the row for its lower neighbors is row one.

For the column locations of the neighbors of the target entry a similar process is used. If the target entry is in column one, then its left neighbors are in column m, and if the target entry is column $j \neq 1$ then its left neighbors are in column j-1. If the target cell is in column m, then its right neighbors are in column one.

c. Section (3)

In this section the second order transformation equation (3.1) is utilized. Since the neighboring cells have been identified for the target cell , equation (3.1) is a simple substitution into the transformation equation. The entries in the code which use X( ), for example *X(k,jl)*, are actually the matrix element in that specified location. The transformation equation is broken down into two lines to allow easy identification.

d. Section (4)

In section (4) a parameter rho is assigned a value depending on the value of T. In fact rho is the conditional probability distribution function of the binary MRF. Next a random value between zero and one is chosen and compared to rho. The comparison value must be chosen randomly to satisfy the conditional probabilities of a GD. If the random value is less than or equal to rho then the value assigned to the targeted entry for the next iteration is a one, otherwise it is assigned a value of zero. This entire process is repeated 120nm times.

D. TEXTURE PARAMETERS AND EXAMPLES

When using MRF textures, the final image is determined mostly by the texture parameters which are used. For the second-order system being discussed there are five texture parameters which are used. Some possible textures using a few well-known parameter vectors are illustrated. The effects of fixing certain pixel values in the image are also addressed.

1. Texture Examples

The images in Figure (3.3) are four different matrices obtained from different Markov parameters. The five parameters for each image are stored as a vector **p** in the matlab file, the actual values are shown in Table (3.2).

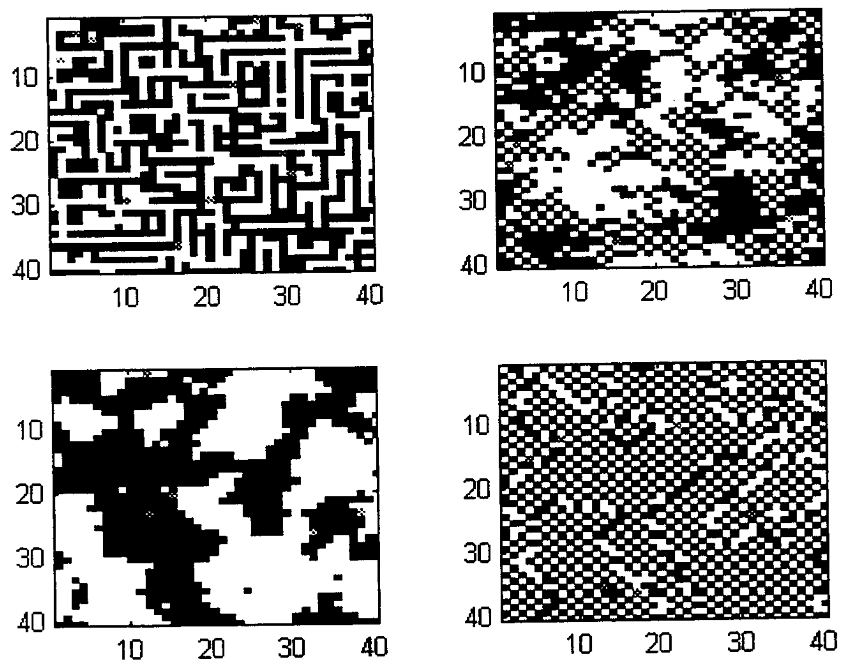

Figure 3.3 Resultant textures for different Markov parameters.

	α	β_h	β_v	β_m	β_r
top left	0	2	2	-2	-2
top right	-4	2	-2	2	2
bottom left	-6	1.5	1.5	1.5	1.5
bottom right	0	-2	-2	2	2

Table 3.2 Markov parameters for Fig. 3.3

The following is the matlab code used to produce Figure (3.3) :

```
»X = rand(40,40);   (1)
» gibbs             (2)
» Y = 64*X;         (3)
» subplot(2,2,1);image(Y);colormap('gray');        (4)
» X = rand(40,40);
» gibbs
» Y = 64*X;
» subplot(2,2,2);image(Y);colormap('gray');
» X = rand(40,40);
» gibbs
» Y = 64*X;
» subplot(2,2,3);image(Y);colormap('gray');
» X = rand(40,40);
» gibbs
» Y = 64*X;
» subplot(2,2,4);image(Y);colormap('gray');
```

In the above code, before each successive use of the texture generation algorithm the p-vector must be adjusted to the desired value in the gibbs file. In step(1) the code produces a random 40X40 matrix of values between zero and one. In step(2) the texture generation algorithm transforms the X matrix into a random matrix of zeroes and ones based on the Markov parameters. The binary matrix is then multiplied by 64 to allow sufficient color separation. Step(4) creates a figure of four images and designates where the current image will be placed. It also assigns a white color to the zero entries of the individual matrices and a black color to the entries equal to 64 in Y.

In Figure (3.4) the same p-vectors have been used. However, only a different initial random matrix is used. The matrices in this figure are also 50X50 instead of the 40X40 matrices used in Figure (3.3). Since the only consistencies between the two figures are the Markov parameters, it is easily seen how these parameters dictate the final texture.

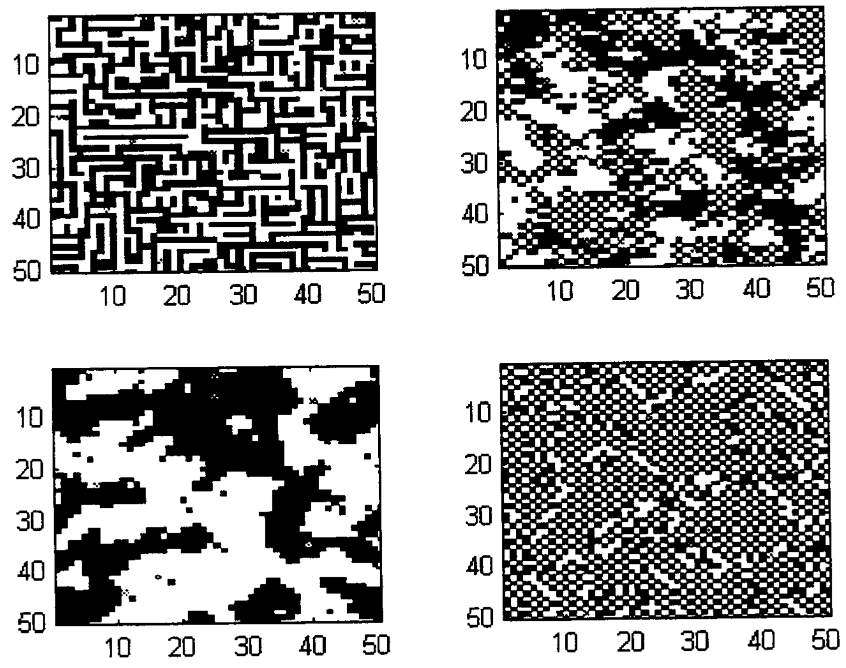

Figure 3.4 Resultant 50X50 textures for different Markov parameters.

2. The Effects of Fixing Boundary Pixels

Since the MRF model is based on neighboring pixels affecting how a targeted pixel changes as the algorithm progresses, consider the effect if some pixels are not allowed to change. In this section the boundary pixels are set to either black or white and the effect on the final image is examined. Table (3.3) shows the Markov parameters used and the figures they were used in. In the figures, the upper left image is obtained with no restrictions, while the upper right image has the boundary pixels set to black, and in the lower image the boundary pixels are white.

	Markov Parameters $[\alpha\ \beta_h\ \beta_v\ \beta_m\ \beta_r]$
Figure 3.5	[0 2 2 -2 -2]
Figure 3.6	[0 -2 -2 2 2]
Figure 3.7	[-4 2 -2 2 2]
Figure 3.8	[-6 1.5 1.5 1.5 1.5]

Table 3.3 Markov parameter table

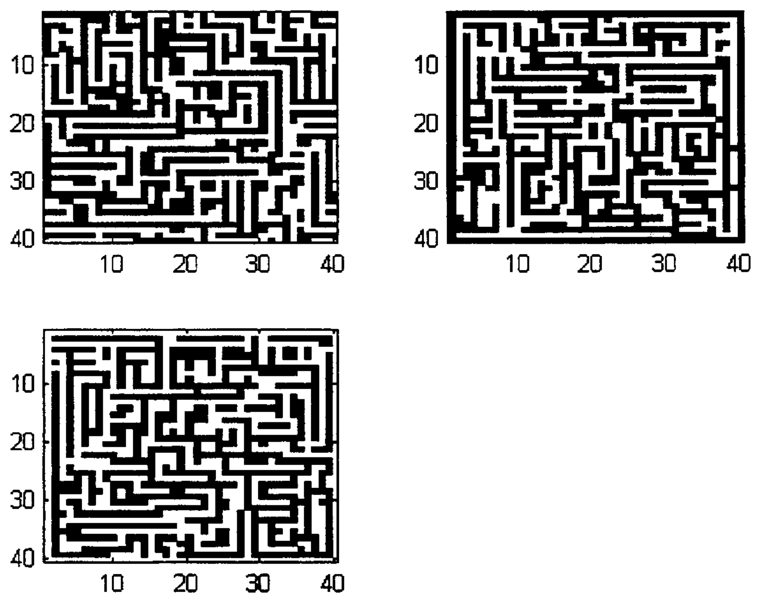

Figure 3.5 The Maze Texture with Fixed Boundaries

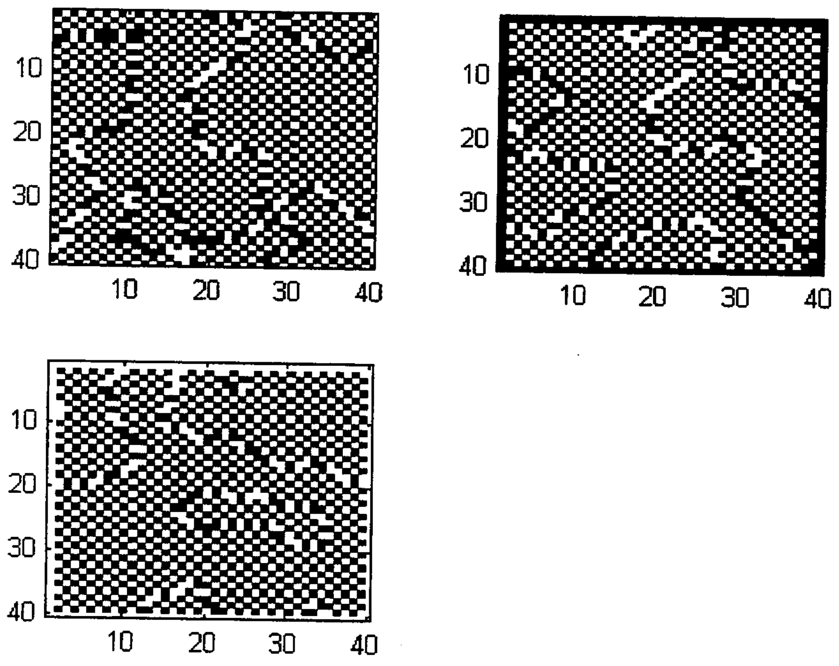

Figure 3.6 The Checkerboard Texture with Fixed Boundaries

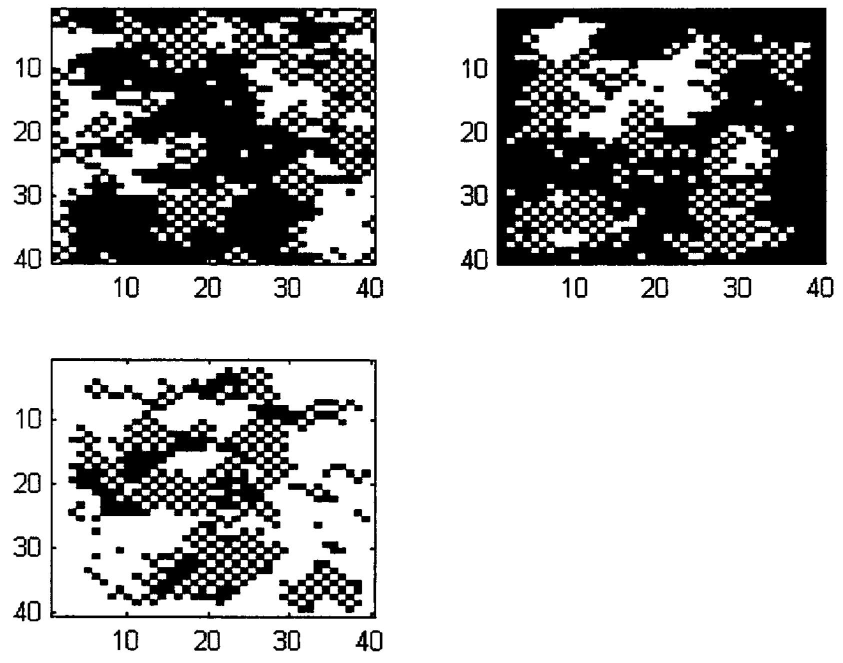

Figure 3.7 The Cloud Texture with Fixed Boundaries

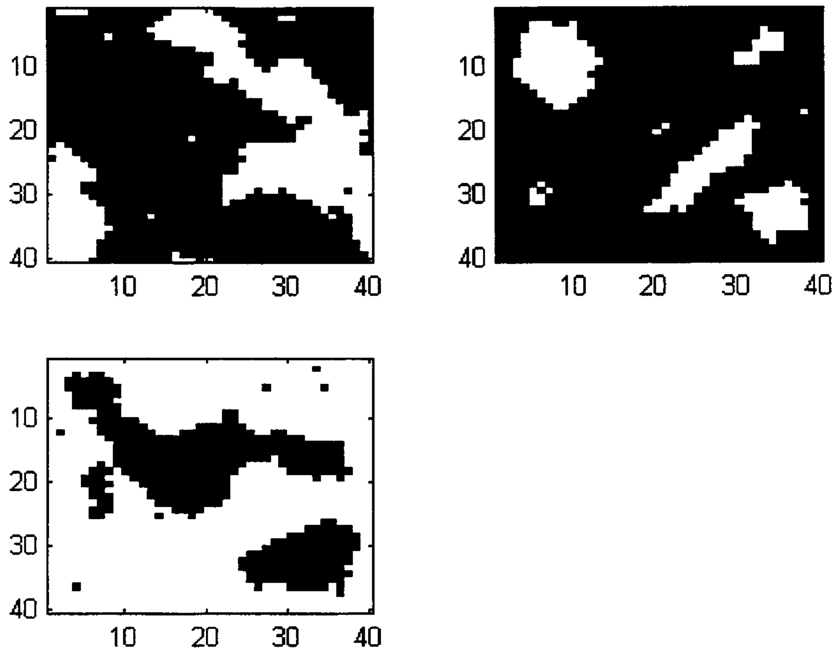

Figure 3.8 The Cow Texture with Fixed Boundaries

As is evident from the above figures, fixing the border color has little effect on the 'maze' and 'checkerboard' textures. However, it has a noticeable effect on the 'cow' and 'cloud' textures. This varied effect is due to the clustering properties of the different textures. Fixing the boundary color actually affects the latter two textures more extensively since they have a more pronounced clustering of colors.

Next, all of the elements in the center column and the elements in the center row will be set to a fixed color. The images on the left in Figure (3.9) have the specified pixels set to white, while the images on the right have them set to black. Only the textures with the higher clustering effects are shown, since the other two textures will not be greatly affected by the fixed pixels. Clearly the effects for the 'cloud' and 'cow' textures are pronounced.

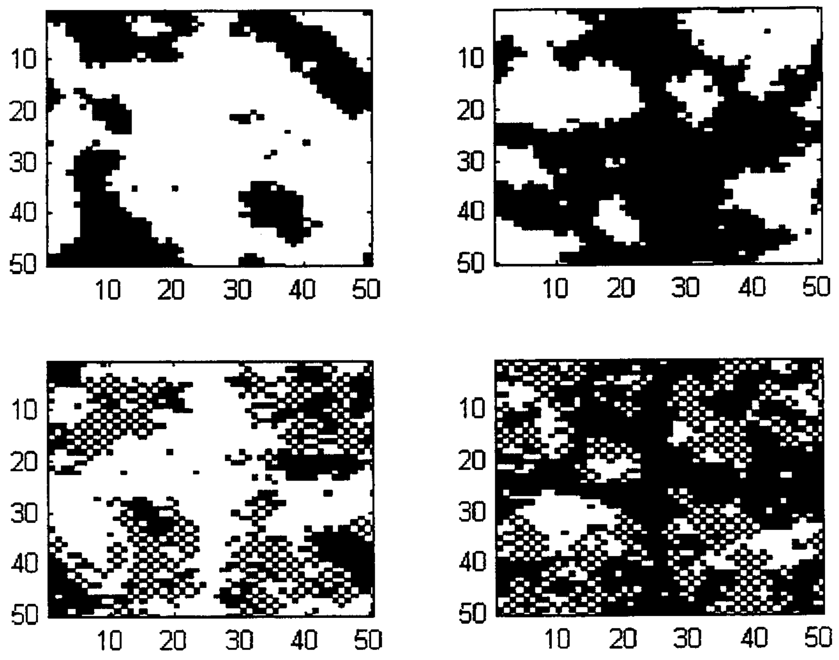

Figure 3.9 Textures with a Fixed Color Cross in the Image

For the next set of images, the left and right boundaries for the 'cow' and 'cloud' textures are set to a fixed color. The top and bottom boundaries are set to the opposite color of the side boundaries. Results are displayed in Figure (3.10). Again, the effects are striking.

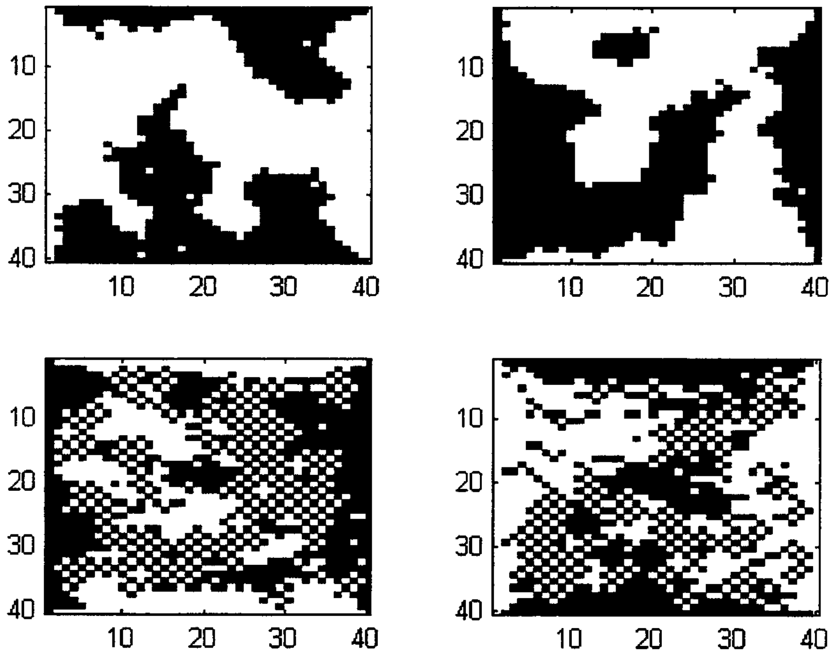

Figure 3.10 Textures with Opposite Boundary Colors Fixed

Fixing the border colors definitely effect the texture images. The textures in Figure (3.9) do show some departures from their base textures due to the excessive number of fixed pixels in the heart of the image. However, the textures in Figure (3.10) are still uniform and have no obvious departures from their normal appearance. This property allows for good color-matching at the boundaries of a texture. Thus, it would be possible to fix the boundary pixels to a desired color and then generate the appropriate texture and have it blend well with the rest of the image. An example of two different Markov textures combined by matching boundaries is shown in Figure (3.11).

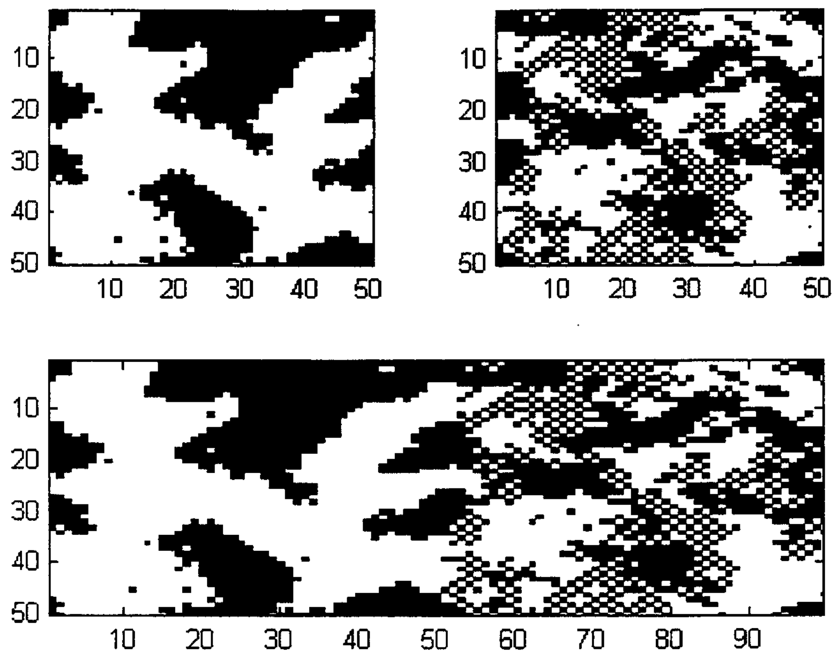

Figure 3.11 Matching Boundaries between 'Cow' and 'Cloud' Textures

IV. USING MARKOV TEXTURES IN IMAGES

Chapter III provided a structure for creating Markov Random Field textures. In order to successfully apply the Markov Random Field concepts to the transmission of tactical imagery, an algorithm for parameter estimation is essential. This Chapter addresses the process for extracting the Markov parameters from an initial image by using the "histogram" method for parameter estimation. Then the image is reconstructed using the estimated parameters and compared to the original image.

A. ESTIMATION OF MRF TEXTURE PARAMETERS

The method of parameter estimation used in this thesis was proposed by Derin and Elliot [Ref. 2]. Other methods such as the maximum likelihood estimator are discussed in [Ref. 7]. The method employed basically consists of taking a sample texture and 'histogramming' it followed by a least squares estimation of the parameters. In order to estimate the parameters in a Markov Random Field, a way must be found to sample the image and derive a useable method to estimate the five Markov parameters. A second-order neighborhood binary system is explained, although higher-order neighborhood systems can be derived using similar ideas.

1. Method of Histograms

The first step in the histogramming process is to consider a location (i,j) and its corresponding neighborhood η_{ij}. Now let $c=x_{ij}$ and let ϕ_t be the local interaction vector constructed from the neighboring values of c as shown in Figure (4.1).

m	u	w
v	c	v'
w'	u'	m'

Figure 4.1 Neighborhood arrangement

The conditional probabilities used are:

$$P\{c=1 \mid \text{values of its neighbors } \eta_{ij}\} = \frac{e^{\eta_t}}{1+e^{\eta_t}} = P_1 \qquad (4.1)$$

$$P\{c=0 \mid \text{values of its neighbors } \eta_{ij}\} = \frac{1}{1+e^{\eta_t}} = P_2 \qquad (4.2)$$

$$\text{where } \eta_t = \alpha + \beta_h(v+v') + \beta_v(u+u') + \beta_m(m+m') + \beta_r(w+w') \qquad (4.3)$$

$$\text{So } \phi_t = \begin{bmatrix} 1 \\ v+v' \\ u+u' \\ m+m' \\ w+w' \end{bmatrix}$$

$$\text{and} \qquad \phi_t^T \beta = \eta_t \qquad (4.4)$$

$$\text{therefore: } \beta^T = [\alpha \ \beta_h \ \beta_v \ \beta_m \ \beta_r] \qquad (4.5)$$

Now the relationships between the local interaction vector, the conditional probability distribution, the neighborhood and the Markov parameters have been defined. The next step is to find a method of estimating the parameters. The case when c=1 is considered first. Start with equations (4.1) and (4.2), since either c=0 or c=1 the sum of

P_1 and P_2 must equal one. Let $P_1 = P$ and $P_2 = 1\text{-}P$ and divide P_1 by P_2 to get equation (4.5).

$$\frac{P}{1-P} = \frac{\dfrac{e^{\eta_t}}{1+e^{\eta_t}}}{\dfrac{1}{1+e^{\eta_t}}} = e^{\eta_t} \tag{4.6}$$

Taking the natural log of both sides of equation (4.6) yields

$\ln\left(\frac{P}{1-P}\right) = \eta_t$ but we already know from the form of equation (4.4) that $\eta_t = \phi_t^T \beta$ so we substitute to obtain equation (4.7).

$$\ln\left(\frac{P}{1-P}\right) = \phi_t^T \beta \tag{4.7}$$

The value of the first entry in the local interaction vector, α, is set equal to 1. The other values in ϕ_t are the summations of pairwise neighbor pixels. The only possible values for these sums are 0, 1, or 2. Therefore the only possible values that ϕ_t can take on are:

$$\phi_t = \begin{bmatrix} 1 \\ 0 \text{ or } 1 \text{ or } 2 \\ 0 \text{ or } 1 \text{ or } 2 \\ 0 \text{ or } 1 \text{ or } 2 \\ 0 \text{ or } 1 \text{ or } 2 \end{bmatrix} = \begin{bmatrix} 1 \\ v+v' \\ u+u' \\ m+m' \\ w+w' \end{bmatrix} \tag{4.8}$$

The resulting probability equation is given as:

$P\{c=1 \mid \text{values of its neighbors } \eta_{ij}\} = P\{c=1 \mid \phi_t\}$

Since there are eight neighbors, there are 2^8 possible neighborhood configurations. However, there are only 81= (1*3*3*3*3) possible codings of vectors for ϕ_t. The next step is to index these vectors. Because the only numbers used are 0, 1, and 2 a base three numbering system is a logical choice. In this ternary numbering system for each possible local interaction vector, $\phi_t^T = [1 \text{ a b c d}]$, the first entry is 1, the second entry (a) corresponds to 3^3, the third (b) corresponds to 3^2 and so on. Then the base ten representation of any ϕ_t is equal to: $1 + a*3^3 + b*3^2 + c*3^1 + d*3^0$. Now each vector is represented by a base 10 number, called its index, and all of the 81 possible local interaction vectors can now be arranged in an 81 X 5 matrix.

The next step is to estimate the probability of each different index as well as the associated probability that c=1 or c=0. Only the case of c=1 will be discussed, since the probability that c=0 is just one minus the probability that c=1. A running count of which index is associated with each of the individual pixels for each subset C_k is kept. In order to obtain a good estimated probability for each index, all that is necessary is to divide the number of times a particular index occurs by the total number of entries surveyed. For example, take ϕ_t= [1 2 2 2 1], the possible neighborhood configurations for this local interaction vector are:

1	1	1
1	1	1
0	1	1

or

1	1	0
1	1	1
1	1	1

Every time either of the above two neighborhoods is detected, one more occurrence of the index of 41= $1 + 2{*}3^3 + 2{*}3^2 + 2{*}3^1 + 1{*}3^0$ is recorded for c=1. Then to obtain the approximate probability that a site of index of 41 has a value of c=1, simply divide the number of occurrences recorded by the total number of sites surveyed with c=1.

Once an estimated probability is found for each index, the next step is to solve the equation:

$$\ln\left(\frac{P}{1-P}\right) = \phi_t^T * \beta \qquad (4.9)$$

Since there are 81 different ϕ_t vectors , there are 81 different P values. The next step is to place the 81 different P values in a useful format. The desired parameter is the left hand side of equation (4.9). So let $n_i = \ln\left(\frac{P_i}{1-P_i}\right)$ and let n be the vector containing all of the n_i entries. We also want to use the transposed values of the local interaction vectors, so a new 81 X 5 matrix M is established. Where $M=\begin{bmatrix} 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,1 \\ .\,.\,.\,.\,. \\ .\,.\,.\,.\,. \\ 1\,2\,2\,2\,2 \end{bmatrix}$

This only leaves one last equation to solve to determine the Markov parameters, equation (4.10):

$$M * \begin{bmatrix} \alpha \\ \beta_h \\ \beta_v \\ \beta_m \\ \beta_r \end{bmatrix} = n \qquad (4.10)$$

This system is an overdetermined linear system, which can be solved by a least squares method to determine the desired approximation for the Markov parameters.

B. PARAMETER ESTIMATION ALGORITHM

1. The Matlab Parameter Estimation Algorithm

The following matlab code will estimate the Markov parameters for a binary image:

```
function [p] = derin(X)

%       This function identifies the parameters in a second-order Markov Random
 %      field ala Derin and Elliot.
%       Copyright 1996 by Carlos F. Borges.  All rights reserved
%       Build the matrix
M = zeros (81,5);                  (1)
for a =0:2
  for b=0:2
    for c=0:2
      for d=0:2
        i = 27*a + 9*b + 3*c + d + 1;
        M(i,:) = [1 a b c d];
      end
    end
  end
end
%       Create a histogram of the data.  For each point on the lattice determine the
%       neighborhood identifier (or index) then increment the "indcount" for that index
%       and add the i,j value to the "indsum" variable for that index
indsum = zeros(81,1); indcount = zeros(81,1);            (2)
[n m] = size(X);
for i=1:n
  for j=1:m
        % Adjust for the toroidal topology
        if i==1 id=n; else id= i-1; end      % Takes care of i==1.
        if i==n iu=1; else iu= i+1; end      % Takes care of i==n.
        if j==1 jl=m; else jl =j-1; end      % Takes care of j==1.
        if j==m jr=1; else jr=j+1; end       % Takes care of j==m.
        % Compute the index value
        a =X(i,jl) + X(i,jr);
        b =X(iu,j) + X(id,j);
        c =X(iu,jl) + X(id,jr);
        d =X(id,jl) + X(iu,jr);
        index = 27*a + 9*b + 3*c + d +1;
        % Update the count and the sum for the current index.
        indcount(index) = indcount(index) + 1;
        if (X(i,j)==1)
          indsum(index) = indsum(index) + 1;
```

```
            end
          end
end
%         Now remove all indexes which have indsum=0 since these will lead to an estimated (3)
%         marginal probability of zero which is not allowed
good= find(indsum);
indsum= indsum(good);
indcount= indcount(good);
M = M(good,:);
%         Now remove all indexes which have indcout-indsum==0 since these will lead to an
%         estimated marginal probability of 1 which is not allowed
good = find(indcount-indsum);
indsum= indsum(good);
indcount = indcount(good);
M =M(good,:);
%         Estimate the marginal probabilities for the remaining indexes   (4)
mpdf = indsum./indcount;
%         Transform the marginal probabilities to local interaction sums.
nu = log(mpdf./(ones(size(mpdf))-mpdf));
%         Compute the least squares estimate of the MRF parameters.
p = M\ nu;
```

2. Explanation of the Parameter Estimation Algorithm

As in the previous algorithm explanation, the explanation of the different annotated steps begins at the number and continues until the next section is designated.

a. Section(1)

Initially the 81 X 5 matrix M is set to have 81 rows and 5 columns of zeroes. The four concentric 'for' loops establish the desired pattern of the M matrix. The first column is a column of all ones. A pattern is created in which the last four entries in the first row will correspond to the base three number for one, in the second row they correspond to the base three number for two, and so on as shown below:

$$M = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,1 \\ 1\,0\,0\,0\,2 \\ \cdot\;\cdot\;\cdot\;\cdot\;\cdot \\ \cdot\;\cdot\;\cdot\;\cdot\;\cdot \\ 1\,2\,2\,2\,2 \end{bmatrix} = \begin{bmatrix} \text{1 followed by base3 for zero} \\ \text{1 followed by base3 for one} \\ \text{1 followed by base3 for two} \\ . \\ . \\ \text{1 followed by base3 for 80} \end{bmatrix}$$

b. Section (2)

In this section we initialize the other matrices which are needed. The *indsum* matrix and the *indcount* matrix are both created as 81 X 1 matrices of zeroes. First the image is basically wrapped around on itself as a torus so that all entries have a complete neighborhood around them. Then the terms corresponding to the neighbor value sums (as seen in Figure(4.1) and equation (4.3)) are created. The variable a is set equal to the sum of the pixel values for the horizontal neighbors, b is set equal to the sum of the vertical neighbors, c is equal to the sum of the main diagonal neighbors and d is set to the sum of the pixel values of the reverse diagonal neighbors.

The *index* is the base 10 representation of the base 3 number described above. Next the running counts are updated. The *indcount(i)* represents the total number of times that a pixel of index i is seen. The *indsum(i)* represents the number of times that a pixel value of one is recorded for index i.

c. Section(3)

In this section the theory used for the algorithm must be applied. Probabilities of zero or one cannot appear since they would cause an inconsistent result for $\ln\left(\frac{P}{1-P}\right)$, since division by zero and ln(0) are undefined. First we handle the probabilities equal to zero. The matlab command *find* is used in this section, *find(indsum)* returns the indices of the vector *indsum* that are non-zero. Then a new *indsum* vector is created which only contains the non-zero entries. Now the *indcount* vector is reassigned to only include the same indices as were found to be non-zero for the *indsum* vector. The matrix M must now be reassigned to contain only the rows which correspond to local interaction vectors which are actually are found in the image.

Now all of the entries whose probabilities are equal to one must be removed from our program. These occur when *indcount* is equal to *indsum* for any index. Again the *find* command is used, but this time it is used on the difference between the *indcount* and the *indsum*. If the difference is equal to zero, then a probability of one exists. The same general method used in the first paragraph in this section is used to update *indcount*, *indsum*, and the M matrix to remove all equations (or rows) which yield a probability of one.

d. Section(4)

In this section we actually calculate the probabilities of the local interaction vectors. The remaining entries in the *indsum* and *indcount* vectors are used to determine the probabilities of the remaining local interaction vectors in the M matrix. The matrix *mpdf* is created which is the matrix of the marginal probabilities obtained by dividing each associated *indsum* value by its corresponding *indcount* value. Now we have the associated probabilities, but what we really need is for each ϕ_t is $\ln\left(\frac{P_i}{1-P_i}\right)$. So now we set the *nu* vector equal to that value. This only leaves the last step in the algorithm in which the values of the Markov parameters are solved for by finding the least-squares solution *p* to the equation M*p* = *nu.*

3. Two Methods for Solving the Matrix Equation

The algorithm shown in section B of this chapter solves the final matrix equation ($Mp = \eta$) by using a least squares method. The method used employs the QR decomposition of M and then solves the equation using the QR method. Next two ways of solving this overdetermined matrix equation are examined. The first is the normal equation approach and the second utilizes the QR decomposition. Usually an overdetermined system has no exact solution. So we strive to minimize the difference between the least squares solution and the vector on the right side of the equation. The objective is to minimize $\|Mp - \eta\|_p$ for p=2 which is the least squares problem.

a. Normal Equations Approach

Since M has full column rank, there is a unique least squares solution (p_{ls}) which solves: $M^T Mp_{ls} = M^T \eta$. The solution process for the least-squares normal problem has 3 steps. In the first step the lower triangular portion of $C=M^TM$ is computed and we set $d=M^T \eta$. In the second step a Choleski factorization of C is performed where G is determined so that $GG^T=C$. In the last step Gy = d is solved for y, and then we solve $G^Tp_{ls} = y$ for p_{ls}.

b. QR Decomposition Method

If the QR method is used, there are 2 basic steps. First the matrix M is factored into matrices Q and R such that QR = M, where Q is an orthogonal matrix and R is upper triangular. Now each side of the equations is pre-multiplied by Q^T to get $Q^TQRp=Q^T\eta$, and since Q is orthogonal, Q^TQ is just the identity matrix. Thus, we must solve $Rp=Q^T\eta$. Since R is an upper-triangular matrix, this system can be solved very easily by back substitution.

c. Comparing the Two Methods

We must decide which method to use to solve for the desired Markov parameters. In order to compare the two methods, a few technical terms must be introduced. The first is the condition number (κ) of the matrix in use. The second degree condition number is related to the matrix M as follows:

$$\kappa_2(M) = \|M\|_2 \, \|M^{-1}\|_2$$

The other required term is the least squares error term ρ_{ls} which is defined below:

$$\rho_{ls} = \|Mp_{ls} - \eta\|_2$$

The two different methods are discussed in [Ref. 8]. We know that if ρ_{ls} is small and κ_2 (M) is large, then the method of normal equations will usually render a least squares solution that is less accurate than the stable QR approach. If the system is an ill-conditioned system, then both methods are likely to be unstable. In our problem the condition number of M is relatively small, and we know that the system is not ill-conditioned so our theoretical choice is the more stable QR approach.

A starting image is created using the texture generation algorithm in Chapter III. Then the two methods above are used to estimate the Markov parameters of those images. The original parameters as well as the difference between the two approximation methods results are shown in Table (4.1).

Actual Markov parameters	Difference between Normal equation and QR estimates
[0 2 2 -2 -2]	[-.1554 .0666 -.0222 0 .1110] X 10^{-14}
[-4 2 -2 2 2]	[.1776 -.0222 .0444 .0222 -.1776] X 10^{-14}
[-6 1.5 1.5 1.5 1.5]	[-.3553 .1332 -.0666 .0888 .1221] X 10^{-14}
[0 -2 -2 2 2]	[-.1155 0 -.0089 .0755 .0311] X 10^{-13}

Table 4.1 Estimated Parameter Differences

As evident from the table above, both methods provide approximately the same answers for all of the parameters. In addition, for the Normal approach to work the final M matrix must be positive definite. A final M which is positive definite is not guaranteed. The method of choice will be the QR method for the above reasons and for some other advantages to be mentioned in the next section.

d. Improving on the QR method

Now that we have decided to exclusively use the QR approach for solving the final matrix equation, the next subject is to see if we can make any improvements on that method to make the algorithm more efficient. We know that the rows in the final matrix M are increasing base three numbers. In fact, when we perform the QR decomposition of M, R is an upper-triangular matrix whose only nonzero entries are its first row and main diagonal. Therefore, there are only nine nonzero terms in the R matrix for the second-order neighborhood in use. So the only portions of Q and R which are used are the first five columns of Q and the top five rows and columns of R. This property can be used to greatly simplify the final equation which must be solved for the estimated Markov parameters.

To illustrate how the structure of the Q and R matrices can be taken advantage of to save computation cycles, the first-order neighborhood model is used. In the first-order neighborhood when the full M matrix undergoes a QR decomposition, the resultant R matrix is an upper-triangular matrix whose only nonzero entries are in the first three rows. Therefore the only portion of Q which matter are the first three columns. These portions of the Q and R matrices are shown below:

$$Q = \begin{bmatrix} -.3333 & -.4082 & .4082 & \cdots \\ -.3333 & -.4082 & 0 & \cdots \\ -.3333 & -.4082 & -.4082 & \cdots \\ -.3333 & 0 & .4082 & \cdots \\ -.3333 & 0 & 0 & \cdots \\ -.3333 & 0 & -.4082 & \cdots \\ -.3333 & .4082 & .4082 & \cdots \\ -.3333 & .4082 & 0 & \cdots \\ -.3333 & .4082 & -.4082 & \cdots \end{bmatrix} \qquad R = \begin{bmatrix} -3 & -3 & -3 \\ 0 & 2.4495 & 0 \\ 0 & 0 & -2.4495 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

Looking at the first three columns of the Q matrix, an easily recognizable pattern exists. There are only four different values in that portion of the matrix. All of the entries on the first column are the same (-.3333). The two other columns only contain three different values (-.4082, 0, .4082). The second column repeats each value three times before switching to the next value, and the third column does not repeat values.

To simplify this pattern all of the entries in the first three columns of Q are divided by the leading entry in that column, this produces a new matrix **q** consisting of values 1, 0, and -1. q is depicted below:

$$q = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & -1 \\ 1 & -1 & 1 \\ 1 & -1 & 0 \\ 1 & -1 & -1 \end{bmatrix}$$

The next step is to produce matrix **D**, it is a diagonal matrix whose diagonal terms are the leading terms in the first three columns of the Q matrix. It is shown below:

$$D = \begin{bmatrix} -.3333 & 0 & 0 \\ 0 & -.4082 & 0 \\ 0 & 0 & .4082 \end{bmatrix}$$

Now the matrix q is multiplied by D, and the result is the original matrix Q. So the entire matrix Q can be produced knowing just the first three entries in the first row of Q. In solving $Mp = nu$, qD is substituted for Q. The resulting solution for p is: $p = R^{-1}Q^T\eta = R^{-1}D^Tq^T = R^{-1}Dq^T$. By solving for p using this method, instead of using nine multiplications and 27 adds, we only use three multiplications and 14 adds. This is a substantial savings in compute time and will greatly speed up the process of solving for the Markov parameters.

This same approach can be used for the second-order neighborhood model. Then the estimates for the Markov parameters are given by the following formulas obtained from [Ref.9].

$$\beta_4 = \frac{1}{54}\sum_{i=0}^{26}\left(\nu_{3i+3} - \nu_{3i+1}\right)$$

$$\beta_3 = \frac{1}{54}\sum_{i=0}^{8}\sum_{j=9i+1}^{9i+3}\left(v_{j+6} - v_j\right)$$

$$\beta_2 = \frac{1}{54}\sum_{i=0}^{2}\sum_{j=27i+1}^{27i+9}\left(v_{j+18} - v_j\right)$$

$$\beta_1 = \frac{1}{54}\sum_{j=1}^{27}\left(v_{j+54} - v_j\right)$$

$$\alpha = \frac{1}{81}\sum_{i=1}^{81} v_i - \beta_1 - \beta_2 - \beta_3 - \beta_4$$

Using this method it is possible to solve for all of the Markov parameters with only 160 adds and 5 divides. This is a substantial savings over the 5x81=405 multiplies and 5x80=400 adds which are required in the basic algorithm.

C. INITIAL AND FINAL IMAGES USING ESTIMATED PARAMETERS

In this section several binary representations of textures will be presented. The preceding parameter estimation algorithm will be used on the images to determine the associated Markov parameters. Then the estimated parameters will be used to reconstruct the image. The initial and final images will then be compared to determine the effectiveness and applicability of this method.

In Figure (4.2) and Figure (4.3), the images on the left are created using the texture generation algorithm in Chapter III using Markov parameters as shown in Table (4.2). The images on the right are created with the same algorithm, but using the Markov parameters obtained from the left image by the parameter estimation algorithm in this chapter.

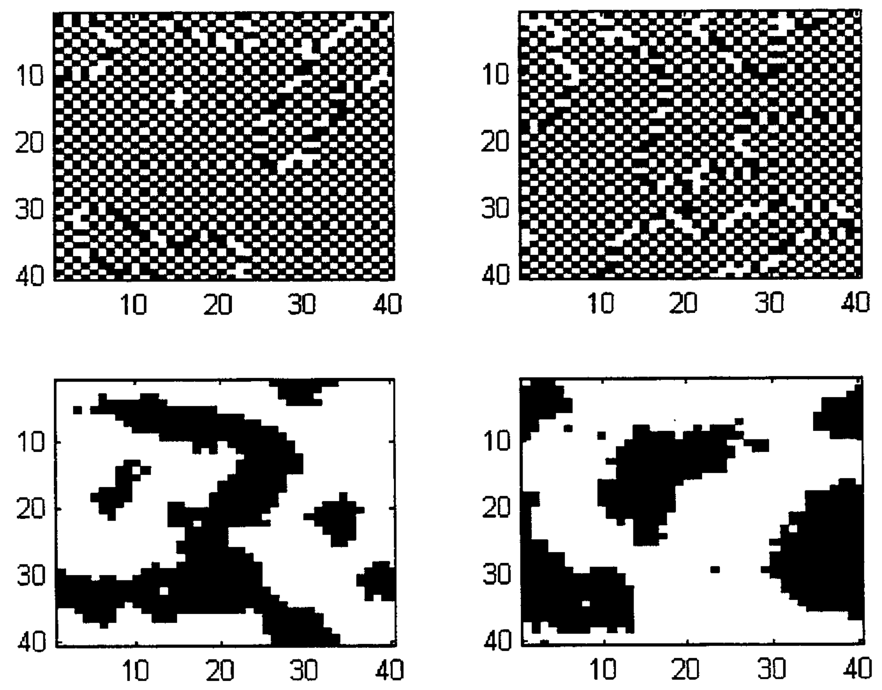

Figure 4.2 Initial and Final Binary Images Using Estimated Parameters

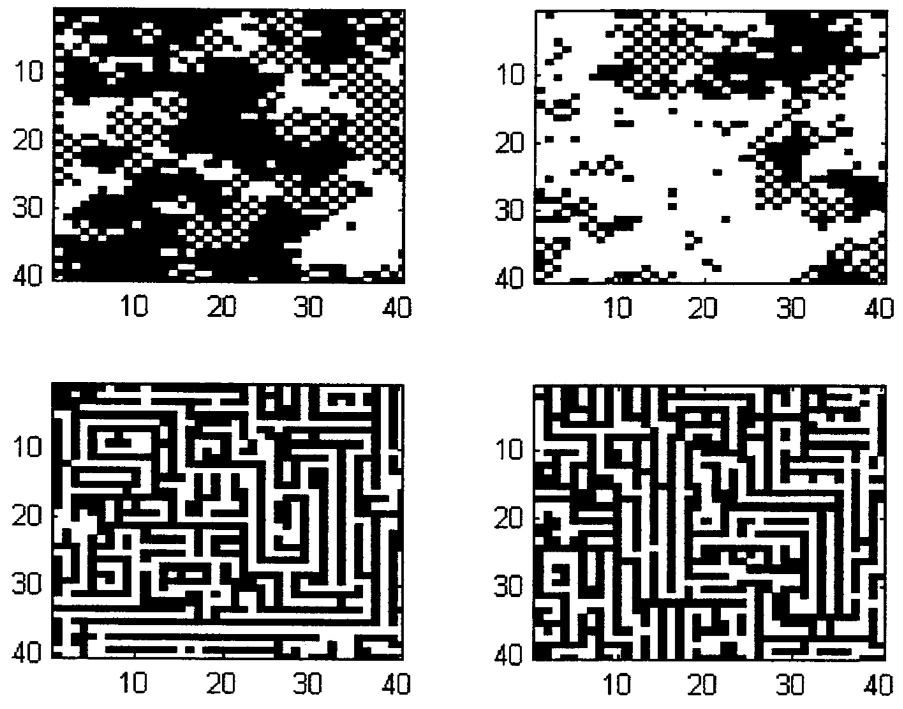

Figure 4.3 Initial and Final Binary Images Using Estimated Parameters

	Original Parameters (left side)	Estimated Parameters (right side)
Fig 4.2 (top)	[0 -2 -2 2 2]	[.9938 -1.6115 -1.9590 1.2103 1.6611]
Fig 4.2 (bottom)	[-6 1.5 1.5 1.5 1.5]	[-5.2304 1.2418 1.2250 1.0646 1.7693]
Fig 4.3 (top)	[-4 2 -2 2 2]	[-3.7650 1.7967 -1.6377 1.9965 1.6798]
Fig 4.3 (bottom)	[0 -2 -2 2 2]	[.7604 1.2530 1.3556 -1.8454 -1.6322]

Table 4.2 Initial and Estimated Markov Parameters for Figure 4.2 and Figure 4.3

It is evident that the images on the right have an extremely similar texture to the ones on the left. Although the estimated parameters are not extremely close to the actual parameters, they are sufficiently close to approximate the same texture. This demonstrates the stability of the MRF process, both in estimation of parameters and production of textures from parameters.

V. CONCLUSIONS AND FURTHER STUDY RECOMMENDATIONS

A. CONCLUSIONS

This paper explains the use of a Markov Random Field model to create and reproduce textures in images. Binary images containing texture regions can be analyzed and reproduced very effectively using the 'histogram' method and the texture generation algorithm presented in this thesis. The ability to match the edges of different texture regions maintains the smoothness of the overall image.

When applied to tactical imagery containing textured regions, the MRF texture model is extremely useful. It can greatly reduce the amount of data transmission required and lead to substantial time savings when updating tactical images. The texture regions created are immune to steganography and could allow for timely downgrading of classified imagery. Consider a tactical image of a surface to air missile (SAM) site in a desert. Using MRF textures it would be possible to replace the region containing the SAM site with a similar texture region. After replacement there would be absolutely no evidence that the SAM site had been removed from the image, permitting downgrading.

B. RECOMMENDATIONS FOR FURTHER STUDY

The MRF models for texture generation and parameter estimation discussed in this paper are basic models. Many improvements can be made in the algorithms presented . Improving the process of estimating Markov parameters has been addressed fairly extensively, one example is [Ref. 10]. Further research in the generation of textures and the estimation of Markov parameters could prove invaluable for improving the overall applicability of MRF texture models. Many other active research areas are discussed in the following sections.

1. SAR Image Detection Algorithms

Another use of MRF models is for detection algorithms. Hidden Markov models (HMM) can be used with synthetic aperture radar for automatic target detection. The HMMs exploit the anisotropic nature of radar returns from man-made objects. Specific HMM structures can be developed to represent the target and clutter pixels based on the way their returns vary at different aspect angles. Further research could produce an HMM automatic target detection algorithm with better detection accuracy than current models while requiring at least two orders of magnitude less calculations.

2. MRF for Gray-scale Images

Only binary images were addressed in this thesis. Construction of MRF models for images with a greater diversity in color would be a good candidate for continuing research. However, substantially more complicated mathematics would be involved because of the increase in possible neighborhoods and local interaction vectors.

3. Computer Vision Applications

Markov Random Field theory provides a basis for modeling contextual constraints in visual processing and interpretation. The basic objective in machine vision is to give the machine a sense of vision, that is to use visual sensors for such tasks as detection and recognition of objects, tracking of objects, navigation, and perhaps reasoning. In computer vision the perspective is to model an image by a Markov random field and then use a Bayesian approach for estimating early vision attributes of the image. Future study in formulation of MRF vision models, MRF parameter estimation, and optimization algorithms is an exciting prospect.

4. Automatic Speech Recognition Algorithms

Hidden Markov Models (HMM) are statistical models that are widely used in automatic speech recognition and molecular biology. The parameters (emission and transition probabilities) of a HMM can be estimated from a set of examples by using a maximum likelihood training algorithm. Recently, there has been a widespread interest in combining neural networks and HMM for speech recognition. If neural networks are used to estimate probabilities in HMM then it is possible to estimate the weights in the neural network and the parameters in the HMM at the same time using a gradient descent algorithm. Further analysis and development of algorithms for training combined neural network and HMM models is needed.

APPENDIX

The Hammersley-Clifford theorem establishes the equivalence of a MRF's local properties to the global properties of a GRF. The theorem states that **F** is an MRF on **S** with respect to η if and only if **F** is a GRF on **S** with respect to η. The proof of this theorem follows in two parts.

1. Proof That a Gibbs Random Field is a Markov Random Field [Ref. 11]

Let **P(f)** be a Gibbs distribution on **S** with respect to the neighborhood system η. Consider the conditional probability $P(f_i \mid f_{s-\{i\}}) = \dfrac{P(f_i, f_{s-\{i\}})}{P(f_{s-\{i\}})} = \dfrac{P(f)}{\sum f^{s}_{i\in C} P(f')}$ where

$f' = \{f_1, \ldots, f_{i-1}, f'_i, \ldots, f_m\}$ is any configuration which agrees with **f** at all sites except possibly **i**. Writing $P(f) = Z^{-1} e^{-\sum_{c\in C} V_c(f)}$ out gives the following formula:

$$P(f_i \mid f_{s-\{i\}}) = \frac{e^{-\sum_{c\in C} V_c(f)}}{\sum_{f'_i} e^{-\sum_{c\in C} V_c(f')}}$$

Next divide **C** into two sets **A** and **B** with **A** consisting of cliques containing **i** and **B** cliques not containing **i**. The previous equation can now be written as:

$$P(f_i \mid f_{s-\{i\}}) = \frac{[e^{-\sum_{c\in A} V_c(f)}][e^{-\sum_{c\in B} V_v(f)}]}{\sum_{f'_i}\{[e^{-\sum_{c\in A} V_c(f')}][e^{-\sum_{c\in B} V_c(f')}]\}}$$

Because $V_c(f)=V_c(f')$ for any clique that does not contain **i**, $e^{-\sum_{c\in B} V_c(f)}$ cancels from both the numerator and denominator. Therefore, the probability only depends on the potentials of the cliques which contain **i**, which are the neighbors of **i**. The resultant probability function is:

$$P(f_i \mid f_{s-\{i\}}) = \frac{e^{-\sum_{c\in A} V_c(f)}}{\sum f'_i\, e^{-\sum_{c\in A} V_c(f')}}$$

This proves that a Gibbs random field is a Markov random field.

2. Proof that a Markov Random Field is a Gibbs Random Field [Ref. 12]

This proof is in two parts:

a. Part 1

We must show there exists a U(x) such that $P(X=x) = \frac{1}{Z}e^{-U(x)}$.

Since $P(X=0) \neq 0$, define U(x) as:

$$U(x) = -\ln\left\{\frac{P(X=x)}{P(X=0)}\right\} \qquad \textbf{(1)}$$

Now take the negative of both sides of the equation and make them powers of e to obtain: $\frac{P(X=x)}{P(X=0)} = e^{-U(x)}$ which can be rewritten as $P(X=x) = P(X=0)\, e^{-U(x)}$,

with $Z = \frac{1}{P(X=0)}$.

b. Part 2

We must establish that the MRF conditions imply that $G_{...}(---)$ is zero if its arguments do not belong to the same clique. Which will in turn allow U(x) to be expressed as $\sum_c V_c(x)$. It must be shown that $G_{...}(---)$ becomes zero if its arguments do not belong to the same clique. For the first step in this portion of the proof, let $\overline{x}^{m,n} = \left[x_{1,1},\cdots,x_{1,n},\cdots,x_{m,n-1},\ 0,\ x_{m,n+1},\cdots,x_{N,1},\cdots,x_{N,N}\right]^T$. Next, let $\overline{x}^{m,n}$ be a realization of $\overline{X^{m,n}}$. Use equation (1) to get:

$$U(x) - U(\overline{x}^{m,n}) = -\ln\left\{\frac{P(X=x)}{P(X=0)}\right\} -\ln\left\{\frac{P(X=\overline{x}^{m,n})}{P(X=0)}\right\} = -\ln\left\{\frac{P(X=x)}{P(X=\overline{x}^{m,n})}\right\}.$$

Now raise **e** to the power of both sides of the equation to obtain:

$$e^{-(U(x)-U(\overline{x}^{m,n}))} = \frac{P(X=x)}{P(X=\overline{x}^{m,n})} = \frac{P[X_{m,n} = x_{m,n} \mid X_{k,l} = x_{k,l} \text{ for all } (k,l) \in S,\ (k,l) \neq (m,n)]}{P[X_{m,n} = 0 \mid X_{k,l} = x_{k,l} \text{ for all } (k,l) \in S,\ (k,l) \neq (m,n)]}$$

By Markovian properties, the right hand side of the previous equation only depends upon the neighborhood of (m,n), and the equation can be rewritten as:

$$e^{-(U(x)-U(\overline{x}^{m,n}))} = \frac{P(X_{m,n} = x_{m,n} | X_{k,l} = x_{k,l} \text{ for } (k,l) \in \eta_{m,n})}{P(X_{m,n} = 0 | X_{k,l} = x_{k,l} \text{ for } (k,l) \in \eta_{m,n})} \qquad \textbf{(2)}$$

Next the Reed-Muller expansion of the general form of U(x) is used to derive:

$$U(x) = \sum_{(i,j)=(1,1)}^{(N,N)} x_{i,j} G_{i,j}(x_{i,j}) + \sum_{(i,j)=(1,1)}^{(N,N-1)} \sum_{(k,l)=(i,j+1)}^{(N,N)} x_{i,j} x_{k,l} G_{i,j;k,l}(x_{i,j}, x_{k,l}) + \sum_{(i,j)=(1,1)}^{(N,N-2)} \sum_{(k,l)=(i,j+1)}^{(N,N-1)} \sum_{(r,s)=(k,l+1)}^{(N,N)} \cdots +$$
$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$
$$+ \; x_{1,1} \cdots x_{N,1} \cdots x_{N,N} G_{1,1;\cdots;N,1;\cdots;N,N}(x_{1,1} \cdots, x_{N,1} \cdots x_{N,N})$$

Without loss of generality any (m,n) can be considered, we will use (m,n) = (1,1). Now $U(x)$ - $U(\overline{x}^{1,1})$ can be simplified based on the following three facts:

1. Since $x_{1,1} = 0$, in $U(\overline{x}^{1,1})$, all terms corresponding to $x_{1,1}$ will give a zero contribution.
2. All terms not involving $x_{1,1}$ in U(x) will cancel all terms not involving $x_{1,1}$ in $U(\overline{x}^{1,1})$.
3. The terms remaining will involve $x_{1,1}$ and will come from U(x).

The simplified equation for $U(x)$ - $U(\overline{x}^{1,1})$ is:

$U(x)$ - $U(\overline{x}^{1,1})$ =

$$x_{1,1} G_{1,1}(x_{1,1}) + \sum_{(k,l)=(1,2)}^{(N,N)} x_{1,1} x_{k,l} G_{i,j;k,l}(x_{1,1}, x_{k,l}) + x_{1,1} \sum_{(k,l)=(1,2)}^{(1,N-1)} \sum_{(r,s)=(k,l+1)}^{(N,N)} x_{k,l} x_{r,s} G_{1,1;k,l;r,s}(x_{1,1}, x_{k,l}, x_{r,s})$$
$$+ \;\; \vdots$$
$$+ \; x_{1,1} \cdots x_{N,1} \cdots x_{N,N} G_{1,1;\cdots;1,N;\cdots;N,N}(x_{1,1}, \cdots, x_{N,1}, \cdots, x_{N,N}) \qquad \textbf{(3)}$$

Now suppose that element (p,q) is not a neighbor of (1,1). Then by equation (2), the right hand side of equation (3) must be independent of $x_{p,q}$. This can be used to show that all

$G_{...}(---)$ which involve $x_{1,1}$ and $x_{p,q}$ will be zero. Therefore, choose x such that $x_{k,l} = 0$, for all $(k,l) \neq (1,1)$ and $(k,l) \neq (p,q)$ to obtain $x = \left[x_{1,1}\ 0 \cdots 0\ x_{p,q}\ 0 \cdots 0\right]^T_{N^2 X1}$. Now $U(x) - U(x^{1,1}) = x_{1,1}G_{1,1}(x_{1,1}) + x_{1,1}x_{p,q}G_{1,1;p,q}(x_{1,1}, x_{p,q})$, and in order for this equation to be independent of $x_{p,q}$, $G_{1,1;p,q}(x_{1,1}, x_{p,q})$ must equal zero. By considering all sites (p,q) which are not neighbors of (1,1), we obtain: $G_{1,1;p,q}(x_{1,1}, x_{p,q}) = 0$ for all $(p,q) \notin \eta_{1,1}$.

For the three node case, x is chosen such that $\{x_{k,l} = 0$ for all $(k,l) \neq (1,1)$ and $(k,l) \neq (p,q)$ and $(k,l) \neq (t,u)\}$ where $(1,1) \neq (p,q) \neq (t,u)$. Here (t,u) is an arbitrary site and an approach similar to the two node case is followed to show that $G_{1,1;p,q;t,u}(x_{1,1}, x_{p,q}, x_{t,u}) = 0$ for all $(p,q) \notin \eta_{1,1}$. Continuing in this fashion, all $G_{...}(---)$ whose arguments are not neighbors have to be zero. Therefore, $U(x) = \sum_c V_c(x)$ where V_c is determined by $G_{...}(---)$ whose arguments form a clique. This completes the proof of the Hammersley-Clifford Theorem.

LIST OF REFERENCES

1. Thompson, Scott. *Radiant Tin Compression.* Space Systems Academic Group, Naval Postgraduate School Monterey, California. June 1995.

2. Derin, Haluk and Elliott, Howard. *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, Jan 1987.

3. Jain, Anil K. *Image Coding Via a Nearest Neighbors Image Model.* IEEE Transactions on Communications, Vol. Com-23, NO.3, Mar 1975.

4. Tamura, H., Mori, S. and Yamawaki, T. *Textural Features corresponding to visual perception.* IEEE Trans. Syst., Man., Cybern., Vol SMC-8, 1978.

5. Jain, Anil K. and Nadabar, Sateesha G. *MRF Model-Based Segmentation of Range Images.* Department of Computer Science, Michigan State University East Lansing, MI 48824.

6. Gurelli, Mehmet I. and Onural, Levent. *A Parallel Algorithm for the Generation of Markov Random Field Textures.* Communication, Control, and Signal Processing. Elsevier Science Publishers B.V.,1990.

7. Cross, George R. and Jain, Anil K. *Markov Random Field Texture Models.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, NO.1, Jan 1983.

8. Golub, Gene H. and Van Loan, Charles F. *Matrix Computations.* The Johns Hopkins University Press. 1996.

9. Borges, Carlos F. *On the Estimation of Markov Random Field Parameters.* Mathematics Department, Naval Postgraduate School Monterey, California. March 1997.

10. Calder, B., Linnett, L., Clarke, S., Carmichael, D. *Improvements in MRF Parameter Estimation.* IEE E4 Colloquium on Multiresolution Image Processing, No. 1995/077, London, April 1995.

11. *Markov-Gibbs Equivalence.* http://ntuix.ntu.ac.sq/~szli/book_1/Chapter_1/node13.html

12. Desai, U. B. *Markov Random Field Models for Early Vision Problems.* Department of Electrical Engineering, Indian Institute of Technology Bombay, India. July 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center..2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library..2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Professor Hal Fredricksen..2
   Mathematics Department
   Naval Postgraduate School
   Monterey, California 93943-5002

4. Professor Carlos Borges..2
   Mathematics Department
   Naval Postgraduate School
   Monterey, California 93943-5002

5. Dr. Nicholas D. Beser..1
   Johns Hopkins University APL
   Johns Hopkins Road 23-328
   Laurel, Maryland 20723-6099

6. Lt. Christopher A. Korn...1
   475 Norris Lane
   Lake Helen, Florida 32744