RANDOM FIELD MODELING FOR INTERPRETATION AND ANALYSIS OF LAYERED DATA

by

Carey David Bunks

B.S. EECS Tufts University
(1980)

M.S. EECS Tufts University
(1980)

SUBMITTED TO THE DEPARTMENT OF
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1987

Signature of Author ____
Department of Electrical Engineering and Computer Science
December 20, 1986

Certified by _____
Alan Willsky
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Random Field Modeling for Interpretation and Analysis of Layered Data

by

Carey David Bunks

Submitted to the Department of Electrical Engineering and Computer Science on December 25, 1986 in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

In this thesis we use Markov random field models for the purpose of analyzing two-dimensional, layered, anisotropic data. We are interested in extracting dip and edge information from this data. Two model types are considered for characterizing dip. The first model type is for binary layered data. The second type is for non-binary layered data and it is based on a new class of covariance functions which we term ARC functions. The primary signal processing tools used in this thesis are Maximum Likelihood and Maximum a Posteriori estimation. Several fast algorithms are developed for dip estimation by utilizing the properties of ARC functions. These algorithms yield approximate ML estimates. Local edge models are developed which utilize prior information about dip. The local edge estimates are used as inputs to a series of global edge estimators which help to identity bed boundaries in the layered data. Finally, the signal processing techniques developed within the thesis are applied to analyzing layered data which contain features at several spatial scales.

Thesis Supervisor: Alan Willsky
Title: Professor of Electrical Engineering

Acknowledgements

What is the source of new ideas? And why do they always come when one has no clean underwear? I don't know the answer to these questions, however, I used to keep a pen and pad near my bed at night so that I could jot down ideas that would come into my head as I drifted to sleep. I discontinued this practice after discharging the entire contents of my black pen on my white sheets one night in a fit of sleep-writing. As is usual, what seemed crystal clear to me in my dream state appeared the next morning as a mixture of hieroglyphics and prophetic text written in tongues.

Regardless of what the timing or source of ideas may be it seems that the magic of ideation is strongly dependent upon the community of people in which an individual is immersed. It is the random day to day contact with one's peers which often is the compelling force which induces one to look at this idea or that. While thinking about this work there have been a great many individuals with whom, on a day to day basis, I have routinely come into contact. Many of these individuals I wish to acknowledge here.

First, there is the group of people who were my most direct contact with things technical: my fellow students with whom I spent many hours discussing mathematics and science. These discussions were often the most stimulating aspects, and in many ways, the best part of my education at MIT. I'd particularly like to recognize Larry Hegi, Apo Sezginer, Peter Doerschuk, Jim Roskind, Jerry Prince, Ahmed Tewfik, Robin Rholicek, Ramine Nikoukhah, and Ken Chou.

Second, there are those members of the MIT faculty with whom I've had close contact. I would like to thank my thesis advisor Professor Alan Willsky for his patience and guidance. Also, I appreciate the time and commitment of the members of my thesis committee, Professors Sanjoy Mitter and Tomaso Poggio. Two other individuals who afforded me great encouragement are my close friends Professor Douglas Preis of Tufts University and Dr. Michael Brown (a master of sport's car notation).

Third, there are my parents who were most important in shaping my outlook on life. They have always been supportive of my interests and they think I'm terrific (even if I *have* been educated beyond my intelligence).

Fourth, there is a group of people whom I feel deserve a lot of credit for the *quality* of my educational experience. This group is the community of Boston street performers. When the fruit of the thesis vine was bitter, when the vagaries of the computer software jungle were vicious, the street performers of Boston were there to lift the weight of failure from my shoulders. Their myriad antics and their welcome friendship was the salve which cooled my feverished brain. Consequently, I include the names of the following groups and individuals: The Shakespeare Brothers (Steve

Aveson and Al Krulick), The Locomotion Circus (Bounce, Cyrus P. Koski III, and Flip), Slaphappy (Brian O'Connor, Al Jacobs, Jeffrey Ernstoff, and Jan Kirshner), The Amazing Fantasy Jugglers (Don and Lana Reed, Rawd Holbrook, and Jim Cunniff), Leonard Solomon, Alex (King of Jesters) Feldman, Peter Sosna, Dario Pittore and Eddie G., Stan Sinberg, The Dexterity Brothers (John Kalonymous Briskin and Barry Rosenberg), The Waldo-Woodhead Show (Mark Keppel, Waldo, and Ron Labbe), David Stiefel, and the many, many others with whom it was my pleasure to associate.

Finally, I would like to acknowledge my thanks for the varied sources of funding which I have had during my studies. I am thankful to the Massachusetts Institute of Technology for its support through the Vinton Hayes fellowship during my first year. I am especially thankful for the support I have had from Schlumberger-Doll Research laboratories in Ridgefield, CT. Also, much thanks is due to the National Science Foundation which provided me support through NSF grant #ECS-8312921.

This must be the sound of one hand clapping.

Contents

List of Figures

Chapter 1

Introduction

Many problems in multi-dimensional signal processing are fundamentally different from those of one-dimensional signal processing. Often these differences have to do with the special geometries associated with the underlying models of the signal processing problem. The existence of geometric relationships and the desire in many cases to identify or estimate geometric features provide unique features to multi-dimensional signal processing problems not found in one-dimensional problems.

This thesis concentrates on a set of models and problems which address a particular class of two-dimensional data processing problems concerned with identifying specific geometric characteristics. In particular the type of data of interest to us consists of random fields which exhibit a layered structure and, therefore, display distinctive anisotropic features. A major portion of the effort expended in what follows concentrates on quantifying exactly what we mean by the phrase "layered" data. The result of this effort consists of models which have special geometric properties. Our signal processing algorithms make explicit use of the geometries of these models.

1.1 Problem Motivation

This thesis is concerned with a particular type of two-dimensional, anisotropic data. The data can be loosely described as layered in the sense that the spatial correlation is much longer in one direction than it is in the orthogonal direction. A particular

problem of geophysical signal analysis motivated this research. The basic elements of this problem are described below.

In geophysics the model of the sedimentation process consists of erosion, transport, and deposition [23]. Material that is worn away at one location is carried by air or water currents to another location and, finally, left at the new location. An example of this is the build up of sediments at the mouth of a river. The particulate is obtained from the river's origin and from other materials worn from the rivers banks or that are washed into the river by rains, etc. The types of sedimentation patterns found at the river's mouth act as a "fingerprint" of the particular process that created it. Consequently, observations of the sedimentary layers can lead to conclusions about the geology that gave rise to them.

Several problems exist, however, in trying to draw conclusions from observations of sedimentary layers. First, the layers we wish to examine are hundreds or thousands of feet deep within the earth. To perform observations of layers deep within the earth an exploration team will drill a borehole. Since coring is expensive and often ineffective due to the destruction of the core during the drilling process measurements are made in the hole after the drilling is completed.

The data with which we are concerned are high-resolution measurements of the electrical conductivity of the rock [9]. These conductivity measurements consist of attempts to pump current into the rock of the borehole wall at many discrete locations in depth and circumference. Different rock types and fluids within the rock affect the amount of current which can flow at each point. Consequently, the measurements of the rock consist of a conductivity map which can be converted, via appropriate scaling, to a gray scale and represented as a two-dimensional image. The image is two-dimensional because the borehole wall is a two-dimensional (roughly cylindrical) surface. We will assume throughout this thesis that the conductivity measurements used are from a portion of the borehole wall limited in its radial extent. This assumption allows us to treat the data as if it were obtained from a planar two-dimensional geometry as opposed to a cylindrical two-dimensional geometry.[1] If the density of measurements is high then the data will appear layered

[1] All the results of this thesis easily generalize to a cylindrical geometry.

when examining a sedimentary region. (See Figure 1.1).

The second problem that arises in interpreting data for the purpose of determining geological processes is the quantity of data that exists. For a hole that approaches a mile in depth with measurements which are taken at the resolution of millimeters [9] we have a huge data processing problem. It is desirable to try and automate the signal analysis process as much as possible. This automation would allow human interpreters to make more effective use of their time, allowing them to work on identifying higher-level features.

One of the features with which expert interpreters work is the dip [23], [4], [11] of sedimentary beds. In a region of sedimentary layering the dip is a vector quantity defined to be the angle of inclination and the angle of azimuth of the layers. A difficulty with determining the dip in a region of bedding is that beds are often hard to identify. Qualitatively, beds are characterized as structures which have a direction of high correlation while in the roughly orthogonal direction they are relatively uncorrelated. Conceptually, the dip is defined by the direction of lowest correlation. Consequently, when a layered region consists of beds with planar boundaries the dip points orthogonal to the boundaries (Figure 1.2). However, in real sedimentary data, beds often do not have planar boundaries. In the vicinity of a non-planar bed boundary our concept of dip is inadequate to model the orientation of the beds. Consequently, identifying bed boundaries is an additional and important part of characterizing the geometry of beds. In our work we obtain estimates both of dip, defined in terms of the correlation structure, and bed boundaries.

Spatial patterns of dip (or bed orientations) provide interpreters with a great deal of information which they use to identify different geological structures [23], [6]. Some simplified examples of geological structures and their dip patterns are illustrated[2] in Figure 1.4. The left side of Figure 1.4a illustrates an ancient marine sand bar which has been covered over by sedimentary layers. As more and more layers cover the sand bar less of the sand bar's hump is visible in the layers.

The right side of Figure 1.4a illustrates the pattern of dips as a function of depth seen from the borehole illustrated in the left side of the figure. The representation

[2]Illustrations taken from [23] pg. 14.

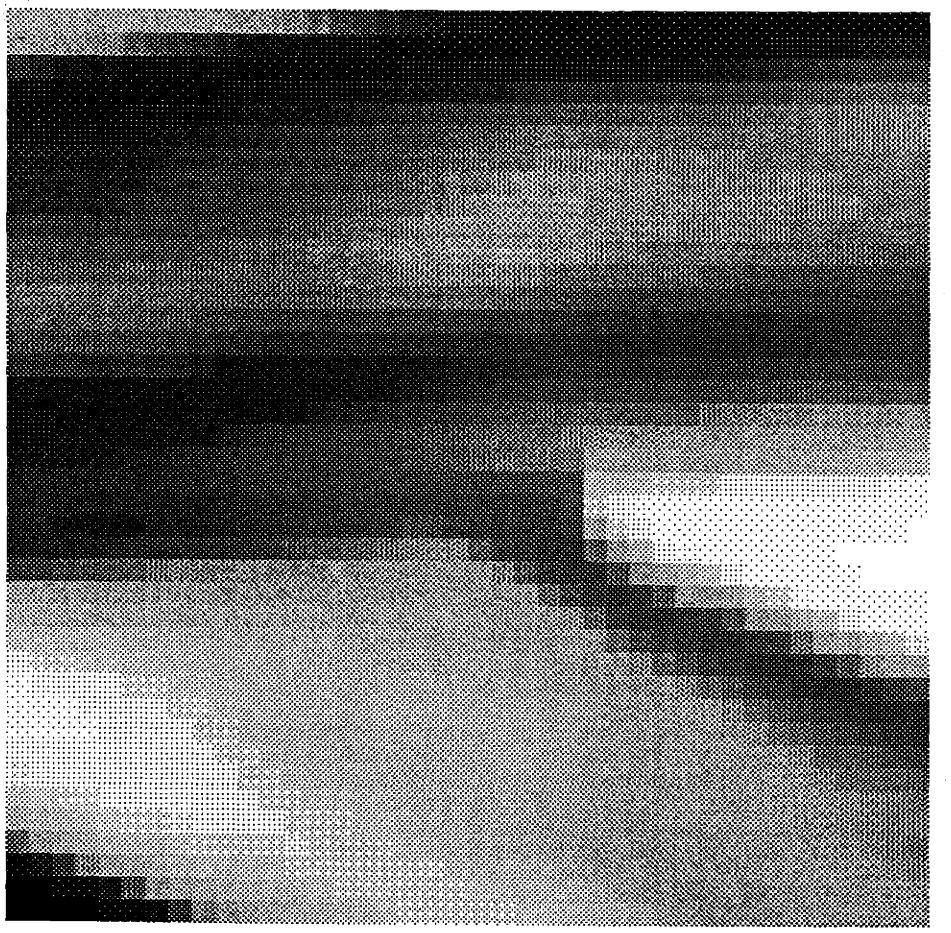

Figure 1.1: Conductivity Data

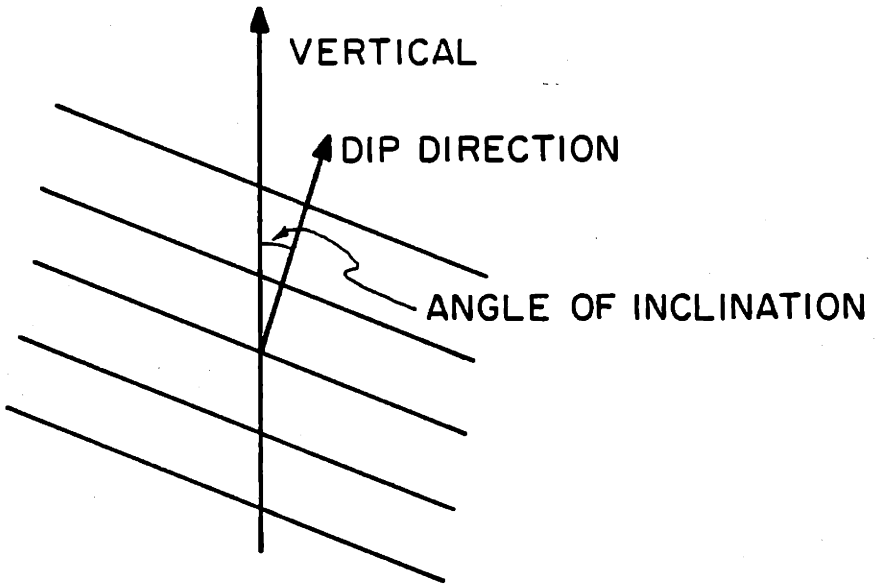

Figure 1.2: Planar Bed Boundaries

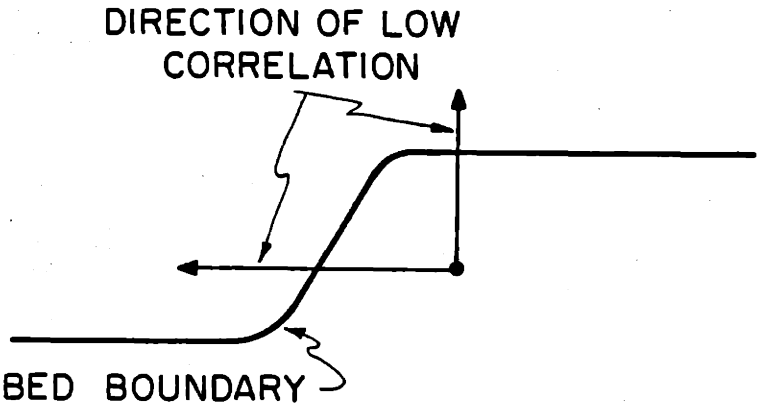

Figure 1.3: Ambiguity of Dip Near Non-Planar Bed Boundary

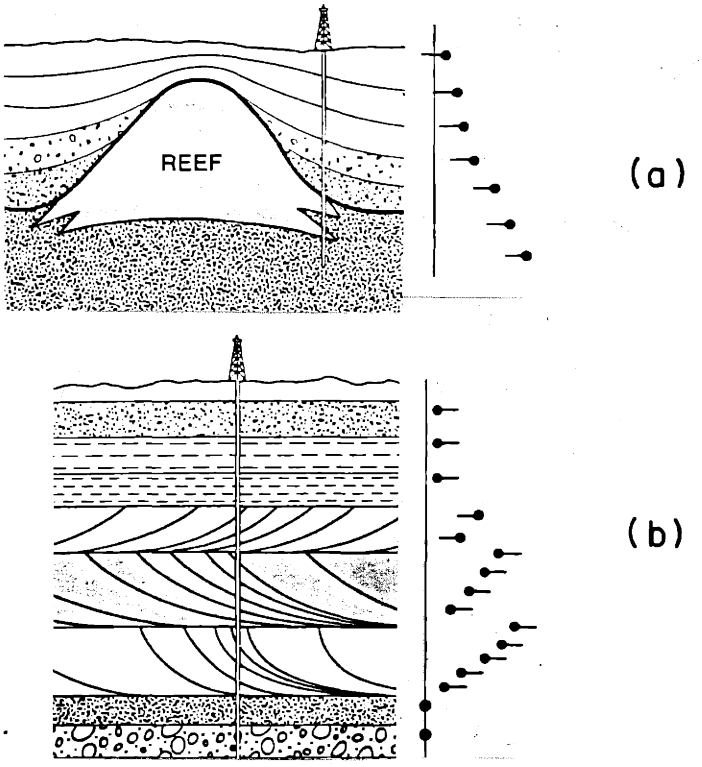

Figure 1.4: Some Simplified Geological Structures and Their Dip Patterns

of dips in the illustration is in the form of "tadpoles" [23]. Each tadpole has a filled circle called the head and an attached straight line called the tail. The location of the head along the horizontal axis is the angle of inclination of the bed at that depth. The angle the tail makes with the vertical is the azimuth of the bed.

The dip log of Figure 1.4a illustrates the prototypical pattern of dips associated with a sandbar. That is, the dips show increasing inclination with depth. The azimuth remains unchanged with depth.

Figure 1.4b illustrates two beds with smaller beds embedded within them. Also, shown is the associated dip log. This figure illustrates a more subtle geological structure which occurs at a much smaller scale than the scale of the sandbar. The internal bedding structures illustrated here are a result of the way in which sediments are deposited by the flowing fluid which carried them. The azimuth of these small scale beds contain information about the direction of fluid flow. The internal beds are usually of much lower contrast than the beds of larger scale since they are composed of primarily similar sedimentary materials.

The dip log of Figure 1.4b is prototypical for these internal beds known as "cross-beds" [23]. The dip log shows the large scale beds at zero dip. The cross-beds are identified by patterns of dip which decrease in inclination with an increase of depth. The azimuth indicates the direction of fluid flow at the time of deposition.

The two examples of Figure 1.4 illustrate the character of the problems associated with automated processing of the data. The data contains features, such as dip, which are spatially varying and which occur at more than one spatial scale. In the next sub-section we discuss our approach to dip estimation from the two-dimensional conductivity data obtained from a borehole wall. Also, we give an overview of the thesis.

1.2 Our Approach to the Dip Estimation Problem

The primary signal processing techniques used in this thesis are *Maximum Likelihood* (ML) and *Maximum a Posteriori* (MAP) estimation. The contributions of this thesis are primarily in the construction and analysis of Markov random field models which capture the character of two-dimensional layered data. These models yield new understanding of a particular class of two-dimensional anisotropic random fields and also lead to efficient and effective algorithms.

The sedimentary process yields a three-dimensional structure, that is, the geological record. However, the underlying process is one that has some temporal causality. We can view the sediments as resulting from two scales of a single process. On a very short time (and thus spatial) scale a sedimentary layer is added to the geological record. Within the lateral extent of the layer one would expect a high correlation of physical properties due to the similarity of materials being deposited. On a much longer time scale many layers have been added to the geological record. The underlying physical process which is depositing the sediment may have found new sources of materials or the process may have totally changed. Thus, we would expect low correlation of physical properties from layer to layer. Even incrementally within a layer the correlation is expected to be higher laterally than it is in the orthogonal direction.

The physical process underlying sedimentation and our conceptual definition of dip point to the importance of relating dip directly to the correlation structure of the data. Markov models are a natural framework for modeling correlation structures. Furthermore, Markov models have the additional advantage of being local models. Local models are useful since they are parsimonious representations and they often lead to efficient signal processing algorithms.

The concept of a Markov Random Field (MRF) [15] is a generalization of the one-dimensional Markov process. MRF's suit our modeling goals and, consequently, they are utilized throughout this thesis. Chapter 2 presents some background materials concerning MRF's. Furthermore, Chapter 2 introduces an important new

class of covariance functions which we term ARC functions. Chapter 3 presents a binary MRF model for conductivity images. This chapter yields some valuable insights into the use of MRF's for modeling and signal processing. Furthermore, Chapter 3 makes an important contribution in that it demonstrates that layered random fields can be generated using Markov models. However, it is difficult to mathematically analyze the correlation structure of the particular binary MRF model introduced in Chapter 3 and its performance falls short of our signal processing goals. Consequently, Chapter 4 reformulates the problem with a non-binary MRF model based on ARC functions. This model is mathematically tractable and leads to an in-depth analysis. The analysis provides us with increased intuition about the MRF models and also suggests a sequence of signal processing algorithms which calculate fast, approximate Maximum Likelihood estimates of dip. The most important contribution of Chapter 4 is the demonstration of how the properties of ARC functions can be used to make our signal processing algorithms more efficient. Chapter 5 addresses the problem of bed boundary estimation. The models used in Chapter 5 are natural extensions of those developed in Chapter 4. The estimation of bed boundaries is an important aspect of dip estimation (as has already been noted). Furthermore, the estimation of bed boundaries helps us to compress the quantity of dip estimates by helping to identify regions of uniform dip. The contribution of Chapter 5 is a new model for local edges. This model is strongly related to the correlation models of Chapter 4 and we show how to combine the results of the two chapters to achieve global line estimates for bed boundaries in layered data. The contributions of Chapter 6 deal with some of the issues of hierarchical processing of data by separating features in the data by their spatial scale. We demonstrate some simple methods for accomplishing separation of the data by spatial scale which depend on our results in previous chapters. Finally, Chapter 7 summarizes the results and contributions of this thesis and identifies some of the major research areas which would be extensions to this work.

Chapter 2

Background

This chapter discusses the primary ideas and mathematical tools used in the remainder of this thesis. The sections on Markov Random Fields, the Metropolis algorithm, and Simulated Annealing are most useful for the results of Chapter 3 and the first half of Chapter 4. The section on Angle Reference Covariance functions is useful in the second half of Chapter 4. All the remaining mathematical tools are developed within the body of the thesis and in the appendices to the chapters.

2.1 Markov Random Fields

Markov Random Fields (MRF's) [15], [25], [5], [2], [28], [12], [14], [13] are the central modeling tool used in this thesis. Analogous to Markov chains, MRF's have special properties which make them very attractive modeling tools. These properties greatly simplify the problem of modeling complex processes both conceptually and computationally.

A MRF is a collection of random variables

$$\underline{s} = \{s_1, s_2, ..., s_M\} \tag{2.1}$$

which have the Markov property. This property states that the distribution of the i^{th} random variable conditioned on the remaining random variables can be simplified so that

$$p(s_i | s_k, k = 1, 2, ..., M, k \neq i) = p(s_i | s_k, k \in N_i) \tag{2.2}$$

where N_i is a subset of the indices $\{1, 2, ..., M\} - i$ and is termed the neighborhood of index i.

So far, the formulation of the MRF is very general. Any random field is a MRF when the appropriate neighborhood structure is imposed. For example, choosing $N_i = \{1, 2, ..., M\} - i$ ensures that any random field is a MRF.

The MRF's of real interest are those with small neighborhoods. In most cases we construct MRF's with neighborhoods that are small and homogeneous (independent of the index i) except possibly at field boundaries.

All MRF's can be represented by an exponential distribution of specific form [3]. This distribution may be written as

$$p(\underline{s}) = \frac{1}{Z} \exp\{u(\underline{s})\} \tag{2.3}$$

where Z is the normalization constant which ensures that $p(\underline{s})$ sums or integrates to unity. The distribution in (2.3) is known as a Gibb's distribution [3] and has had wide spread application in statistical mechanics [20].

The function $u(\underline{s})$ can always be expanded in the following useful form [3]

$$u(\underline{s}) = \sum_{1 \le i \le M} G_i(s_i) + \sum\sum_{1 \le i < j \le M} G_{ij}(s_i, s_j) + \cdots + G_{12\cdots M}(s_1, s_2, ..., s_M) s_1 s_2 \cdots s_M \tag{2.4}$$

Equation (2.4) is completely general in that any function of M variables can be expanded in such a finite series. The restriction on the G functions is that they are non-zero only over sets of indices which are mutual neighbors of each other [3].

The bulk of the work presented here involves functions $u(\underline{s})$ which can be represented by only the second term in the expansion (2.4). That means that the distributions of interest have the following form

$$p(\underline{s}) = \frac{1}{Z} \exp\{ \sum\sum_{1 \le i < j \le M} G_{ij} s_i s_j \} \tag{2.5}$$

where the dependence of G_{ij} on s_i and s_j persists and has been dropped for notational convenience.

2.2 Simulation of MRF's Using the Metropolis Algorithm

It is extremely important to be able to produce sample functions drawn from a MRF such as the one represented in (2.5). Not only does this allow us to examine the results of our modeling efforts and provide us with synthetic data, but it is also an essential element of the Simulated Annealing algorithm to be discussed later in this chapter. The method by which we draw sample functions from MRF's is known as the Metropolis algorithm [18].

A flow chart of the Metropolis algorithm is illustrated in Figure 2.1. The algorithm begins by choosing an arbitrary set of initial values for the elements of the set $\underline{s} = \{s_1, s_2, ..., s_M\}$ from the range of valid values for $\underline{s}$. The algorithm then begins its main loop. This loop begins by choosing one of the indices $\{1, 2, ..., M\}$, say index k. The value of s_k, say r_1, is saved and a new value, for example r_2, is chosen from its valid range of values. The function $u(\underline{s})$ is evaluated at $s_k = r_2$ and $s_k = r_1$. The change in energy in going from the old value to the new value is computed as

$$\Delta u(\underline{s}) = u(\underline{s})|_{s_k=r_2} - u(\underline{s})|_{s_k=r_1} \tag{2.6}$$

where the value of the other elements remain constant. If Δu is negative then the new value of s_k replaces the old value. If Δu is positive then a random sample, x, is drawn from a uniform distribution and is compared to $\exp\{-\Delta u\}$. If x is less than $\exp\{-\Delta u\}$ then, again, the new value of s_k replaces the old value. Otherwise, the old value of s_k is retained. The algorithm then checks to see if it is done and returns to the choice of index if it is not.

In the limit it can be shown that the Metropolis algorithm produces samples from the Gibb's distribution with the function $u(\underline{s})$ [18]. The only constraint is that the algorithm must cycle through the indices $\{1, 2, ..., M\}$ in such a way that all the indices are considered an infinite number of times. For our implementation of the algorithm we step through the indices sequentially hitting each index an equal number of times.

In the practical implementation of the Metropolis algorithm one must provide a

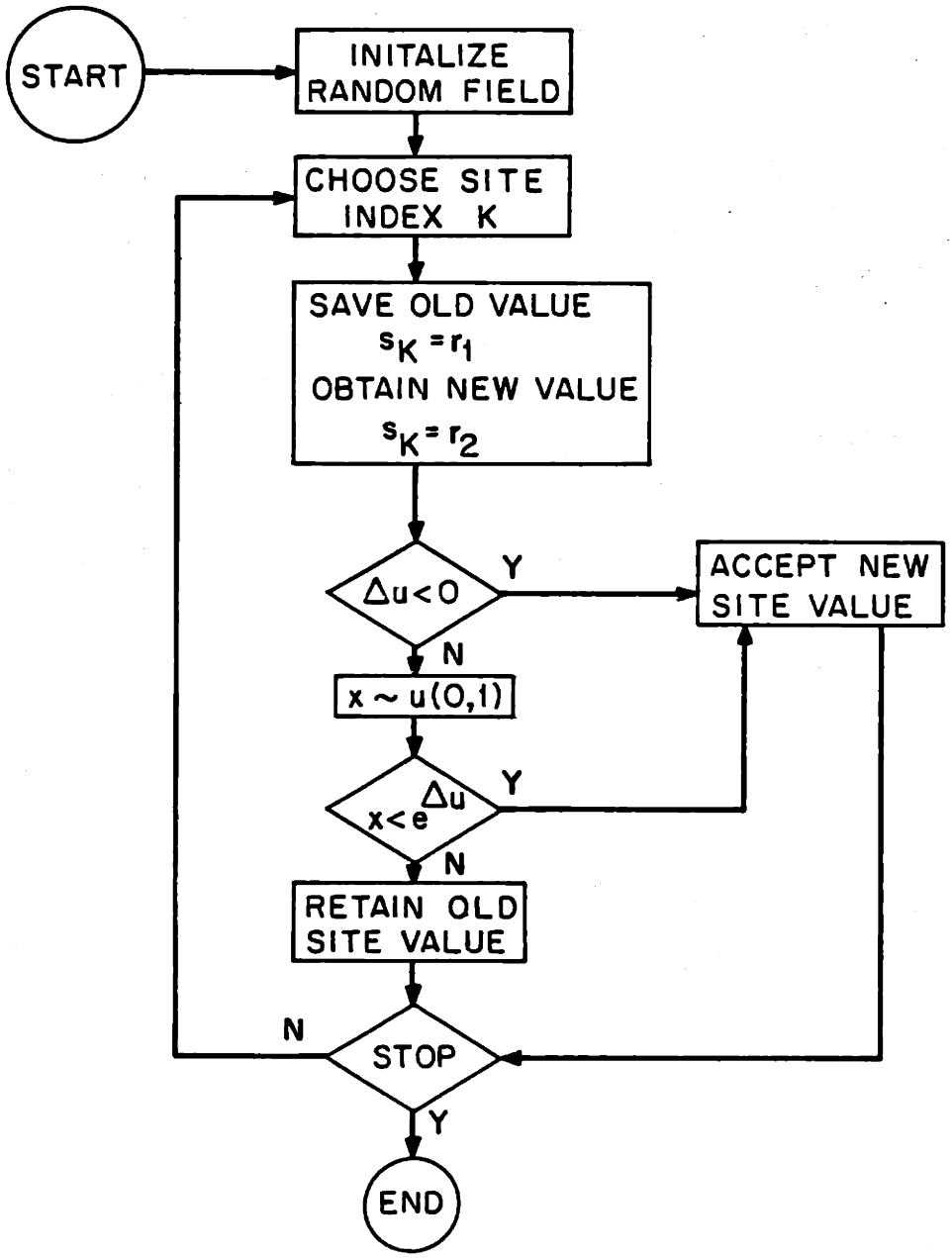

Figure 2.1: Flow Chart of the Metropolis Algorithm

stopping rule. This rule indicates when the algorithm has converged sufficiently to the equilibrium Gibb's distribution. A rule which we have found useful is as follows.

We associate the three time series $f_i(t_k)$, $i = 1,2,3$ with the three possible decision outcomes of the algorithm at time t_k described in the previous paragraph. We assign the values $f_1(t_k) = 1$ and $f_2(t_k) = f_3(t_k) = 0$ if $\Delta u < 0$. If $\Delta u \geq 0$ and $x < \exp\{-\Delta u\}$ then we assign $f_2(t_k) = 1$ and $f_1(t_k) = f_3(t_k) = 0$. The only remaining possibility is to assign $f_3(t_k) = 1$ and $f_1(t_k) = f_2(t_k) = 0$.

The three time series reflect the state of the Metropolis algorithm. When the algorithm starts up the relative ratio of 1's in the three series will reflect the random initialization of $\underline{s}$. Since the initial value of $\underline{s}$ does not necessarily come from a highly likely portion of its distribution we can expect the ratio of 1's in the three series to be changing. However, as the algorithm proceeds and begins to converge we would expect that the ratios of 1's in the three series would also converge to some constant values. We have employed a stopping rule which examines the ratios of the moving windowed sums of the time series. When these ratios start to converge we stop the Metropolis algorithm.

Finally, we point out that the smaller the neighborhood size, the easier it is to compute Δu. The computation of Δu due to a change in the site s_k depends only on the values of s_k and its neighbors. Consequently, the Metropolis algorithm has less computational complexity for smaller neighborhoods.

2.3 Simulated Annealing

Simulated Annealing (SA) [16], [10], [17], [27], [22], [24] is an optimization procedure which has the Metropolis algorithm at its core. We are not concerned with the theoretical aspects of SA in this thesis. Rather, we make use of SA as an optimization technique for computing *Maximum A Posteriori* (MAP) estimates of dip and other quantities. In this section a qualitative description of how SA works is given.

As already noted, SA incorporates the Metropolis algorithm as its central component. Suppose it is desired to find the $\underline{s}$ which minimizes $u(\underline{s})$. The SA algorithm

proceeds as follows. First a parameter, T, known as the temperature is chosen. The value of T is chosen initially to be "high". Then, using the Metropolis algorithm, sample functions are generated from

$$p(\underline{s}) = \frac{1}{Z}\exp\{u(\underline{s})/T\} \tag{2.7}$$

The value of T is then lowered gradually while the Metropolis algorithm is continued. This version of the SA algorithm can be shown (under the correct conditions) to converge to an $\underline{s}$ which minimizes $u(\underline{s})$ [10].

Conceptually, the SA algorithm works in the following fashion. When T is high the distribution in (2.7) is relatively flat. The sample functions all occur with almost equal probability. As the temperature is lowered, the distribution tends to have sharper peaks at the minima of $u(\underline{s})$ and more of the density is concentrated about the major peaks of the distribution. Consequently, the sample functions tend to occur more frequently in the vicinity of the major peaks. As the value of T gets even smaller almost all the distribution is concentrated about the sample points which are the maxima of the distribution and, therefore, the global minima of $u(\underline{s})$. This process is illustrated in Figure 2.2.

The schedule of values that T takes in a practical implementation of the SA algorithm is an important issue. Convergence to the extrema of the distribution depends on a schedule which slowly lowers T [10]. A schedule which is too fast will yield only local as opposed to global minima of $u(\underline{s})$.

2.4 MAP Estimation By SA

Much of the signal processing done in this thesis is accomplished using MAP estimation. To review the MAP estimation methodology it is assumed that the problem formulation consists of two random vectors, $\underline{x}$ and $\underline{y}$, for which the joint distribution, $p(\underline{x},\underline{y})$, is known. One of the vectors, say $\underline{y}$, is observed and its value is $\underline{Y}$. The MAP estimate of $\underline{x}$ is the vector value which maximizes the conditional

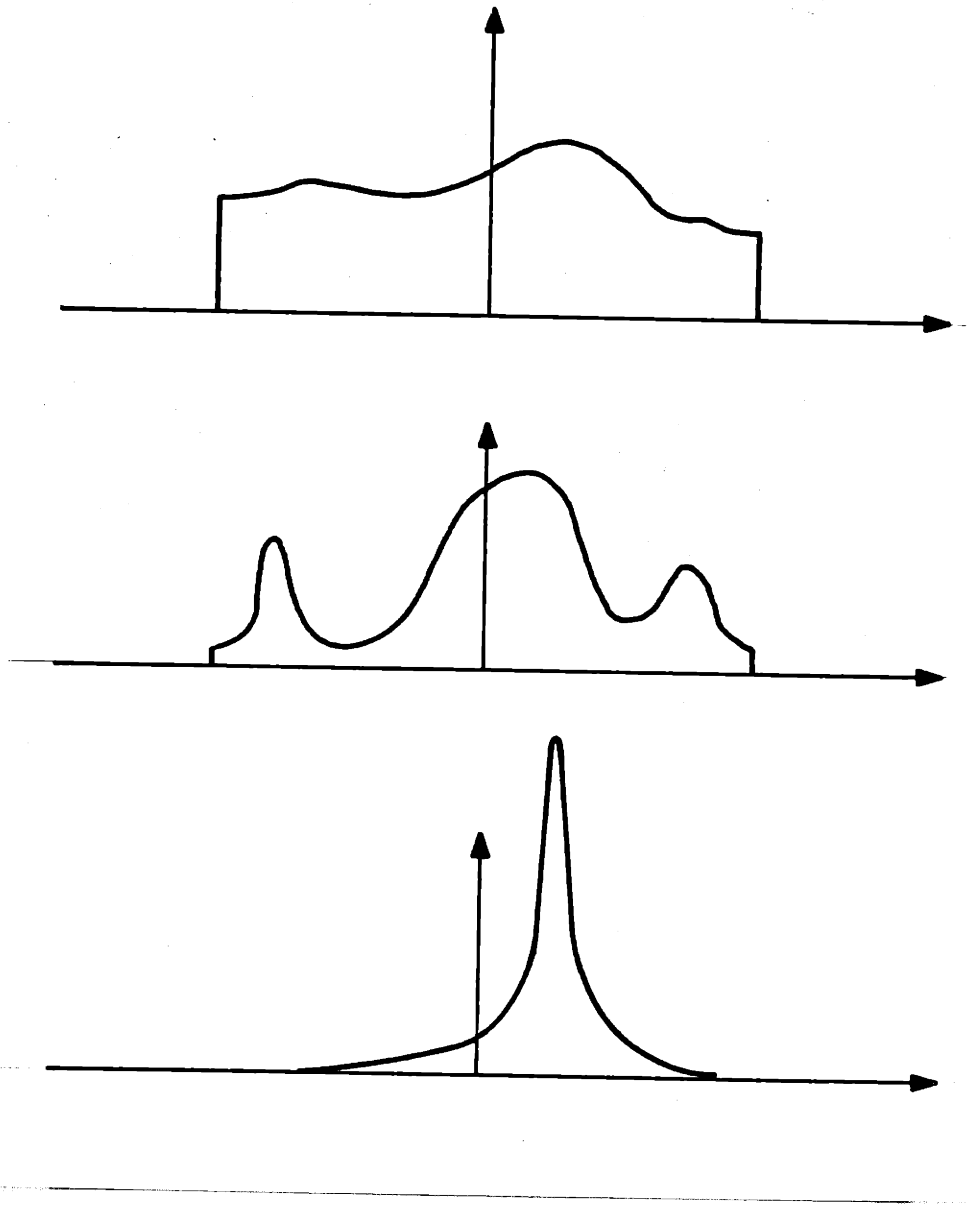

Figure 2.2: Conceptual Illustration of the SA Algorithm

distribution of $\underline{x}$ given $\underline{y} = \underline{Y}$, or

$$\begin{aligned} \hat{\underline{x}}_{MAP} &= \arg\max_{\underline{x}} p(\underline{x}|\underline{y} = \underline{Y}) \\ &= \arg\max_{\underline{x}} \frac{p(\underline{x}, \underline{y} = \underline{Y})}{p(\underline{y} = \underline{Y})} \\ &= \arg\max_{\underline{x}} p(\underline{x}, \underline{y} = \underline{Y}) \end{aligned} \tag{2.8}$$

As (2.8) indicates the MAP estimate consists of determining the value of $\underline{x}$ which maximizes $p(\underline{x}, \underline{y})$ when $\underline{Y}$ is substituted in as the value of $\underline{y}$.

Thus, MAP estimation is an optimization procedure. SA is one of the methods used to find MAP estimates in this thesis. In particular the joint distributions of interest to us have the form

$$p(\underline{s}, \underline{x}) = \frac{1}{Z} \exp\{-u(\underline{s}, \underline{x})\} \tag{2.9}$$

where $\underline{s}$ is an observed vector and we desire an estimate of $\underline{x}$.

2.5 Angle Reference Covariance Functions

This section introduces a class of covariance functions termed Angle Reference Covariance (ARC) functions. The ARC functions are covariances of homogeneous, anisotropic, two-dimensional random fields. A special property of ARC functions is very useful for the signal processing algorithms we introduce in Chapter 4.

ARC functions depend on a two-dimensional unit vector, d. Since d is of unit length it is completely specified by an angle, ϕ (see Figure 2.3). A function $K(\underline{x}, \underline{y})$, where $\underline{x}$ and $\underline{y}$ are both points in the plane, is an ARC function if

$$K(\underline{x}, \underline{y}) = \hat{K}(d \cdot (\underline{x} - \underline{y})) \tag{2.10}$$

By expressing the difference vector, $\underline{x} - \underline{y}$, in polar coordinates (r, θ) we can write (2.10) as

$$K(\underline{x}, \underline{y}) = \hat{K}(r \cos(\phi - \theta)) \tag{2.11}$$

From (2.10) and (2.11) it can be seen that K is homogeneous and anisotropic. The anisotropy is due to the comparison of θ to an absolute reference angle, ϕ (which is

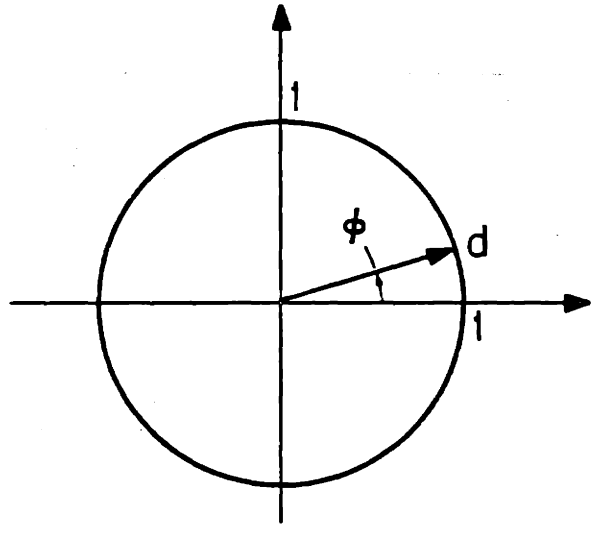

Figure 2.3: Unit Direction Vector d

specified by the unit vector d). Thus, K is called an "Angle Reference Covariance". In the ensuing discussion the dependence of K on the direction vector d is denoted by a subscript as in K_d or K_ϕ.

The most important signal processing property of ARC functions is presented in the following theorem.

Theorem 1 *Let K be an ARC function defined on a disk, D, centered at the origin. Also, let $\hat{d}$ and $\tilde{d}$ be two direction vectors related by $\hat{d} = V\tilde{d}$ where V is a two-dimensional rotation matrix. Furthermore, denote the i^{th} eigenvalue and eigenfunction of $K_{\hat{d}}$ and $K_{\tilde{d}}$ as $\hat{\lambda}_i$, $\hat{e}_i(\underline{x})$ and $\tilde{\lambda}_i$, $\tilde{e}_i(\underline{x})$, respectively. Under the above conditions $\tilde{\lambda}_i = \hat{\lambda}_i$ and $\tilde{e}_i(\underline{x}) = \hat{e}_i(V\underline{x})$.*

Proof 1 *The eigenstructure of $K_{\tilde{d}}$ is obtained by solving the integral equation*

$$\int_{\underline{y}\in D} K[\tilde{d}\cdot(\underline{x}-\underline{y})]\tilde{e}_i(\underline{y})d\underline{y} = \tilde{\lambda}_i\tilde{e}_i(\underline{x}) \tag{2.12}$$

where for y_1 and y_2 the components of $\underline{y}$ we take the differential area $d\underline{y} = dy_1 \cdot dy_2$. Similarly, for $K_{\hat{d}}$ we have

$$\int_{\underline{y}\in D} K[\hat{d}\cdot(\underline{x}-\underline{y})]\hat{e}_i(\underline{y})d\underline{y} = \hat{\lambda}_i\hat{e}_i(\underline{x}) \tag{2.13}$$

Substituting for $\hat{d}$, (2.13) becomes

$$\int_{\underline{y}\in D} K[V\tilde{d}\cdot(\underline{x}-\underline{y})]\hat{e}_i(\underline{y})d\underline{y} = \hat{\lambda}_i\hat{e}_i(\underline{x}) \tag{2.14}$$

Since V is a rotation and rotations preserve angles (2.14) can be written as

$$\int_{\underline{y}\in D} K[\tilde{d}\cdot V^T(\underline{x}-\underline{y})]\hat{e}_i(\underline{y})d\underline{y} = \hat{\lambda}_i\hat{e}_i(\underline{x}) \tag{2.15}$$

Finally, making the change in variables $\underline{\eta} = V^T\underline{x}$, $\underline{\nu} = V^T\underline{y}$ and noting that $d\underline{\nu} = |V^T|d\underline{y} = d\underline{y}$ in disk D we obtain

$$\int_{\underline{\nu}\in D} K[\tilde{d}\cdot(\underline{\eta}-\underline{\nu})]\hat{e}_i(V\underline{\nu})d\underline{\nu} = \hat{\lambda}_i\hat{e}_i(V\underline{\eta}) \tag{2.16}$$

Since the kernels of (2.12) and (2.16) are identical we have that $\tilde{\lambda}_i = \hat{\lambda}_i$ and $\tilde{e}_i(\underline{x}) = \hat{e}_i(V\underline{x})$.

The consequences of Theorem 2.1 are exploited in Chapter 4. Section 4.4 shows how the properties of ARC functions can be used to improve the computational efficiency of some of our signal processing algorithms.

Chapter 3

Binary MRF models

This chapter develops a MRF model for use in estimating spatially varying dip and bed frequency from observations of binary layered random fields. Section 3.1 of this chapter presents a model for MRF's with constant dip and bed frequency. Section 3.2 expands the model in Section 3.1 to address MRF's with spatially varying dip and bed frequency. Section 3.3 uses the model in Section 3.2 to produce some synthetic data sets. These synthetic data sets substantiate the models presented in Section 3.2 and are useful for illustrating the use of signal processing algorithms developed in Section 3.4. Section 3.5 contains examples and Section 3.6 contains a critique and summary of the results in this chapter.

3.1 Constant Dip and Bed Frequency Model

In this section a simple MRF model is introduced which is dependent on the two parameters dip, d, and bed frequency, λ. Arguments for why the model represents layered random fields are given. Section 3.3 substantiates these arguments by presenting examples of synthetic data generated by the models discussed here.

The model discussed in this section is a second order MRF. Its joint distribution

can be expressed as

$$\begin{aligned} p(s|d,\lambda) &= p(s_1, s_2, ..., s_M|d,\lambda) \\ &= \frac{1}{Z}\exp\{\sum_{1\le i<j\le M}\sum G_{ij}s_i s_j\} \end{aligned} \tag{3.1}$$

where the s_i are binary valued (± 1), Z is a normalization constant and the G_{ij} are non-zero only when i and j are mutual neighbors. There is an implied geometry which accompanies the model in (3.1). The data is assumed to be arranged on an $N \times N$ Cartesian grid. Thus, $M = N^2$ and the location of data element s_i is at grid location (k,l) where[1] $l = [\text{int}(i/N) + 1]$ and $k = [i - (l-1)N]$. If p_i is a two vector representing the location of the data element s_i then $p_i = (k,l)$ in this geometry.

The only remaining part of the model to be specified are the functions G_{ij}. Let,

$$G_{ij} = \begin{cases} \cos[\lambda\pi|d \cdot (p_i - p_j)|] & i \in N_j, j \in N_i \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

The parameters in (3.2) are the unit dip vector, d, the bed frequency, λ, the location vector of the data point, p_i, and the neighborhood set, N_i.

The model specified by (3.1) and (3.2) has sample functions which are composed of layers at "average" thickness $1/\lambda$. Examination of Figure 3.1 helps provide a heuristic argument for why this is so. Assume, for the following discussion, that the neighborhood sets, N_i, are very large. This assumption allows us to ignore the effect of the neighborhood structure and, so, the function G_{ij} is only dependent on the 2-vectors p_i and p_j. The illustration in Figure 3.1 displays the cross-section of G_{ij} along the direction of the 2-vector, d. G_{ij} is a constant in the direction orthogonal to d.

When two elements of the data vector, s_i and s_j, are positioned so that their difference vector $p_i - p_j$ is orthogonal to d then $G_{ij} = +1$. Consequently, according to (3.1), it is more likely that s_i and s_j have the same sign. That is $s_i = s_j = +1$ or $s_i = s_j = -1$. When the difference vector $p_i - p_j$ is parallel to d and is of length $1/\lambda$,

[1] int is the function which takes the integer portion of i/N

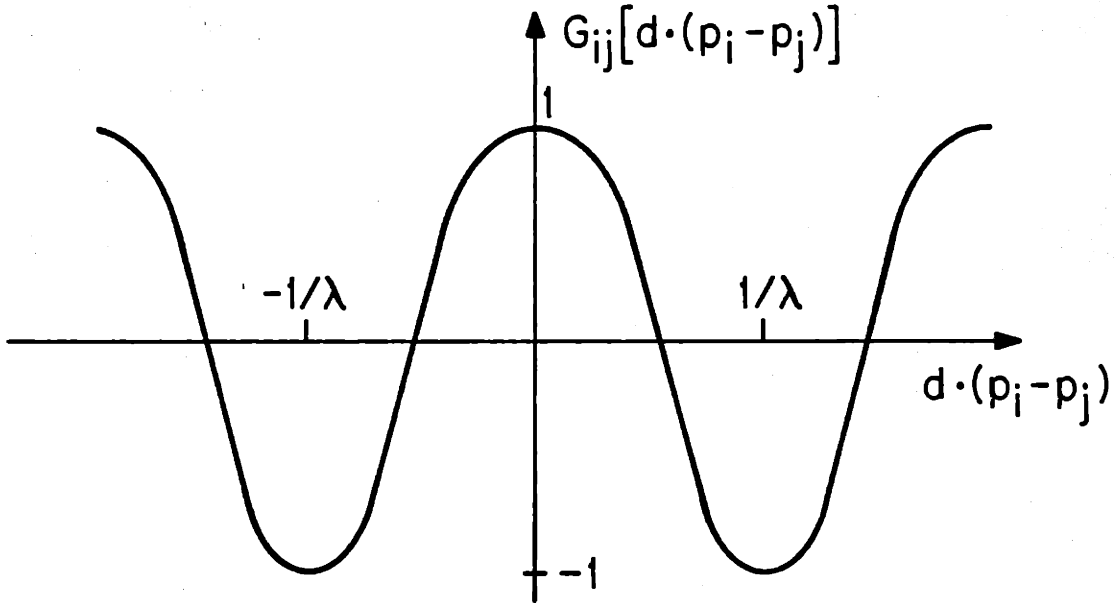

Figure 3.1: Cross-Section of G_{ij} in Direction of d

the function $G_{ij} = -1$. In this case it is more likely that s_i and s_j are of opposite sign. That is $s_i = -s_j = +1$ or $s_i = -s_j = -1$.

The neighborhood structure imposes a limit on the range of possible values for λ. To get a layered sample function from the model requires that the neighborhood set includes data points which are at least $1/\lambda$ in distance from each other. For the remainder of this chapter it is assumed that the neighborhood N_i is the set of indices which are contained in a square centered at p_i, the position of data element i. Furthermore, it is assumed that N_i contains enough indices so that some of the data positions are at least $1/\lambda$ away from each other. Given that the neighborhood structure is consistent with the actual value of λ, the above analysis suggests that sample functions generated by (3.1) and (3.2) tend to look like the illustration in Figure 3.2.

The next section of this chapter expands on the model presented here. The emphasis is on using this model for spatially varying dip and bed frequency. This is accomplished by thinking of the dip and bed frequency as spatially varying fields themselves and, in particular, as components of a vector MRF consisting of dip; bed frequency; and the binary pixel values, s_i.

3.2 Spatially Varying Dip and Bed Frequency Model

In this section the model in Section 3.1 is expanded and modified to account for random fields which exhibit spatially varying dip and bed frequency. The model in this section takes the dip and bed frequency to be vectors of the same dimensionality as the data. Consequently, for each component of the data, s_i, there is a corresponding component of the dip, d_i, which quantifies the value of dip locally around the location p_i. Similarly, there is a component of the bed frequency, λ_i, which is a measure of the bed thickness in the vicinity of location p_i. Furthermore, the dip and bed frequency are no longer considered to be parametric components of the model. Rather, the dip and bed frequency are themselves modeled as components

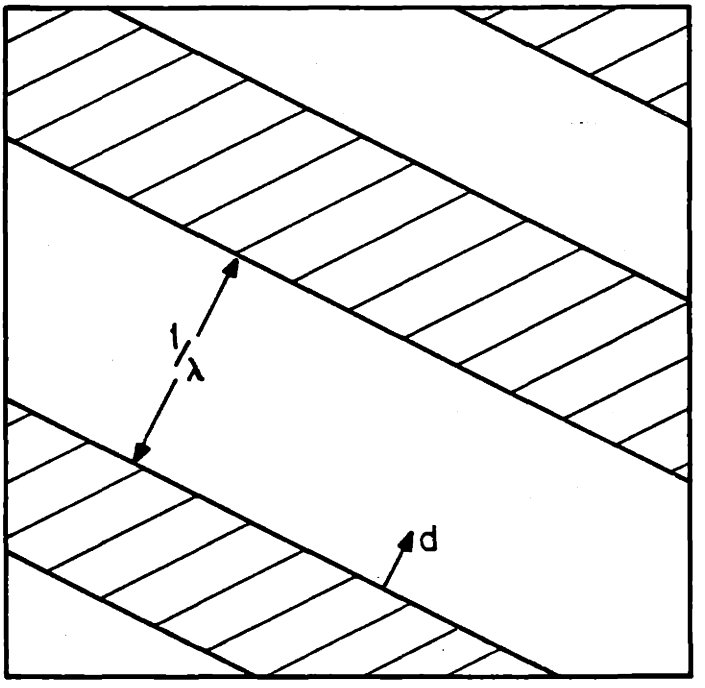

Figure 3.2: Typical Sample Function for MRF

of a vector MRF which interact with each other and the data vector, $\underline{s}$.

Specifying the dip and bed frequency vectors by $\underline{d}$ and $\underline{\lambda}$, respectively, the joint distribution of the data with $\underline{d}$ and $\underline{\lambda}$ is modeled as

$$p(\underline{s},\underline{d},\underline{\lambda}) = \frac{1}{Z}\exp\{U(\underline{s},\underline{d},\underline{\lambda})\} \tag{3.3}$$

where Z is the normalization constant[2] and U is a function still to be specified.

The U function is given by

$$\begin{aligned} U(\underline{s},\underline{d},\underline{\lambda}) &= \sum_{1\le i<j\le M}\sum f(d_i,d_j)g(\lambda_i,\lambda_j)\cos[\bar{\lambda}_{ij}\pi|\bar{d}_{ij}\cdot(p_i-p_j)|]s_is_j \\ &\quad + \sum_{1\le i<j\le M}\sum Df(d_i,d_j) + \sum_{1\le i<j\le M}\sum \Lambda g(\lambda_i,\lambda_j) \qquad (3.4) \\ f(d_i,d_j) &= |d_i\cdot d_j|^{\gamma_d} \qquad (3.5) \\ g(\lambda_i,\lambda_j) &= [1-\frac{|\lambda_i-\lambda_j|}{\max|\lambda_i-\lambda_j|}]^{\gamma_\lambda} \qquad (3.6) \\ \bar{\lambda}_{ij} &= \frac{(\lambda_i+\lambda_j)}{2} \qquad (3.7) \\ \bar{d}_{ij} &= \frac{(d_i+d_j)}{|d_i+d_j|} \qquad (3.8) \end{aligned}$$

where D, Λ, γ_d, and γ_λ are constants. The terms N_i, $\tilde{N}_i$, and $\hat{N}_i$ are three, possibly different, neighborhood structures. In our work all three neighborhoods are taken to be square and centered at element i with $\tilde{N}_i = \hat{N}_i$ and $|\hat{N}_i| < |N_i|$. The choice of constants and neighborhood sizes are non-trivial and are discussed more in Section 3.3.

Notice that (3.4) is in the form of a canonical second order MRF. That is,

$$U(\underline{s},\underline{d},\underline{\lambda}) = \sum_{1\le i<j\le M}\sum G_{ij}s_is_j + \sum_{1\le i<j\le M}\sum D_{ij}f(d_i,d_j) + \sum_{1\le i<j\le M}\sum \Lambda_{ij}g(\lambda_i,\lambda_j) \tag{3.9}$$

The first term in (3.4) is

$$\sum_{1\le i<j\le M}\sum f(d_i,d_j)g(\lambda_i,\lambda_j)\cos[\pi\bar{\lambda}_{ij}|\bar{d}_{ij}\cdot(p_i-p_j)|]s_is_j \tag{3.10}$$

[2]This is a different Z than the one in (3.1)

The term in (3.10) is similar to the model introduced in Section 3.1. The differences are that d and λ are replaced by $\bar{d}_{ij}$ and $\bar{\lambda}_{ij}$ and there are two functions, $f(d_i, d_j)$ and $g(\lambda_i, \lambda_j)$ which scale the term.

The functional form of $\bar{\lambda}_{ij}$ and $\bar{d}_{ij}$ is given in (3.7) and (3.8). These expressions are averages of the i^{th} and j^{th} elements of the dip and bed frequency vectors. Since dip and bed frequency vary throughout the field when relating the i^{th} and j^{th} elements of the conductivity field it is sensible to incorporate the values of dip and bed frequency from both locations. This accounts for use of the averages $\bar{d}_{ij}$ and $\bar{\lambda}_{ij}$.

The two scale functions $f(d_i, d_j)$ and $g(\lambda_i, \lambda_j)$ are used to regulate the effect that the data has on the energy function. The model in (3.4) is supposed to account for conductivity samples which are contained in similarly bedded regions. This means that samples are related to each other when they come from beds of similar dip and bed frequency. If, in examining samples s_i and s_j we find that they come from beds of different dip and/or bed frequency, then it is desirable to remove the mutual effect of the two samples from the evaluation of the energy function. The terms $f(d_i, d_j)$ and $g(\lambda_i, \lambda_j)$ accomplish this by approaching zero when their arguments are very different and unity when their arguments are similar.

The two remaining terms in (3.4) are

$$\sum_{1 \le i < j \le M} \sum D f(d_i, d_j) + \sum_{1 \le i < j \le M} \sum \Lambda g(\lambda_i, \lambda_j) \tag{3.11}$$

These terms are intended to specify how much variability of the dip and bed frequency fields is desired or expected in our model. This is accomplished by setting the values of D and Λ. When D and Λ have large positive values the likelihood function in (3.3) is larger for slowly varying dip and bed frequency fields. The critical aspect of the model is the balance of values between the data term in (3.10) and the smoothing terms in (3.11). When (3.11) is larger in value than (3.10) then smoothing dominates the model. When the alternative is true, then the data dominates the model. It is desirable to obtain smoothing without dominating the data. More about this is discussed in Section 3.3.

3.3 Synthetic Data From MRF Models

In this section we describe the generation of synthetic data based upon models presented in Section 3.2. In particular, sample functions of the data vector $\underline{s}$ are obtained from the distribution given in (3.3) and (3.4) conditioned on knowledge of the vectors $\underline{d}$ and $\underline{\lambda}$. The synthetic data presented in this section is used in Section 3.4 to illustrate the inverse problem of estimating $\underline{d}$ and $\underline{\lambda}$ given observations of the data, $\underline{s}$.

The method used to generate the synthetic data is the Metropolis algorithm discussed in Chapter 2. Three sample functions are illustrated in this section. For each of the sample functions, the functional form of the $\underline{d}$ and $\underline{\lambda}$ vectors is given. The first sample function is the result of constant dip and bed frequency. The second sample function has dip and bed frequency which change across a boundary and are constant on either side of the boundary. The final sample function is generated from spatially varying dip and bed frequency. Each example assumes a square neighborhood structure consisting of a 7×7 array of pixel sites for N_i. It should be noted that the energy terms in (3.11) play no role in the Metropolis algorithm since they do not change from iteration to iteration.

Figures 3.3-3.5 illustrate the resulting sample function generated from constant dip and bed frequency vectors. The values of dip and bed frequency are

$$\begin{aligned} d_i &= \tan^{-1}(\tfrac{1}{3}) \quad i = 1, 2, ..., 1600 \\ \lambda_i &= \tfrac{1}{4} \qquad\quad i = 1, 2, ..., 1600 \end{aligned} \tag{3.12}$$

The MRF associated with (3.12) consists of a 40×40 array of sites where the lower left hand corner element corresponds to $i = 1$ and upper right hand corner to $i = 1600$. The indexing of the remaining elements is obtained by moving along rows from left to right as the field is traversed from bottom to top. Figure 3.3 illustrates the dip field, Figure 3.4 illustrates the bed frequency field, and Figure 3.5 is the resulting synthetic data sample.

The results illustrated in Figure 3.5 reflect the intuition concerning sample functions expounded in Section 3.1. The sample function in Figure 3.5 is layered with beds of approximately $\frac{1}{\lambda} = 4$ pixels per bed in thickness and at approximately

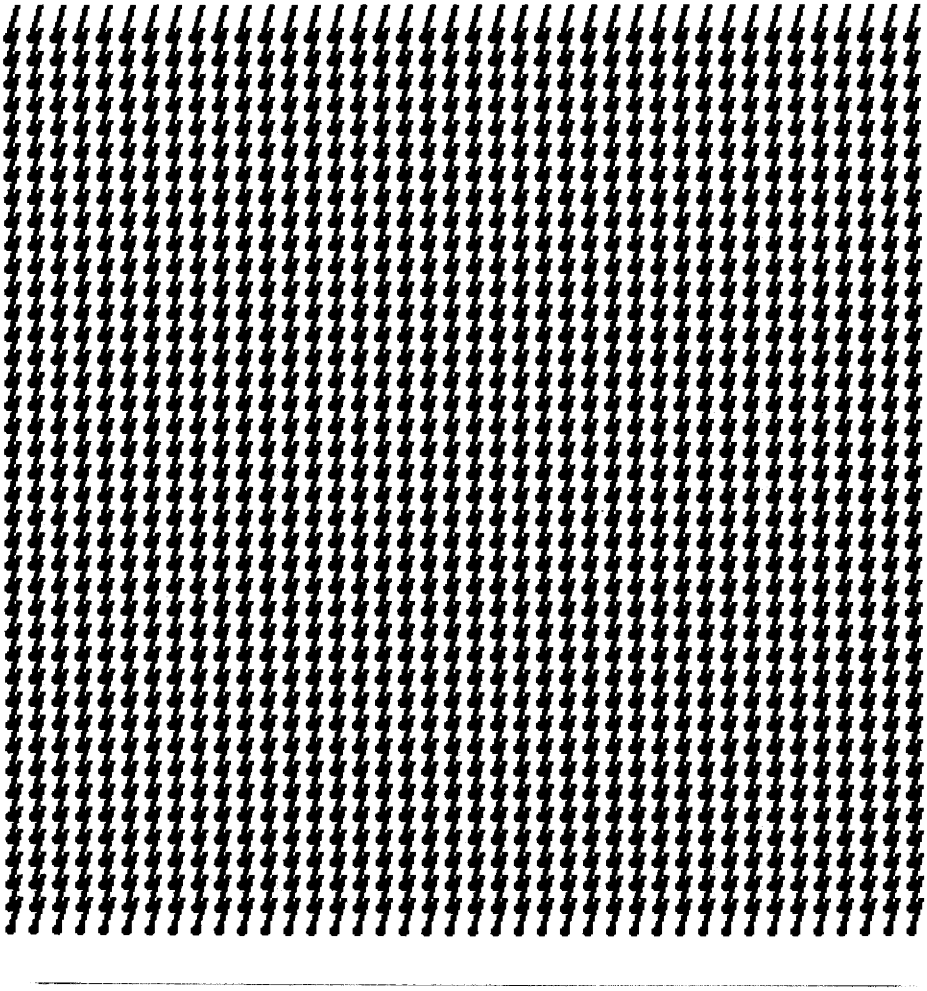

Figure 3.3: Constant $\underline{d}$ field

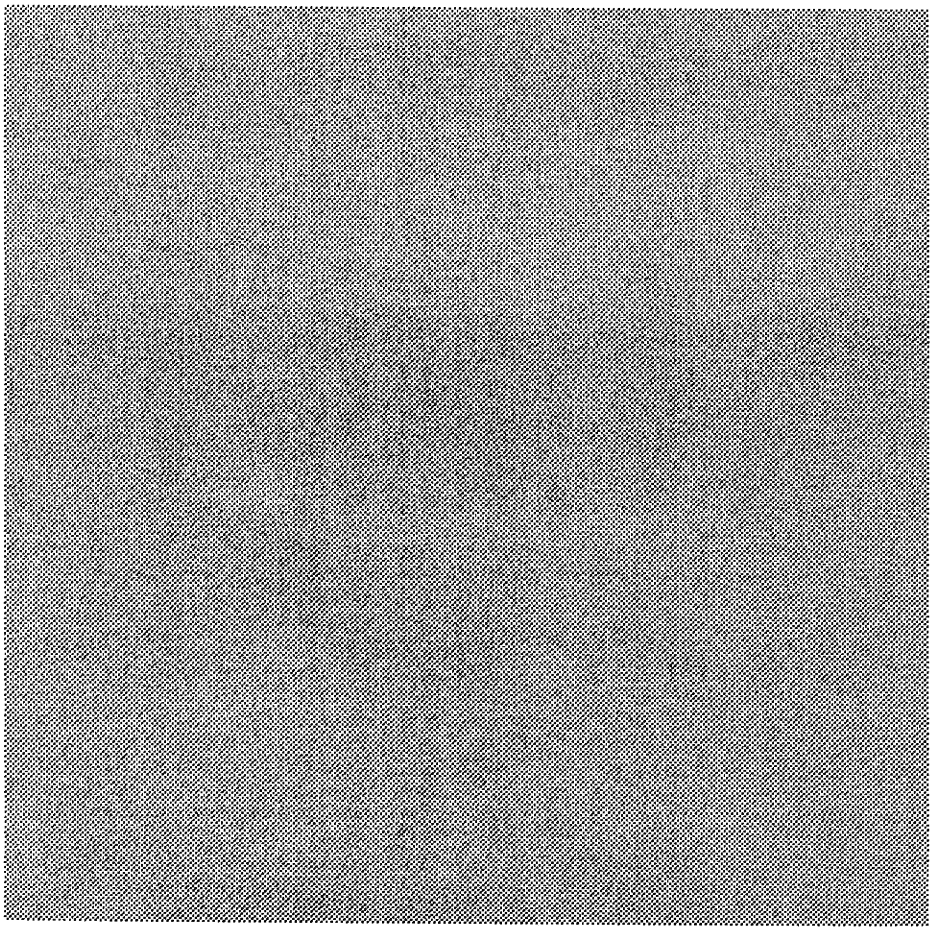

Figure 3.4: Constant $\underline{\lambda}$ field

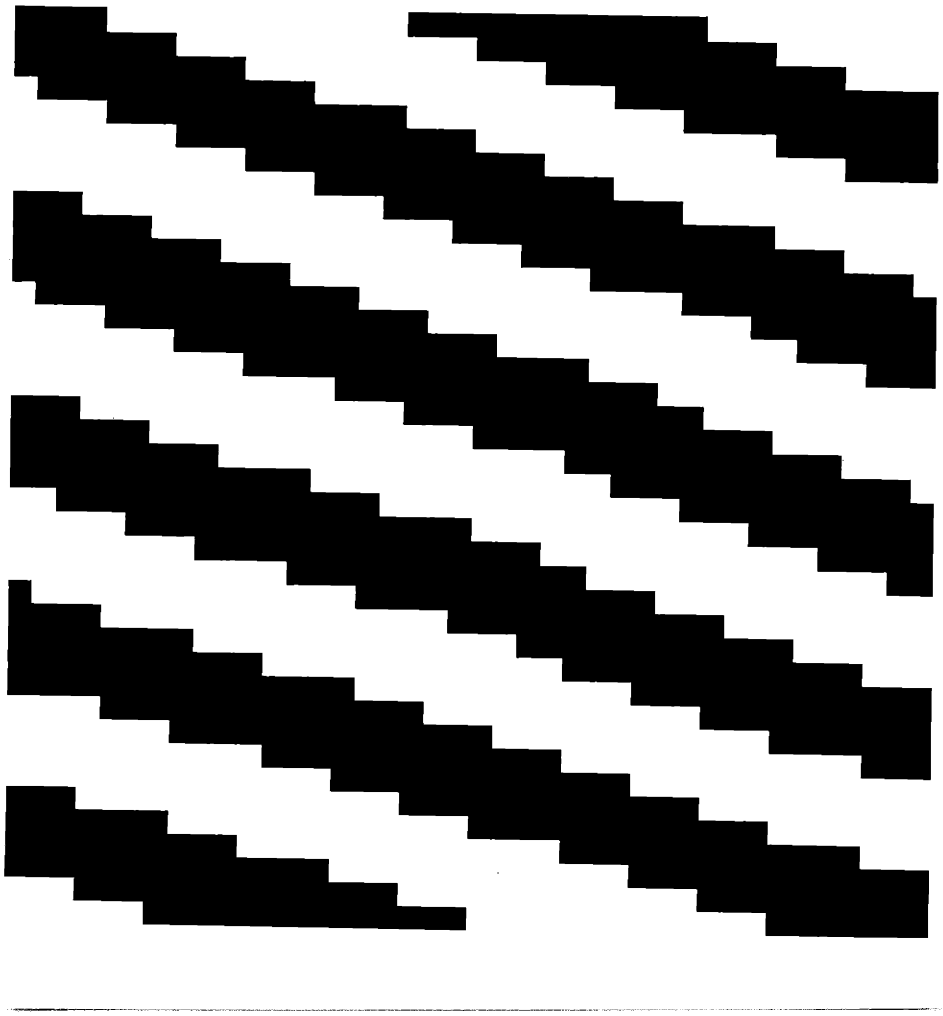

Figure 3.5: Sample Function Generated From Constant $\underline{d}$ and $\underline{\lambda}$ fields

$d = \tan^{-1}(1/3)$ in dip. Note the the generated data is not quite regular since it is a sample from a MRF.

The second sample function is illustrated in Figures 3.7-3.9. In this example the values of the dip and bed frequency change suddenly across a boundary and are constant on either side of the boundary. The boundary is a straight line described by the equation

$$y = \frac{15}{39}x + \frac{315}{26} \tag{3.13}$$

where again the data consist of a 40×40 array of sites. The elements are indexed $i = 1, 2, ..., 1600$ where the position of element i is at $p_i = (k, l)$ and $l = [\text{int}(i-1/N)+1]$, $k = [i-(l-1)N]$. Consequently, the line in (3.13) cuts between the 12^{th} and 13^{th} rows at column 1 and between the 27^{th} and 28^{th} rows at column 40 which is illustrated in Figure 3.6. The functional form for $\underline{d}$ and $\underline{\lambda}$ is

$$d_i = \begin{cases} \tan^{-1}(\frac{1}{2}) & l \geq \frac{15}{39}k + \frac{315}{26} \\ \tan^{-1}(-2) & l < \frac{15}{39}k + \frac{315}{26} \end{cases} \tag{3.14}$$

$$\lambda_i = \begin{cases} \frac{1}{3} & l \geq \frac{15}{39}k + \frac{315}{26} \\ \frac{1}{2} & l < \frac{15}{39}k + \frac{315}{26} \end{cases} \tag{3.15}$$

Once again, due to the fact that Figure 3.9 is a sample from a MRF, the data is not regular as is seen by the "split bed" in the lower half of the data.

The final example is for continuously varying dip and bed frequency. One point of this example is that dip and bed frequency cannot be specified totally independently of each other. For example, assume that dip is constant and bed frequency is changing in value. This would result in a sample function which attempts to create layers at the inclination d and which varies in thickness according to the functional form of the bed frequency. However, for a layer at inclination d to become thicker requires the layer to change its inclination. This is illustrated in Figure 3.10.

Consequently, specification of varying dip and bed frequency fields requires some careful thought. The following example illustrates a method for choosing a varying dip and bed frequency field that roughly corresponds to layered earth structure called cross-beds [23].

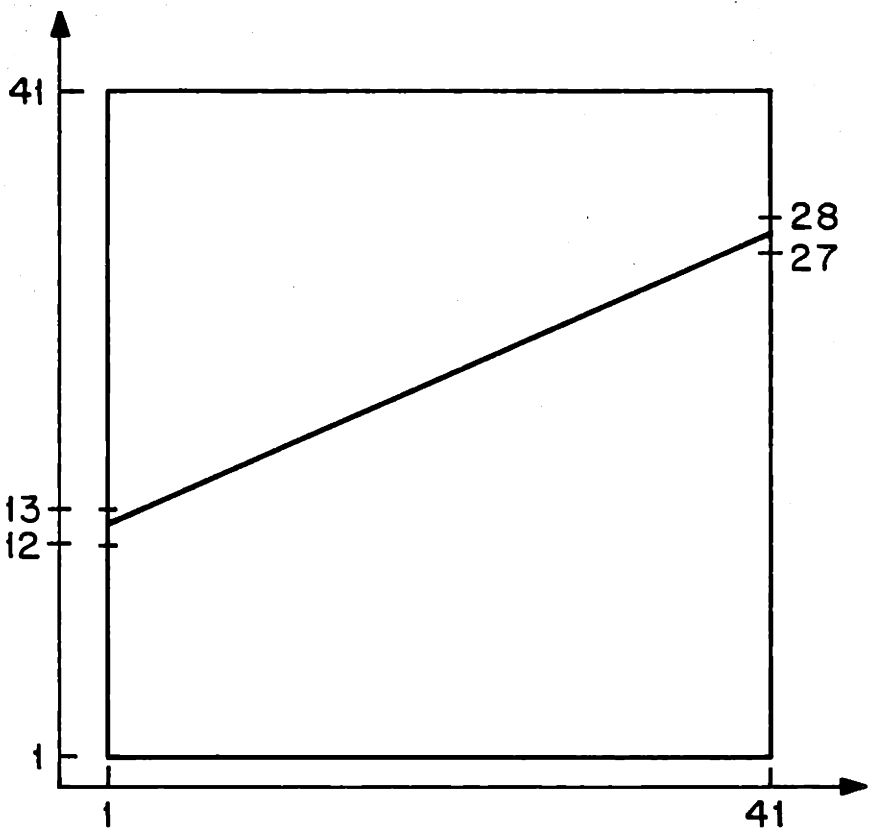

Figure 3.6: Boundary Location in MRF

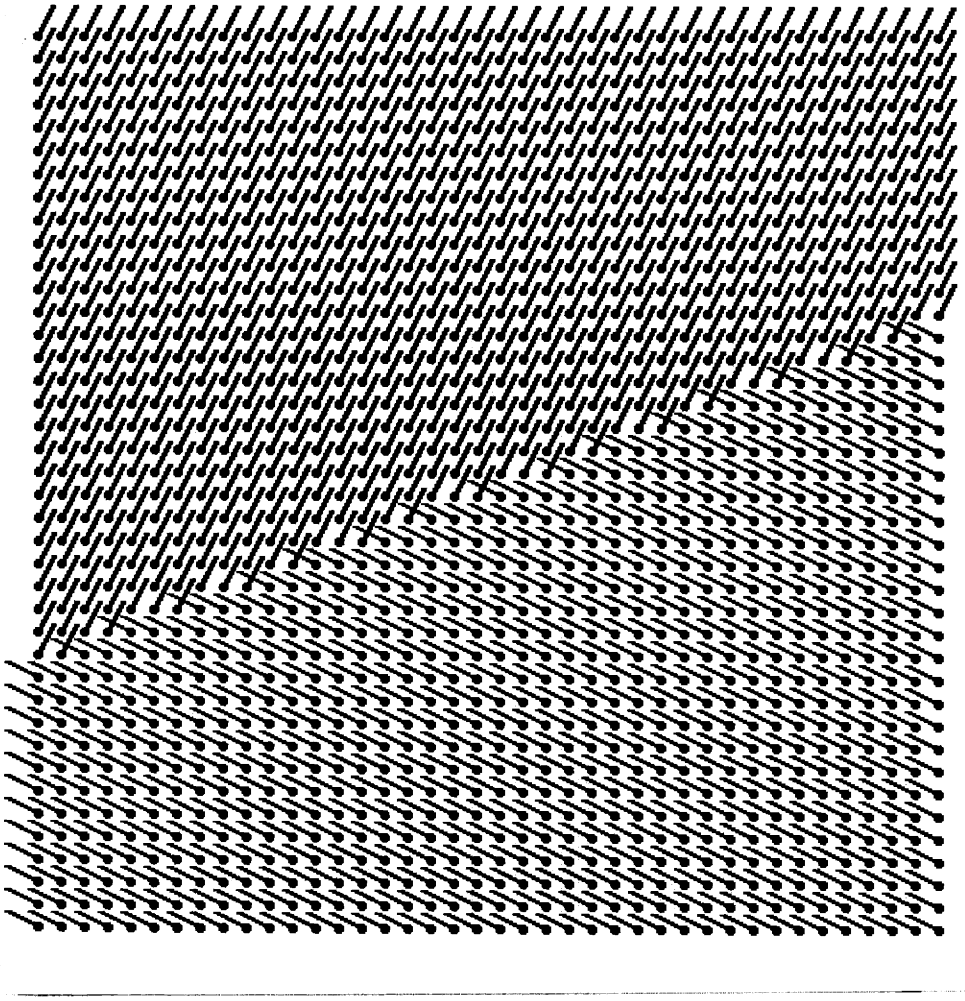

Figure 3.7: Piece-Wise Constant $\underline{d}$ field

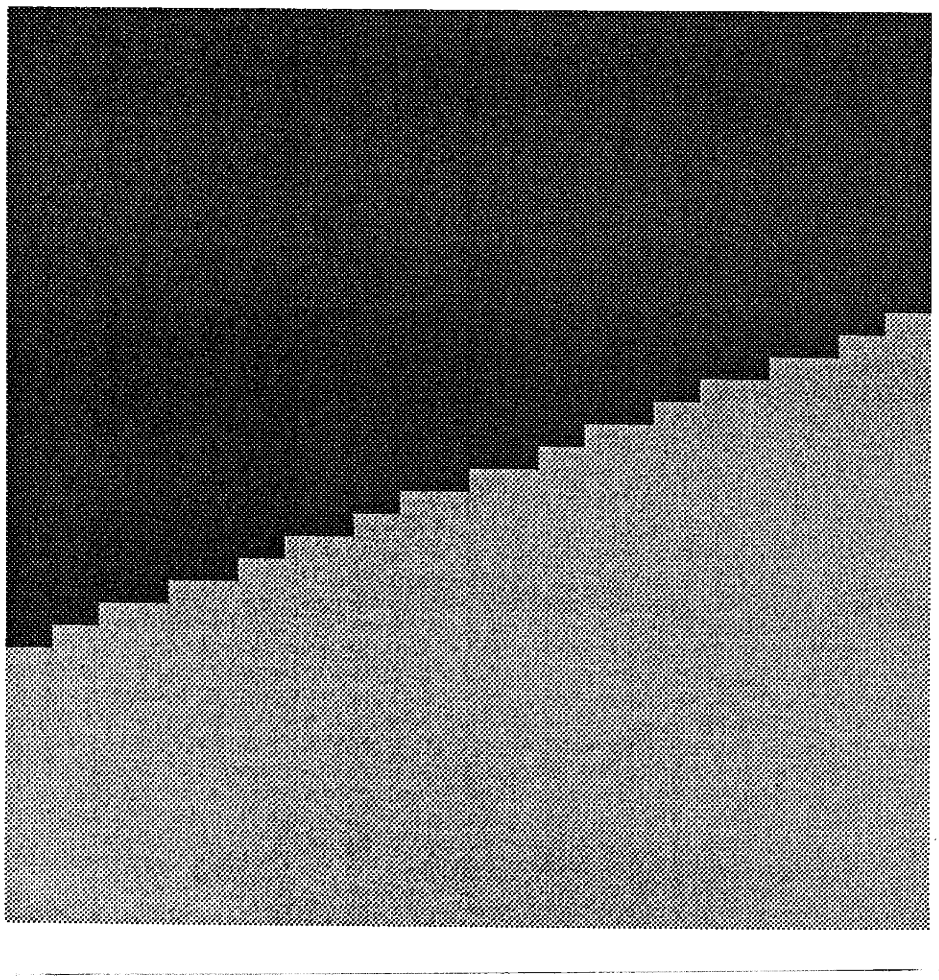

Figure 3.8: Piece-Wise Constant $\underline{\lambda}$ field

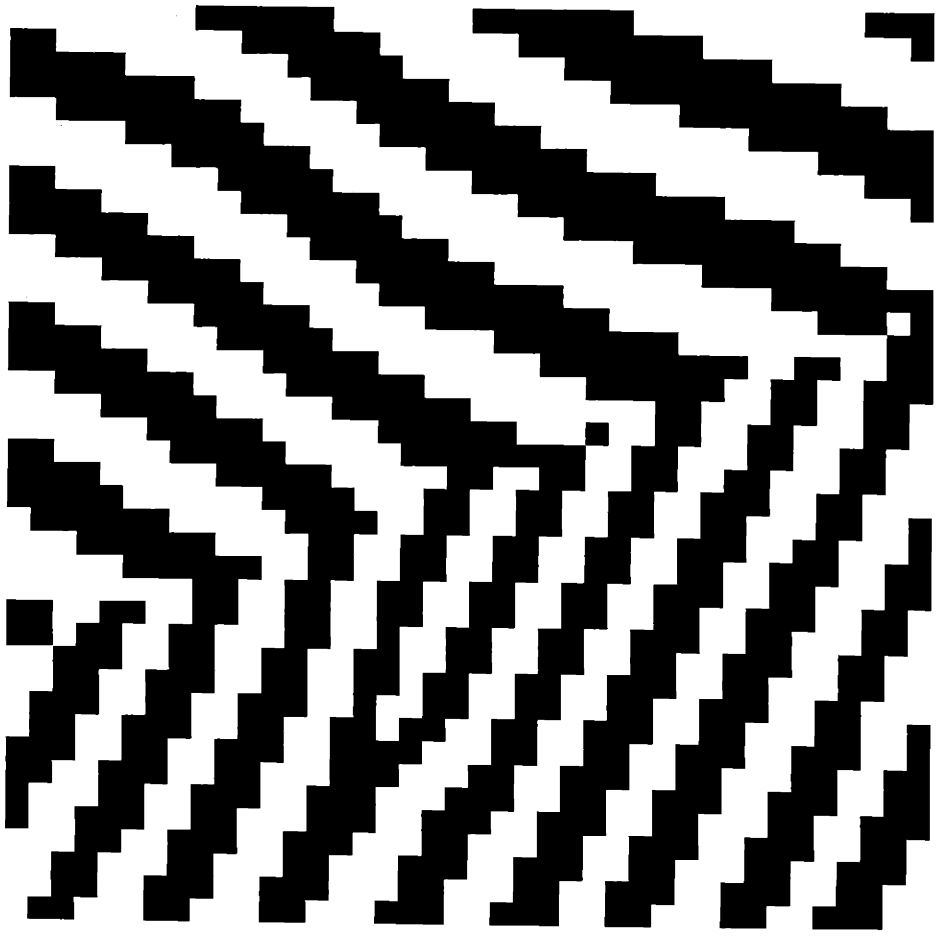

Figure 3.9: Sample Function From Piece-Wise Constant $\underline{d}$ and $\underline{\lambda}$ fields

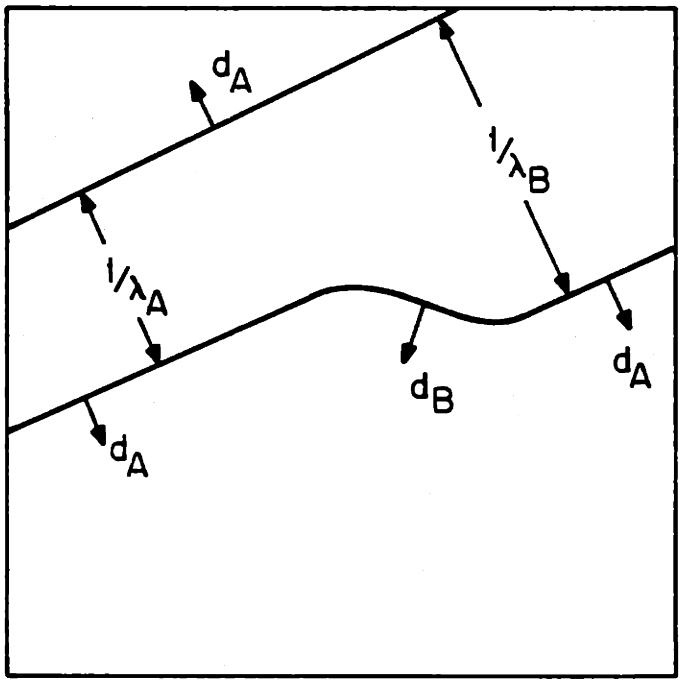

Figure 3.10: Interdependence of Dip and Bed Frequency

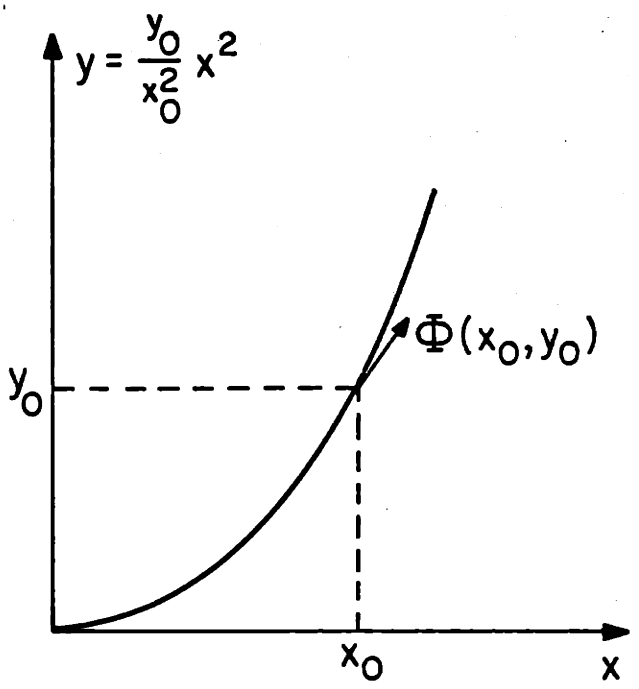

Figure 3.11: Unit Tangent Vector Field Φ for Parabolic Boundaries

The objective of this example is to obtain sample functions varying in dip and bed frequency. The layers have boundaries which are parabolic and the bed thickness should be consistent with this parabolic condition.

To begin, a parabola which passes through the origin and the point (x_0, y_0) has the functional form (see Figure 3.11)

$$y = \frac{y_0}{{x_0}^2} x^2 \tag{3.16}$$

The vector field $\Phi(x, y)$ which consists of unit vectors tangent to the parabola through the origin and the point (x, y) is described by

$$\Phi(x,y) = \frac{x\hat{x}}{(x^2+4y^2)^{\frac{1}{2}}} + \frac{2y\hat{y}}{(x^2+4y^2)^{\frac{1}{2}}} \qquad (x,y) \neq (0,0) \tag{3.17}$$

where $\hat{x}$ and $\hat{y}$ are unit vectors parallel to the x and y axes, respectively.

Dip can be defined, using (3.17), to be the continuously varying vector field which consists of unit vectors which are orthogonal to the vector field $\Phi(x,y)$ at point (x,y). Consequently,

$$d(x,y) = \frac{-2y\hat{x}}{(x^2+4y^2)^{\frac{1}{2}}} + \frac{x\hat{y}}{(x^2+4y^2)^{\frac{1}{2}}} \tag{3.18}$$

One possible method for defining a consistent bed frequency field is to use the divergence operator, $(\nabla\cdot)$, to specify bed thickness variations. Since divergence computes the spatial rate of change of a vector field, using the divergence of Φ to model bed thickness changes is a reasonable way of choosing a bed frequency field which accounts for the changing dip field.

We take

$$\begin{aligned} \lambda(x,y) &= \alpha[\nabla\cdot\Phi(x,y)] + \beta \\ &= \alpha\frac{2x^2+4y^2}{(x^2+4y^2)^{\frac{3}{2}}} + \beta \end{aligned} \tag{3.19}$$

as our bed frequency field. The constants α and β are used to scale and shift the divergence of Φ so that $\lambda(x,y)$ takes reasonable values. This is more completely discussed in the following example.

In this example there are two separate parabolic fields. The data is arranged on a 40×40 Cartesian grid where the position of the i^{th} sample is at $p_i = (k,l)$ where $l = [\text{int}(i/40)+1]$ $k = [i-(l-1)40]$ and $k,l = 1,2,...,40$. The lower parabolic field has its origin at $(0,0)$. That is, the parabolic vertex is at $(0,0)$. The upper parabolic field has its vertex at $(41,20)$. The data which is below the line $y = 21$ is subject to the lower parabolic field. The remaining data is subject to the upper parabolic field.

The choice of the constants α and β are made by evaluating the λ field at its extreme points in value. For example, at $p_i = (6,3)$ we have $\lambda(6,3) = \alpha(.18)+\beta$ and at $p_i = (30,20)$ we have $\lambda(30,20) = \alpha(.027)+\beta$. The points $(6,3)$ and $(30,20)$ have λ values which are essentially the extreme values for the lower parabolic field. Beds cannot be any thinner than one pixel and our neighborhood constraint of 7×7 pixels limits beds to thicknesses less than about 6 pixels.

Thus, choosing $\lambda(6,3) = 1$ and $\lambda(30,20) = .3$ requires

$$\begin{aligned} \alpha &= \frac{(1.-.3)}{(.1768-.0272)} &&= 4.68 \\ \beta &= 1.-(4.68)(.1768) &&= .1726 \end{aligned} \tag{3.20}$$

which should yield beds of thickness 2 pixels at $(1,1)$ and thickness 5 pixels at $(20,20)$. A similar analysis is made for the upper parabolic field. The results of this analysis for our example yields

$$d_i = \begin{cases} \frac{-2l\hat{x}}{(k^2+4l^2)^{\frac{1}{2}}} + \frac{k\hat{y}}{(k^2+4l^2)^{\frac{1}{2}}} & l \leq 20 \\ \frac{-2(l-20)\hat{x}}{((k-41)^2+4(l-20)^2)^{\frac{1}{2}}} + \frac{(k-41)\hat{y}}{((k-41)^2+4(l-20)^2)^{\frac{1}{2}}} & l \geq 21 \end{cases} \tag{3.21}$$

$$\lambda_i = \begin{cases} 4.68[\frac{2k^2+4l^2}{(k^2+4l^2)^{\frac{3}{2}}}] + .1726 & l \leq 20 \\ 4.68[\frac{2(k-41)^2+4(l-20)^2}{((k-41)^2+4(l-20)^2)^{\frac{3}{2}}}] + .1726 & l \geq 21 \end{cases} \tag{3.22}$$

Figures 3.12-3.14 illustrate the results of this example.

3.4 MAP Estimation of $\underline{d}$ and $\underline{\lambda}$

In Sections 3.1 and 3.2 we discussed modeling of binary layered data. The models discussed depend on the interaction of the data field, $\underline{s}$, with the dip and bed frequency fields, $\underline{d}$ and $\underline{\lambda}$. Given a sample function of the distribution in (3.3) it is of interest to estimate the values of $\underline{d}$ and $\underline{\lambda}$ which gave rise to the sample. The purpose of this section is to describe algorithms which produce the MAP estimate of $\underline{d}$ and $\underline{\lambda}$ (see Chapter 2 for a discussion of MAP estimation).

The MAP estimate of dip and bed frequency are denoted by $\hat{\underline{d}}_{MAP}$ and $\hat{\underline{\lambda}}_{MAP}$, respectively. They are obtained by solving

$$\begin{bmatrix} \hat{\underline{d}}_{MAP} \\ \hat{\underline{\lambda}}_{MAP} \end{bmatrix} = \text{Arg Max } p(\underline{s}, \underline{d}, \underline{\lambda}) \tag{3.23}$$

where $p(\underline{s}, \underline{d}, \underline{\lambda})$ is as in (3.3). Since $p(\underline{s}, \underline{d}, \underline{\lambda})$ is a non-linear function of $\underline{d}$ and $\underline{\lambda}$ (3.23) is difficult to solve analytically. In our work we have adopted the method of Simulated Annealing (SA) discussed in Chapter 2 in order to compute these

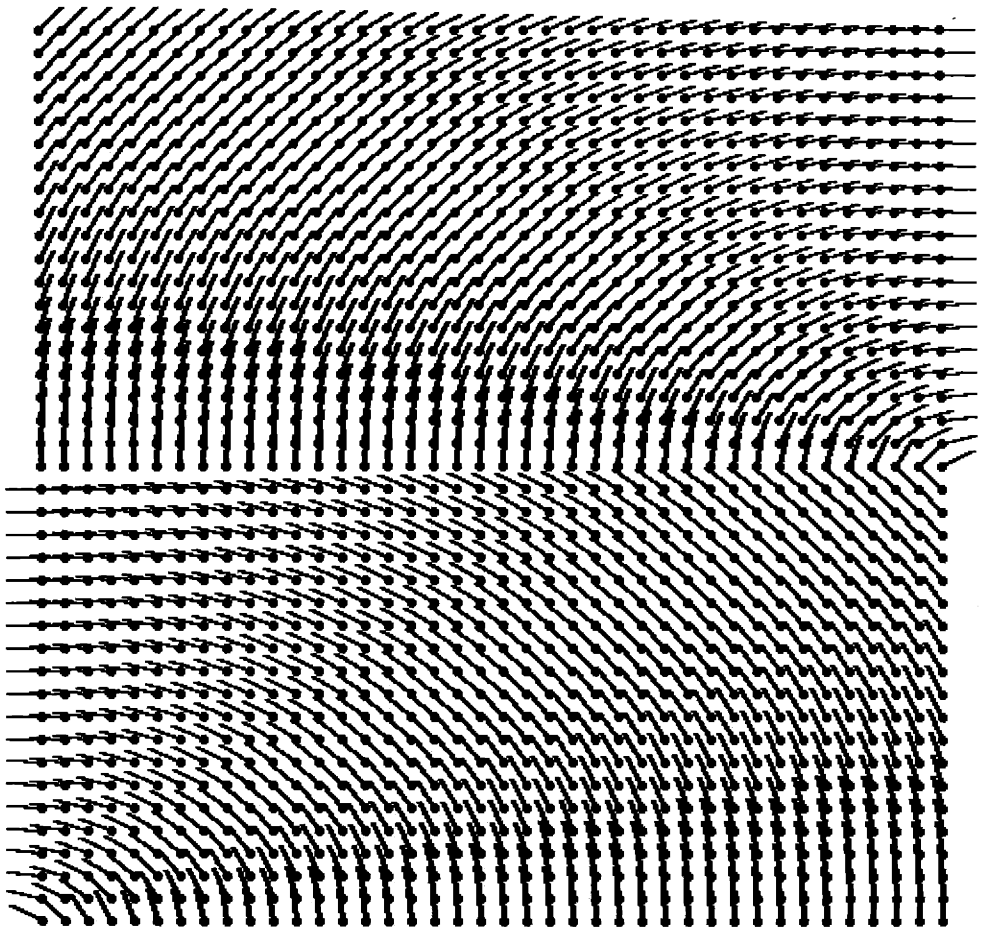

Figure 3.12: Spatially Varying $\underline{d}$ field

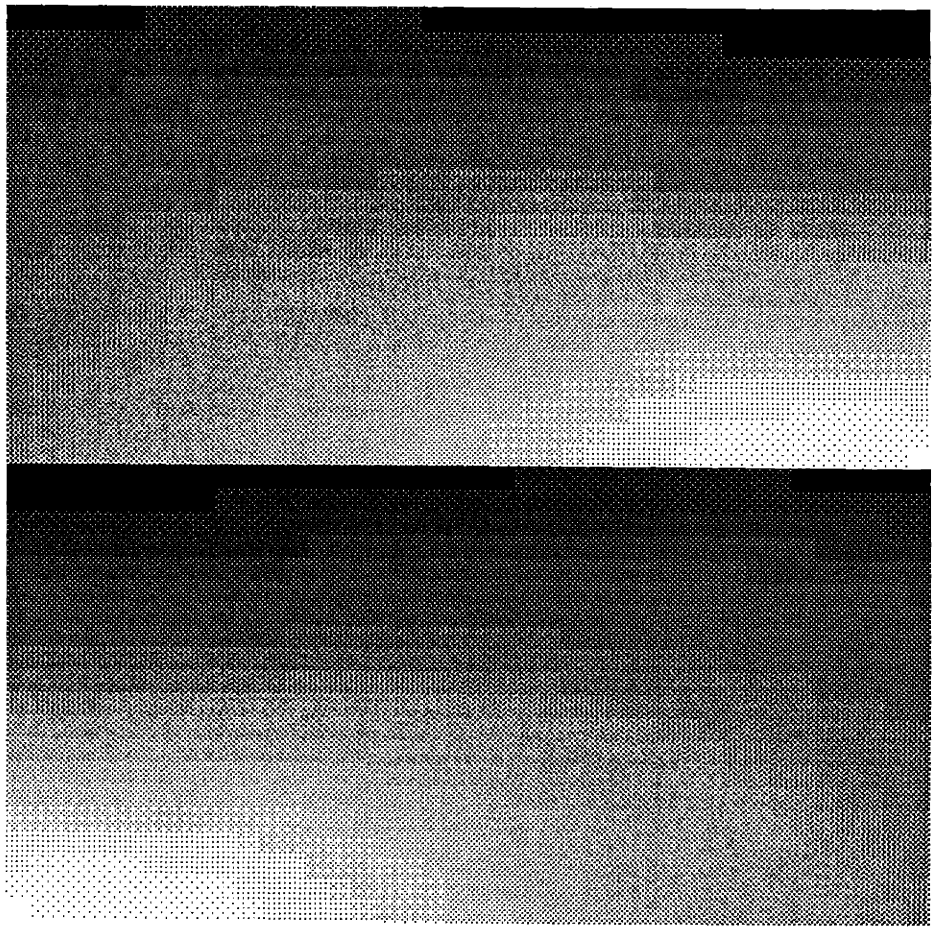

Figure 3.13: Spatially Varying $\underline{\lambda}$ field

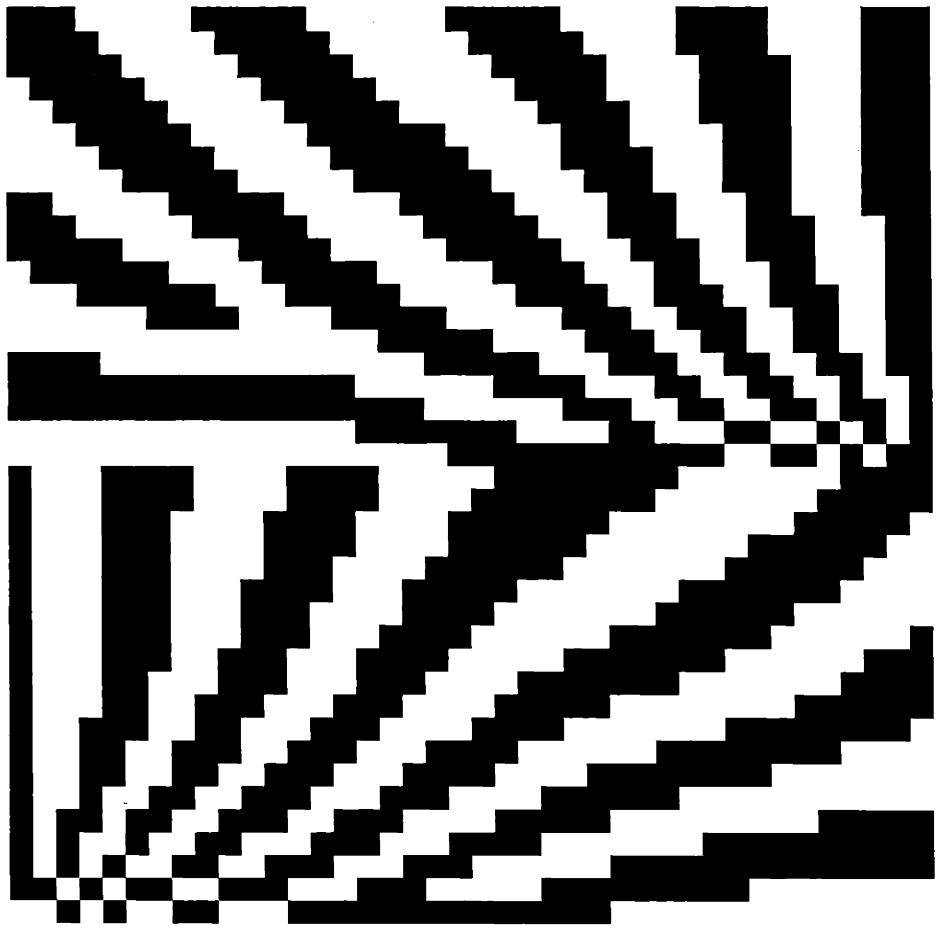

Figure 3.14: Sample Function From Spatially Varying $\underline{d}$ and $\underline{\lambda}$ fields

estimates. The estimates resulting from this process, which under perfect conditions should be the MAP estimates, will be denoted $\hat{\underline{d}}_{SA}$ and $\hat{\underline{\lambda}}_{SA}$.

Before applying SA to the model of Section 3.2 to estimate $\underline{d}$ and $\underline{\lambda}$ there are a few parameters which need to be set. These parameters are the smoothing weights D and Λ, the metric powers γ_d and γ_λ (see (3.4) and (3.5-3.8)), and the neighborhood structures N_i, $\hat{N}_i$, and $\tilde{N}_i$. The choice of these parameters affects the estimates $\hat{\underline{d}}_{SA}$ and $\hat{\underline{\lambda}}_{SA}$ so some experimentation is necessary. The following section illustrates the use of the SA algorithm and develops a rationale for the choice of the model parameters.

3.5 Examples of MAP Estimation for $\underline{d}$ and $\underline{\lambda}$

This section illustrates the use of the SA algorithm for obtaining MAP estimates of $\underline{d}$ and $\underline{\lambda}$. In all the examples the neighborhood structure is as described in the preceding sections. The first set of examples all use the same synthetic data set which was generated using constant dip and bed frequency fields. The various estimates of $\underline{d}$ and $\underline{\lambda}$ obtained from this synthetic data field is used to illustrate the issues involved in the choice of the parameters γ_d, γ_λ, D, and Λ. The second set of examples illustrate estimates for $\underline{d}$ and $\underline{\lambda}$ when the SA algorithm is applied to more complicated synthetic data sets using the values of the parameters obtained from our first series of tests. Finally, the SA algorithm is used to estimate $\underline{d}$ and $\underline{\lambda}$ from a field of synthetic non-binary data followed by a piece of real non-binary data.

Figure 3.15 illustrates the synthetic data set used for the first set of examples. It is identical to the data set in Figure 3.5. The objective of the first set of examples is to illustrate the use and effect of the parameters γ_d, γ_λ, D, and Λ which were introduced in Section 3.2. These examples show estimates of $\underline{d}$ and $\underline{\lambda}$ given the data in Figure 3.15 for, (1) different values of γ_d and γ_λ when $D = \Lambda = 0$, (2) different values of D and Λ when $\gamma_d = \gamma_\lambda = 0$, and, (3) for non-zero values of γ_d, γ_λ, D, and Λ together based on the results in (1) and (2).

To begin, Figures 3.16 and 3.17 illustrate $\hat{\underline{d}}_{SA}$ and $\hat{\underline{\lambda}}_{SA}$ (see notation in Section 3.4) when $\gamma_d = \gamma_\lambda = D = \Lambda = 0$. No smoothing is imposed on the model

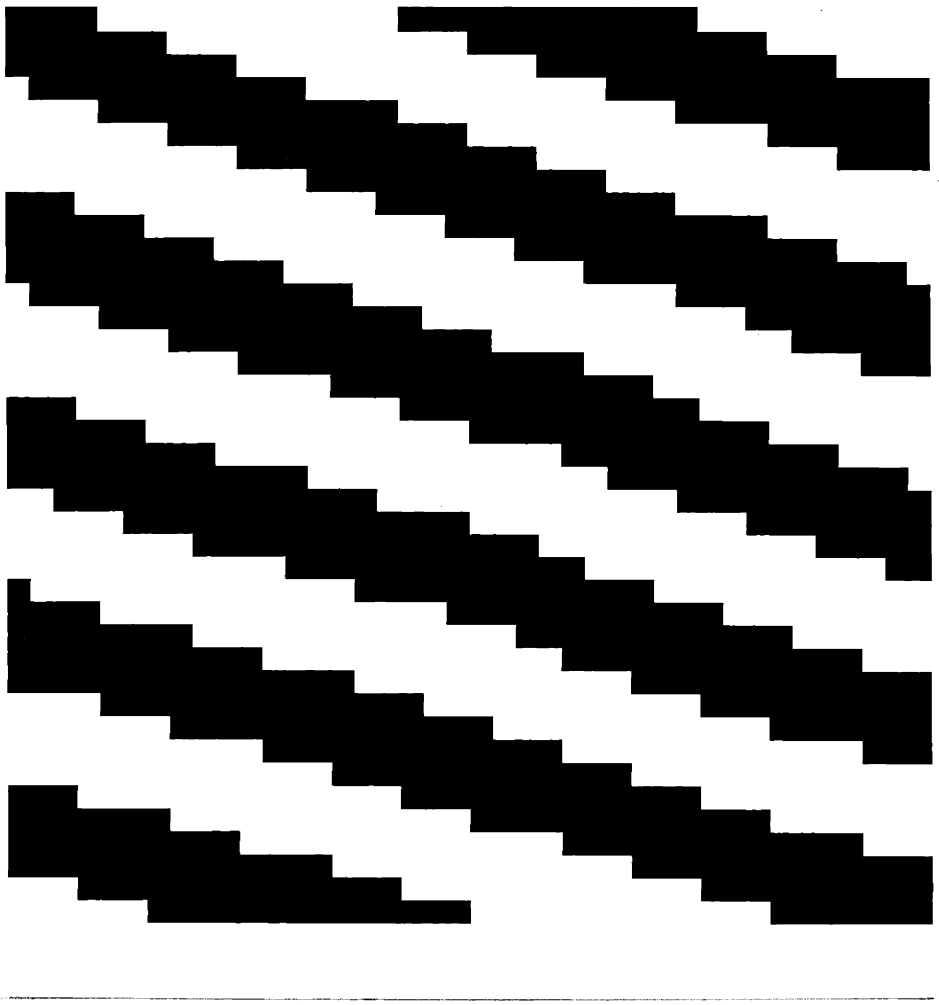

Figure 3.15: Synthetic Data Obtained from Constant Dip and Bed Frequency Fields

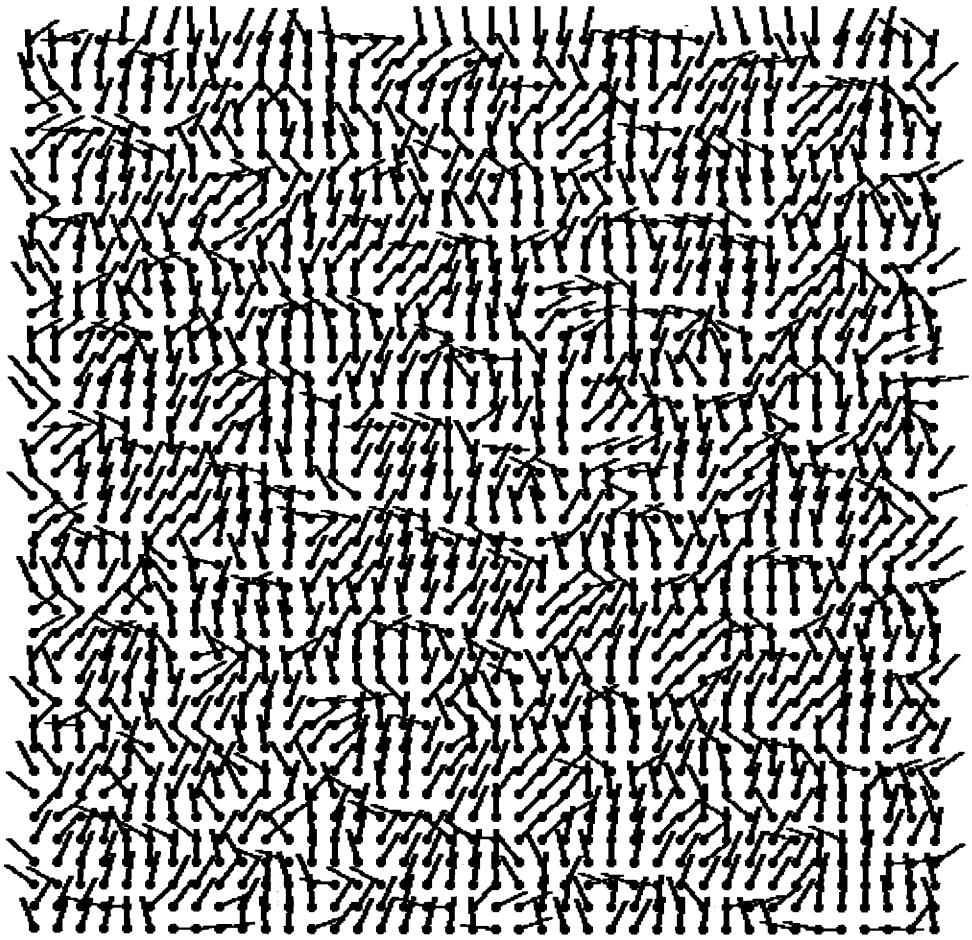

Figure 3.16: Estimate of $\underline{d}$ for $\gamma_d = \gamma_\lambda = 0$ and $D = \Lambda = 0$

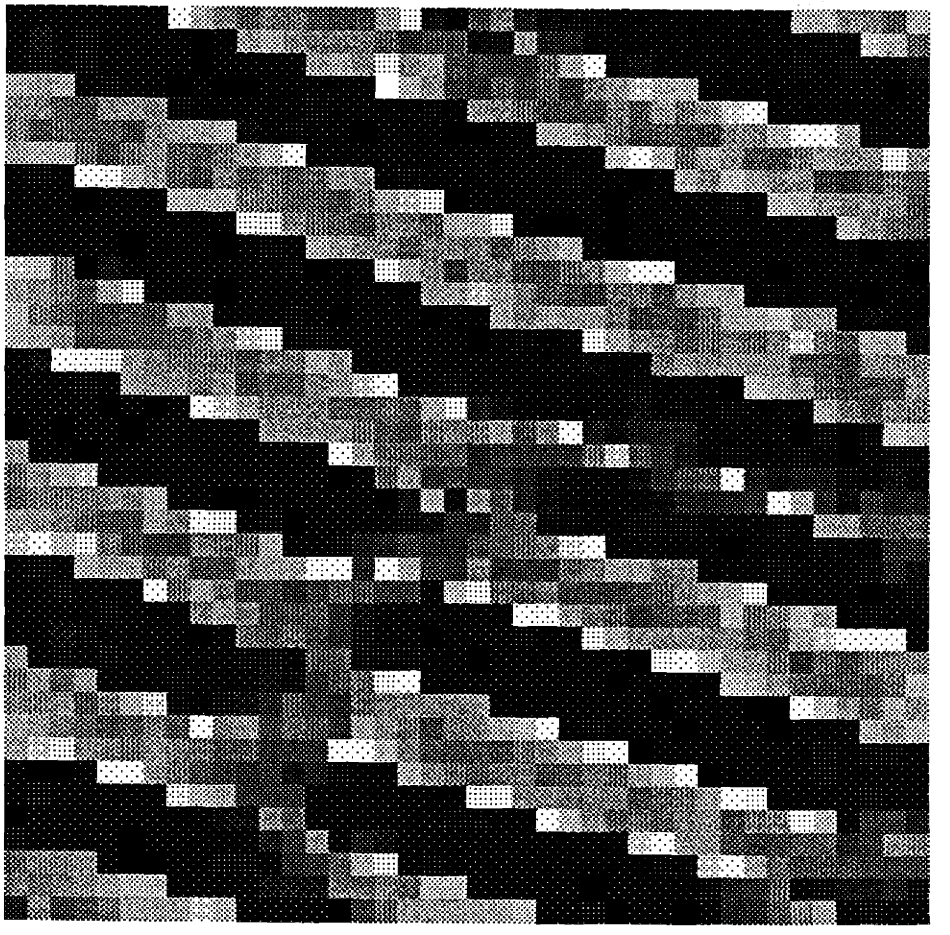

Figure 3.17: Estimate of $\underline{\lambda}$ for $\gamma_d = \gamma_\lambda = 0$ and $D = \Lambda = 0$

except for the averaging which comes from calculating $\bar{d}_{ij}$ and $\bar{\lambda}_{ij}$. The absence of smoothing is apparent in the estimates of $\underline{d}$ and $\underline{\lambda}$. The estimated dip field consists of clumps of dip which nominally point in the correct direction interlaced with dip which point orthogonally to these clumps. Careful examination of the orthogonal dips shows that these dips are only found on (or very near) bed boundaries and point along the direction of the bed boundary.

The estimated bed frequency field also displays features which correspond to the bed boundaries of the data. The bed frequency field consists of alternating beds of high bed frequency (light colored beds) and low bed frequency (dark colored beds). The explanation for the effects seen in the estimates for $\underline{d}$ and $\underline{\lambda}$ are associated with an aspect of the model which is described below.

The model as described in Section 3.2 is specified by an energy function. The cross-section of the energy function along the direction of dip is as illustrated in Figure 3.18. What the model expects, therefore, is that the "correlation" between the data point at element i and its neighbors is positive within a distance of $1/2\lambda$. Thus, for a data point at or near a bed boundary the model is bound to be confused concerning the value of λ or the value of d.

Figures 3.19 and 3.20, 3.21 and 3.22, and 3.23 and 3.24 illustrate estimates of $\underline{d}$ and $\underline{\lambda}$ given the data in Figure 3.15 for various values of γ_d and γ_λ. In Figure 3.19 and 3.20, $\gamma_d = \gamma_\lambda = 4$ and $D = \Lambda = 0$. In Figure 3.21 and 3.22, $\gamma_d = \gamma_\lambda = 16$ and $D = \Lambda = 0$. Finally, in Figure 3.23 and 3.24, $\gamma_d = \gamma_\lambda = 64$ and $D = \Lambda = 0$.

Note first that the estimates in Figures 3.19 and 3.20 are superior to those in Figures 3.16 and 3.17. However, in Figures 3.21, 3.22, and Figures 3.23, 3.24 we see that the estimates of $\underline{d}$ and $\underline{\lambda}$ become poorer as the values for γ_d and γ_λ get larger. The estimate for each field tend to consist of clumps which are close in value (a common effect of smoothing in MRF's) and this phenomenon worsens as γ_d and γ_λ get bigger. The best of the three sets of estimate is the set illustrated in Figure 3.19 and 3.20. Here there are many individual spurious dip and bed frequency estimates (although not nearly as many as in Figures 3.16 and 3.17). However, since these do not clump together they do not look like features of the data. This allows for post processing on the smoothing terms (for D, $\Lambda \neq 0$) methods to improve the

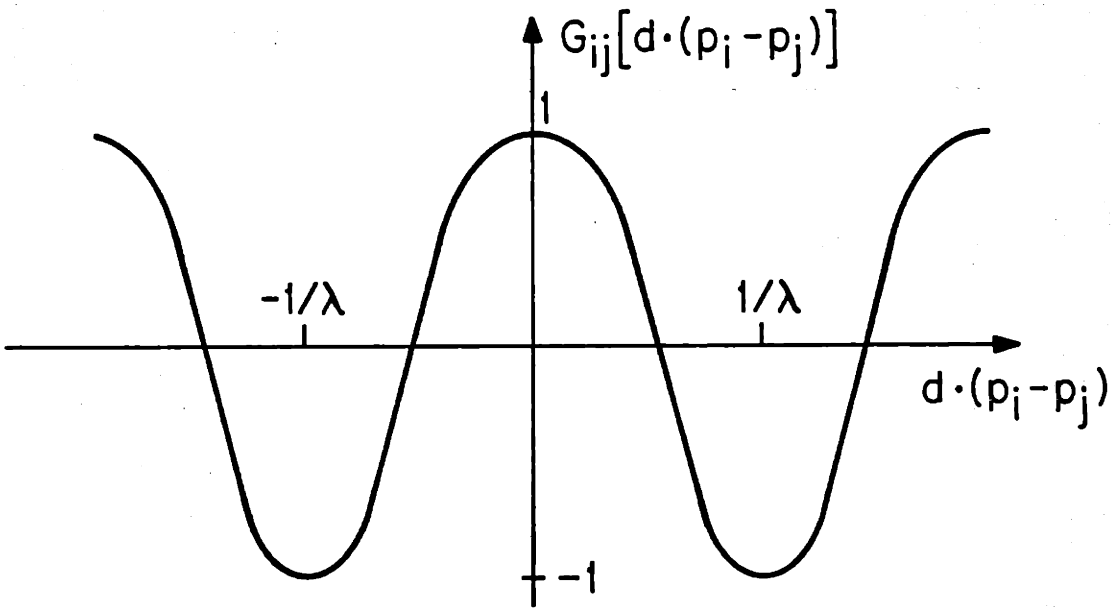

Figure 3.18: Cross-Section of G-function

estimates.

Figures 3.25 and 3.26, 3.27 and 3.28, and 3.29 and 3.30 illustrate estimates of $\underline{d}$ and $\underline{\lambda}$ given the data in Figure 3.15 for various values of D and Λ. In the three sets of figures, D and Λ are set to the values 1, 10, and 100, respectively. For each of these figures $\gamma_d = \gamma_\lambda = 0$.

As in Figures 3.19-3.24, Figures 3.19 and 3.20 ($\gamma_d = \gamma_\lambda = D = \Lambda = 0$) and Figures 3.25-3.30 illustrate an increasing and then decreasing performance of the SA algorithm as a function of the parameters. Figure 3.25 shows an estimate of $\underline{d}$ which has the correct character, however, it is peppered with large regions of incorrect local estimates. The estimate of $\underline{\lambda}$ depicted in Figure 3.26 is definitely unacceptable and is similar to the $\underline{\lambda}$ estimate of Figure 3.17.

In contrast to Figures 3.25 and 3.26 are Figures 3.27 and 3.28. Here both the estimate of $\underline{d}$ and $\underline{\lambda}$ are greatly improved. The increase in the value of D and Λ has smoothed out the estimate of the $\underline{\lambda}$ field. Correspondingly, the $\underline{d}$ field estimate is greatly improved.

Figures 3.29 and 3.30 show the results of choosing values for D and Λ which are too large. The smoothing terms start to dominate the data terms of the model here. As can be seen, the estimate of $\underline{d}$ is flawed by several regions which are highly biased. However,the regions do display a high degree of smoothness as the flow into one another. The estimate of the $\underline{\lambda}$ field appears less affected by the increase in the value of Λ.

The results illustrated in Figures 3.19-3.30 have provided some insight concerning appropriate values for the parameters D, Λ, γ_d, and γ_λ. The results are now applied to data fields which are more complicated than the data field illustrated in Figure 3.15. In Figure 3.31 the data set of Figure 3.9 is reproduced. Figures 3.32 and 3.33 are estimates of the $\underline{d}$ and $\underline{\lambda}$ fields given this data. The parameters are set according to the results of the previous experiments, $D = \Lambda = 10$ and $\gamma_d = \gamma_\lambda = 4$. These parameter values provided good results when applied independently to the data field exhibiting characteristics of constant dip and bed frequency.

As might be expected, Figures 3.32 and 3.33 illustrate reasonably good estimate of the $\underline{d}$ and $\underline{\lambda}$ fields. However, there is a bit too much smoothing observed, especially

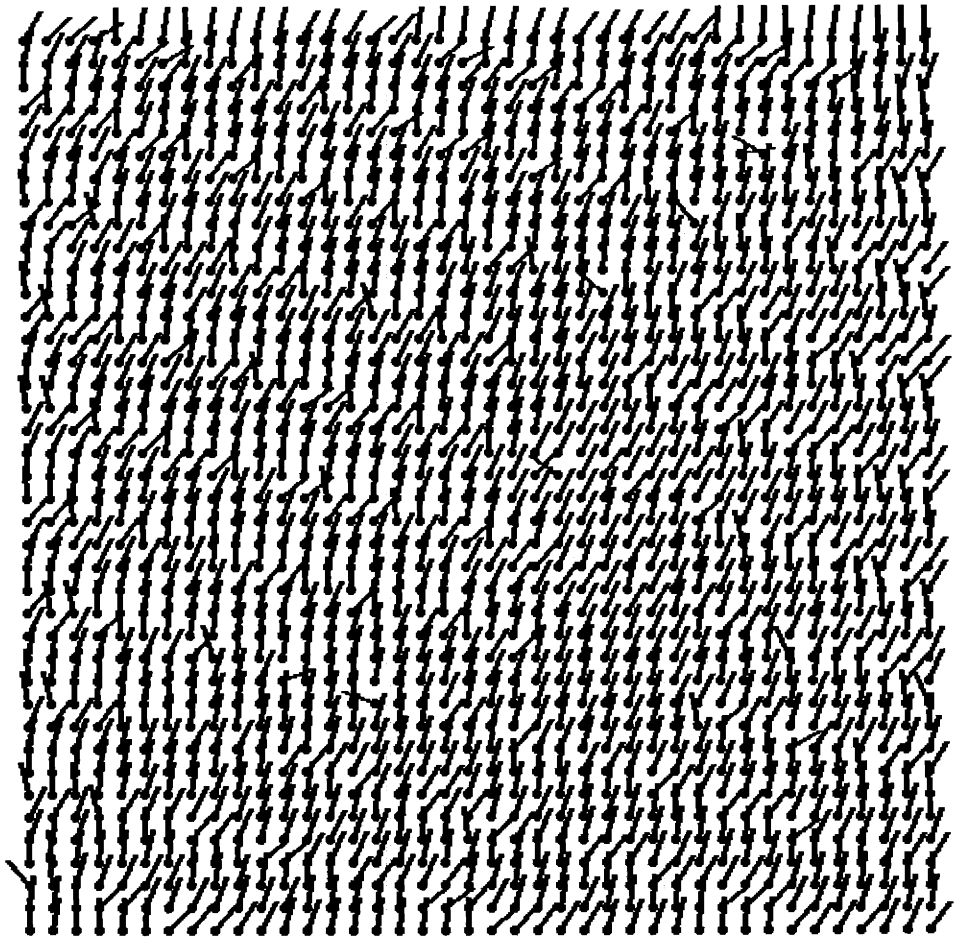

Figure 3.19: Estimate of $\underline{d}$ for $\gamma_d = \gamma_\lambda = 4$ and $D = \Lambda = 0$

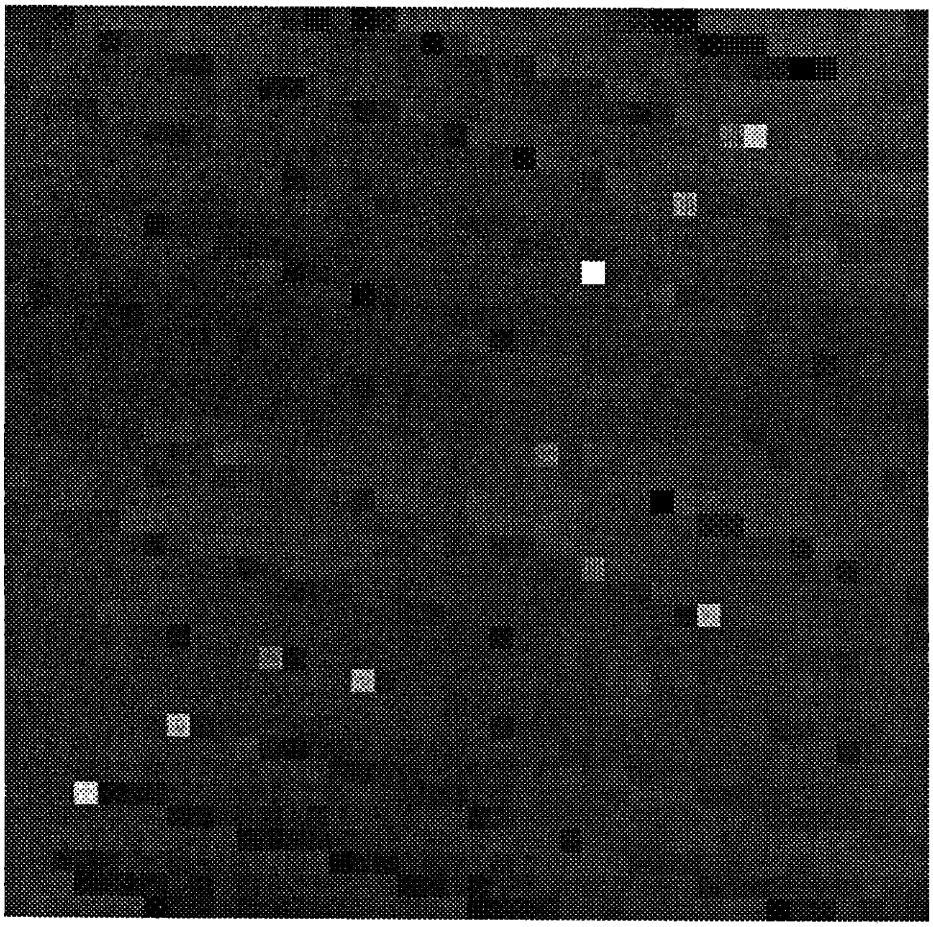

Figure 3.20: Estimate of $\underline{\lambda}$ for $\gamma_d = \gamma_\lambda = 4$ and $D = \Lambda = 0$

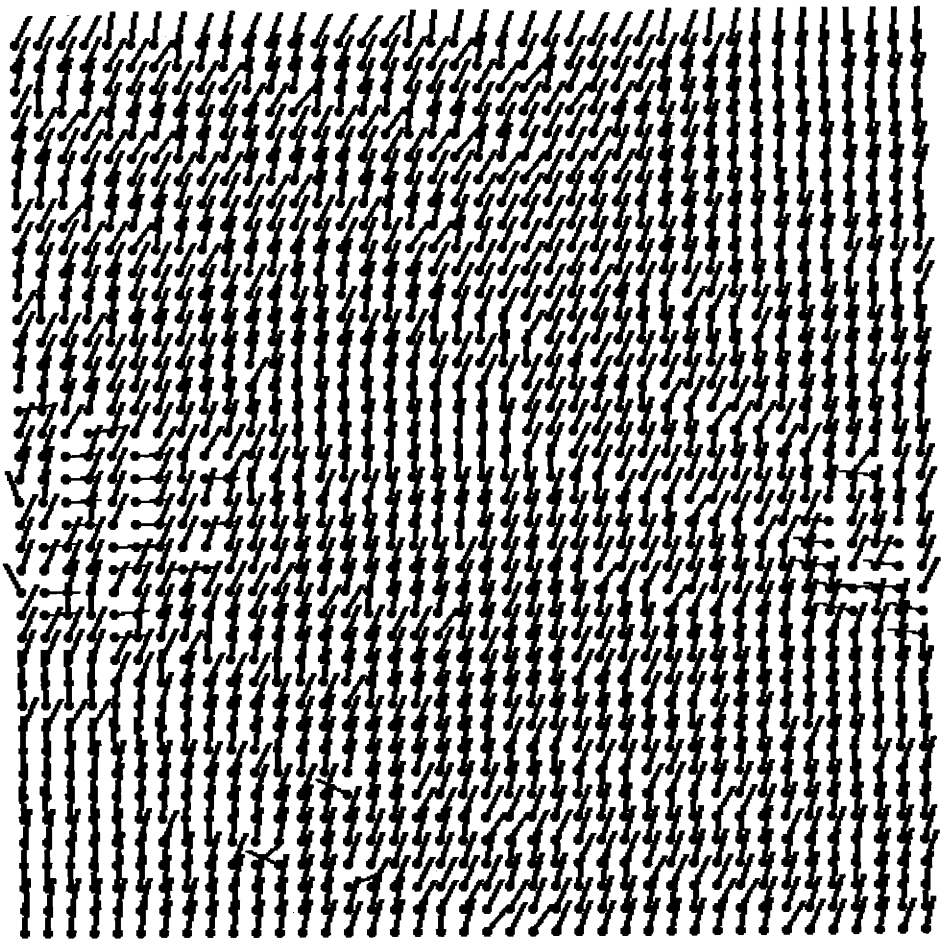

Figure 3.21: Estimate of $\underline{d}$ for $\gamma_d = \gamma_\lambda = 16$ and $D = \Lambda = 0$

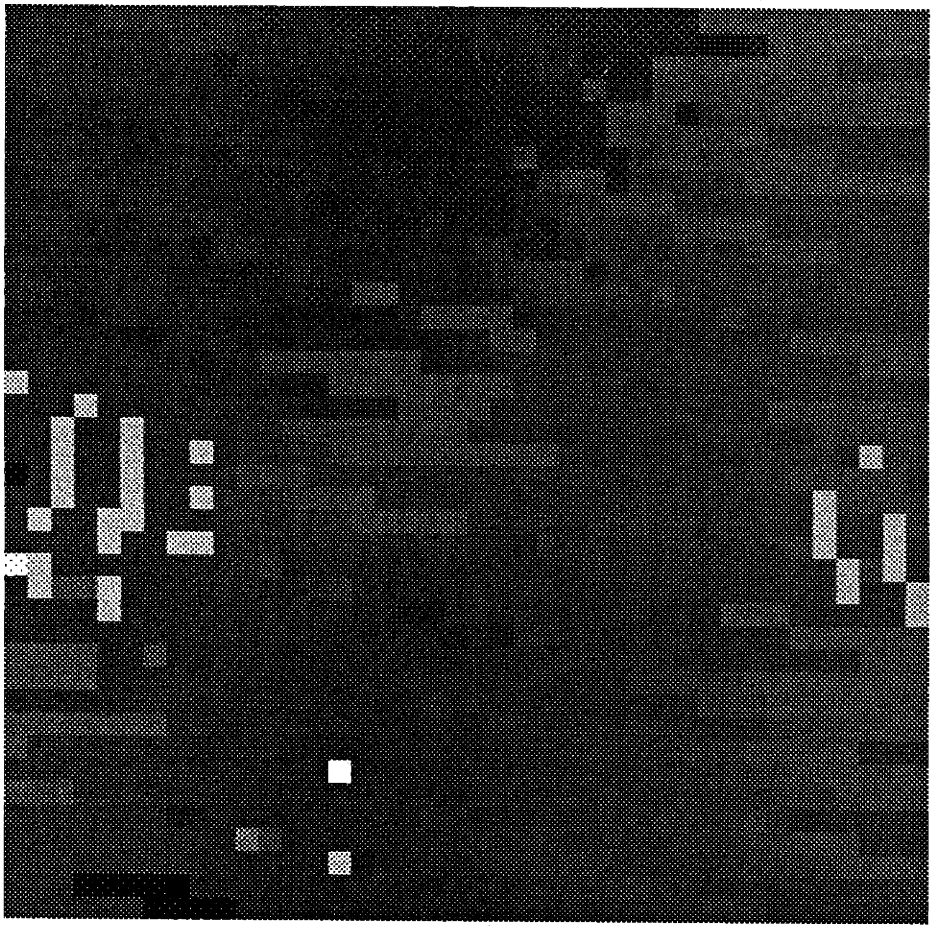

Figure 3.22: Estimate of $\underline{\lambda}$ for $\gamma_d = \gamma_\lambda = 16$ and $D = \Lambda = 0$

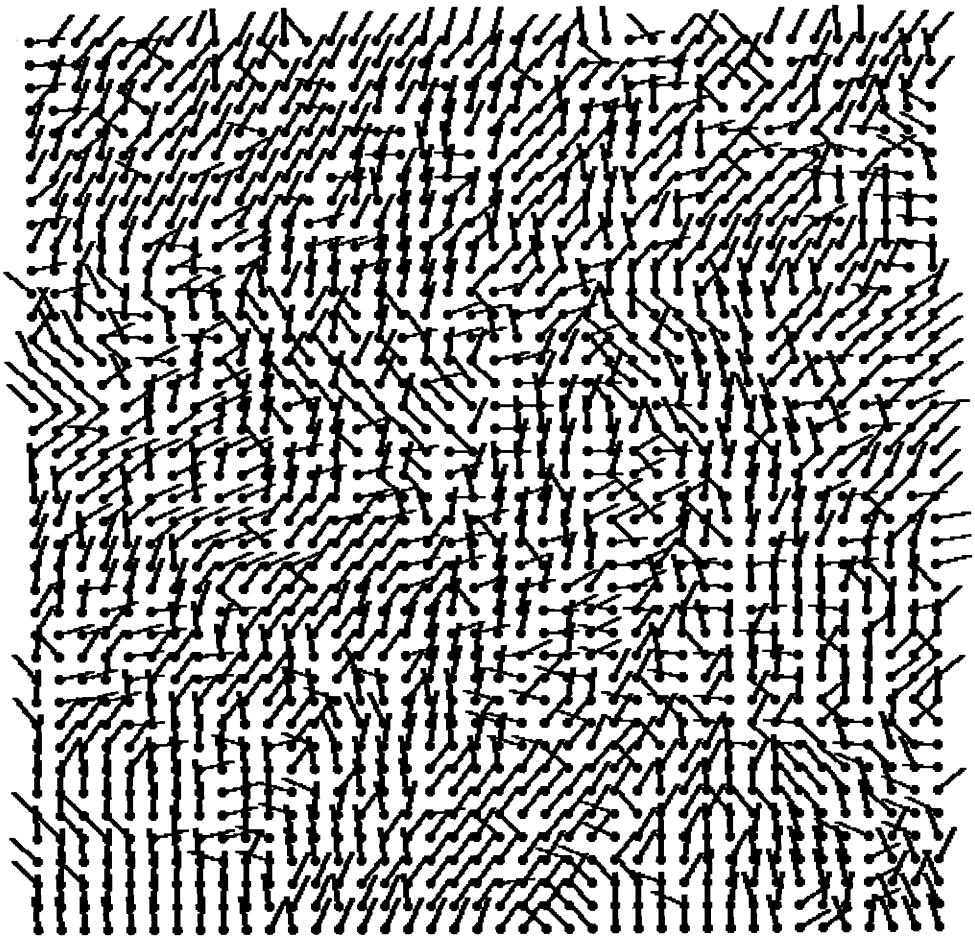

Figure 3.23: Estimate of $\underline{d}$ for $\gamma_d = \gamma_\lambda = 64$ and $D = \Lambda = 0$

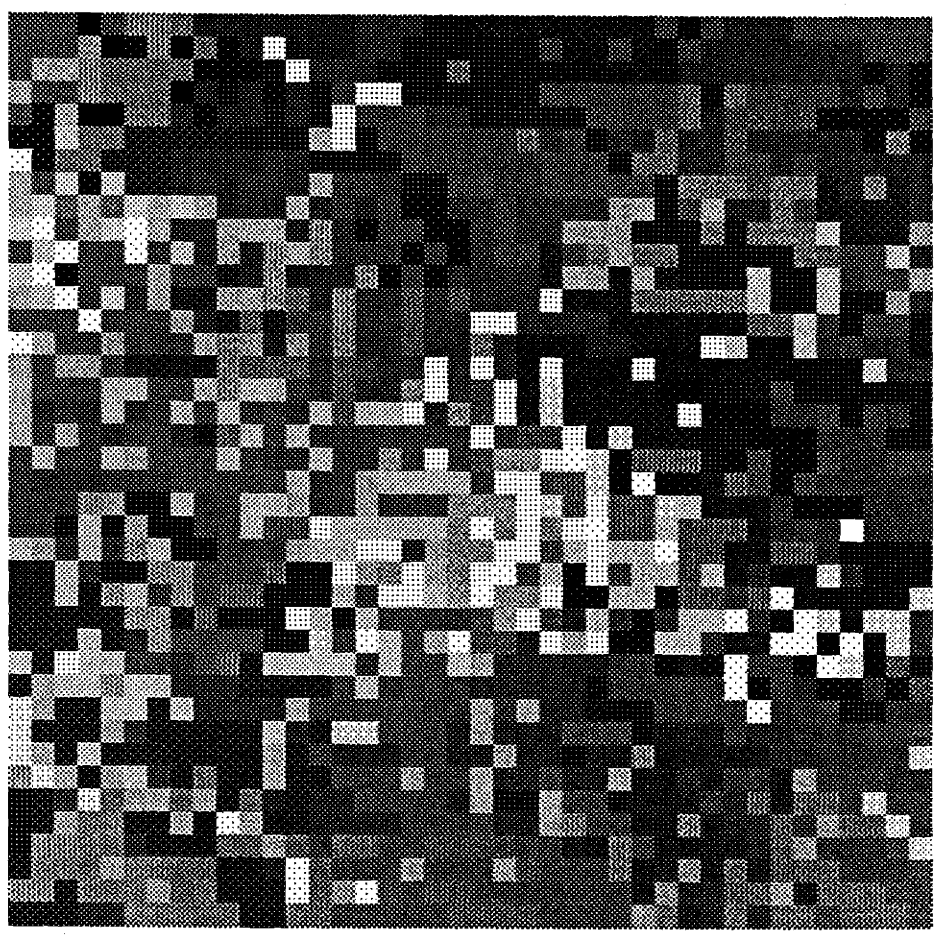

Figure 3.24: Estimate of $\underline{\lambda}$ for $\gamma_d = \gamma_\lambda = 64$ and $D = \Lambda = 0$

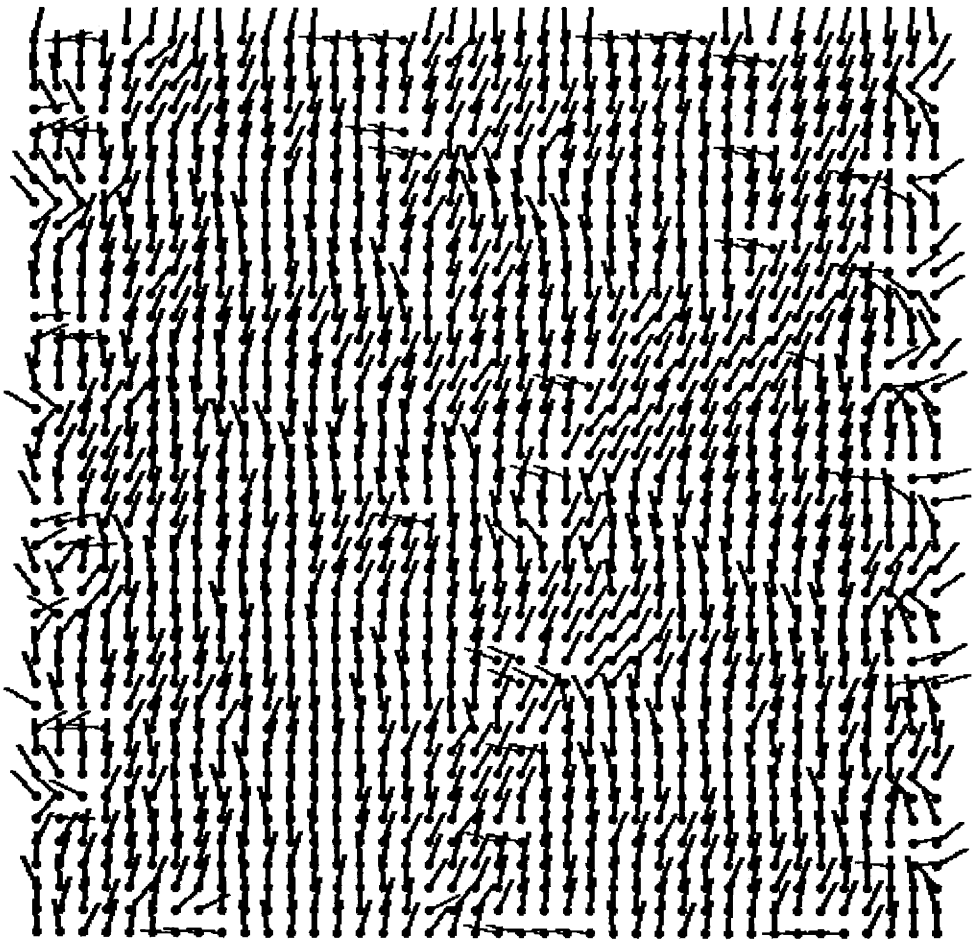

Figure 3.25: Estimate of $\underline{d}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 0$

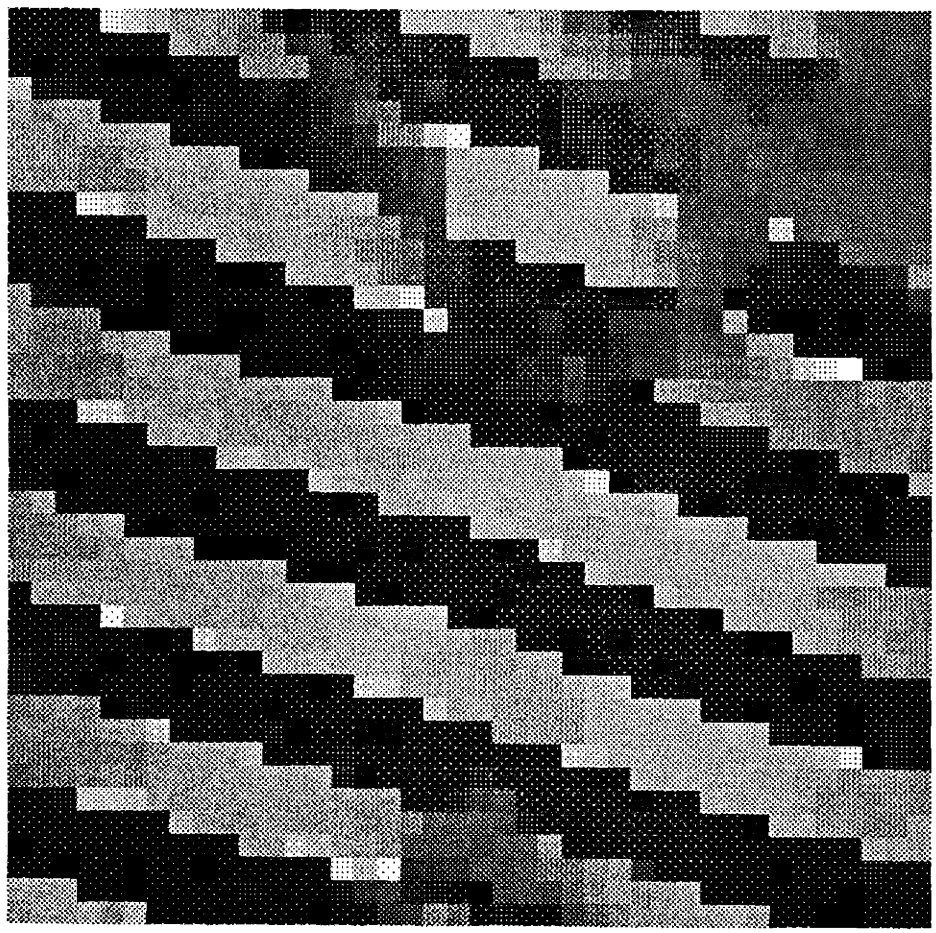

Figure 3.26: Estimate of $\underline{\lambda}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 0$

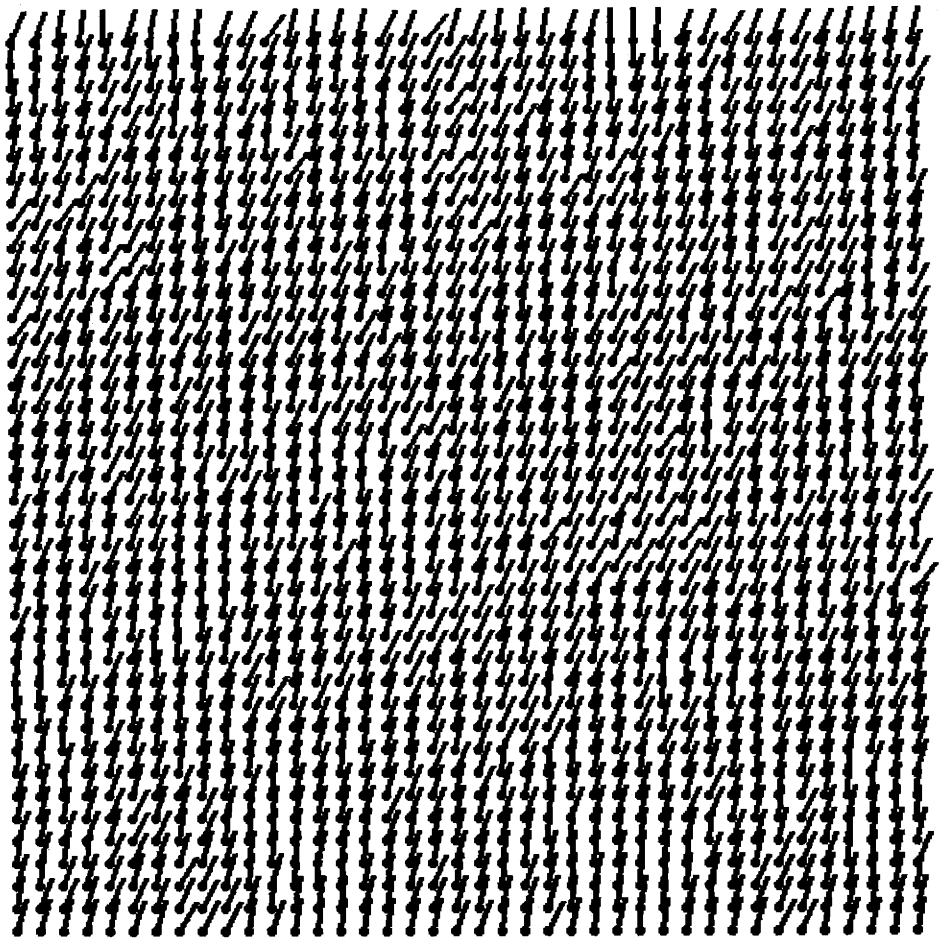

Figure 3.27: Estimate of $\underline{d}$ for $D = \Lambda = 10$ and $\gamma_d = \gamma_\lambda = 0$

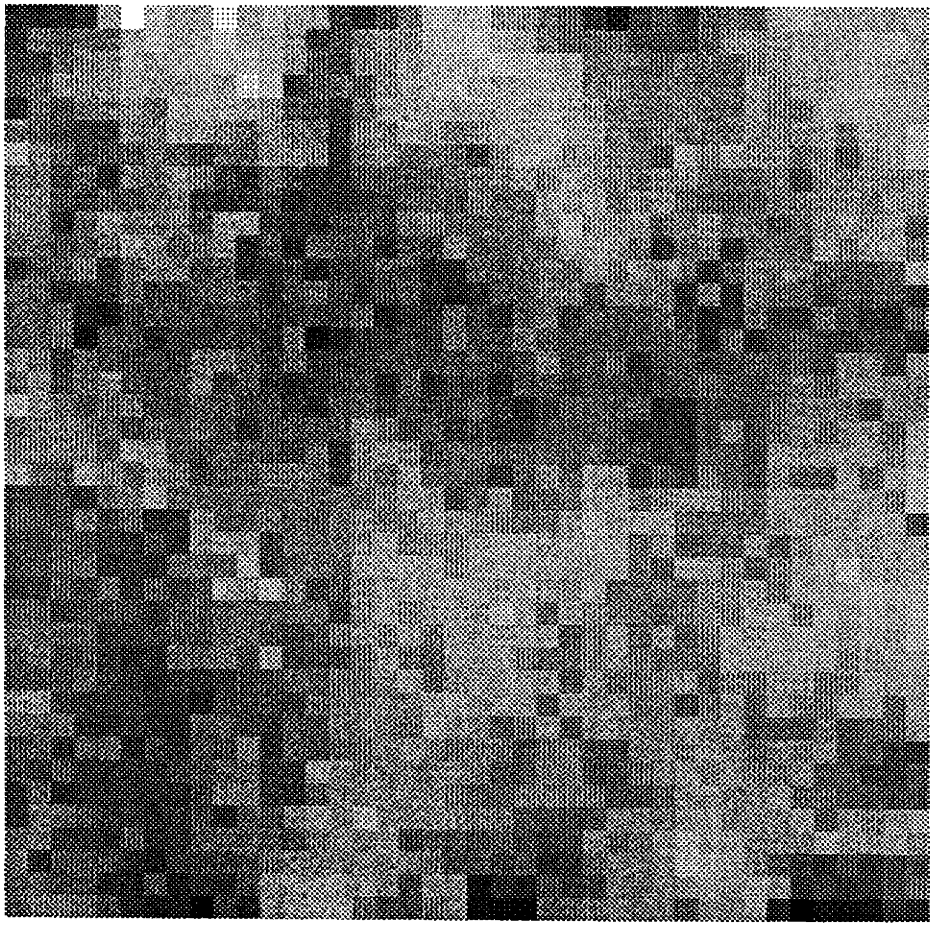

Figure 3.28: Estimate of $\underline{\lambda}$ for $D = \Lambda = 10$ and $\gamma_d = \gamma_\lambda = 0$

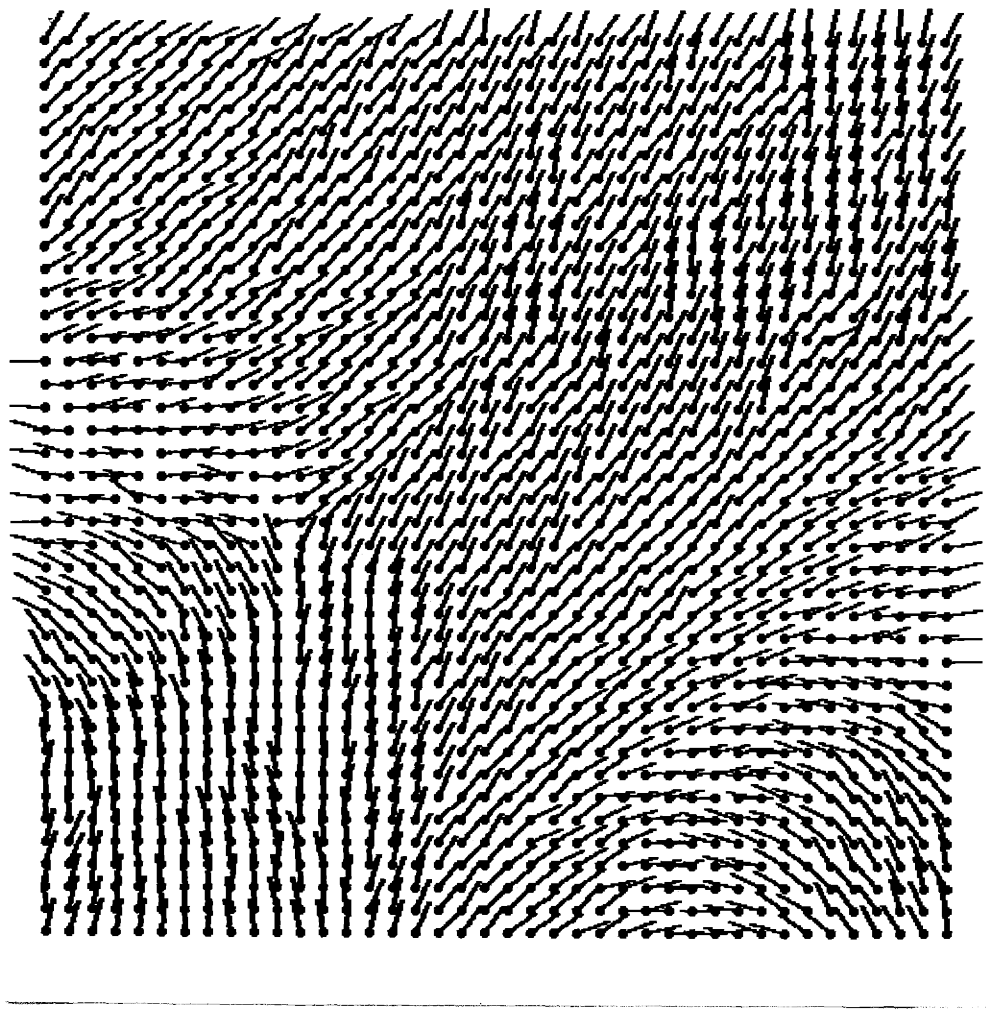

Figure 3.29: Estimate of $\underline{d}$ for $D = \Lambda = 100$ and $\gamma_d = \gamma_\lambda = 0$

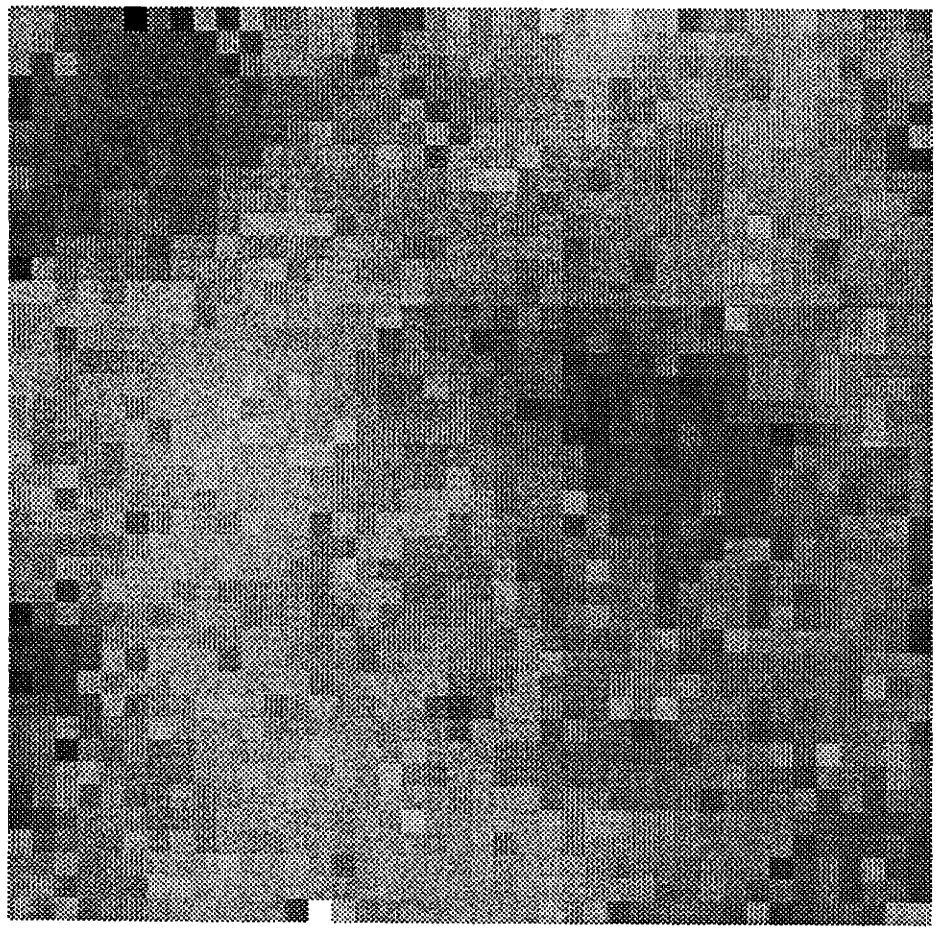

Figure 3.30: Estimate of $\underline{\lambda}$ for $D = \Lambda = 100$ and $\gamma_d = \gamma_\lambda = 0$

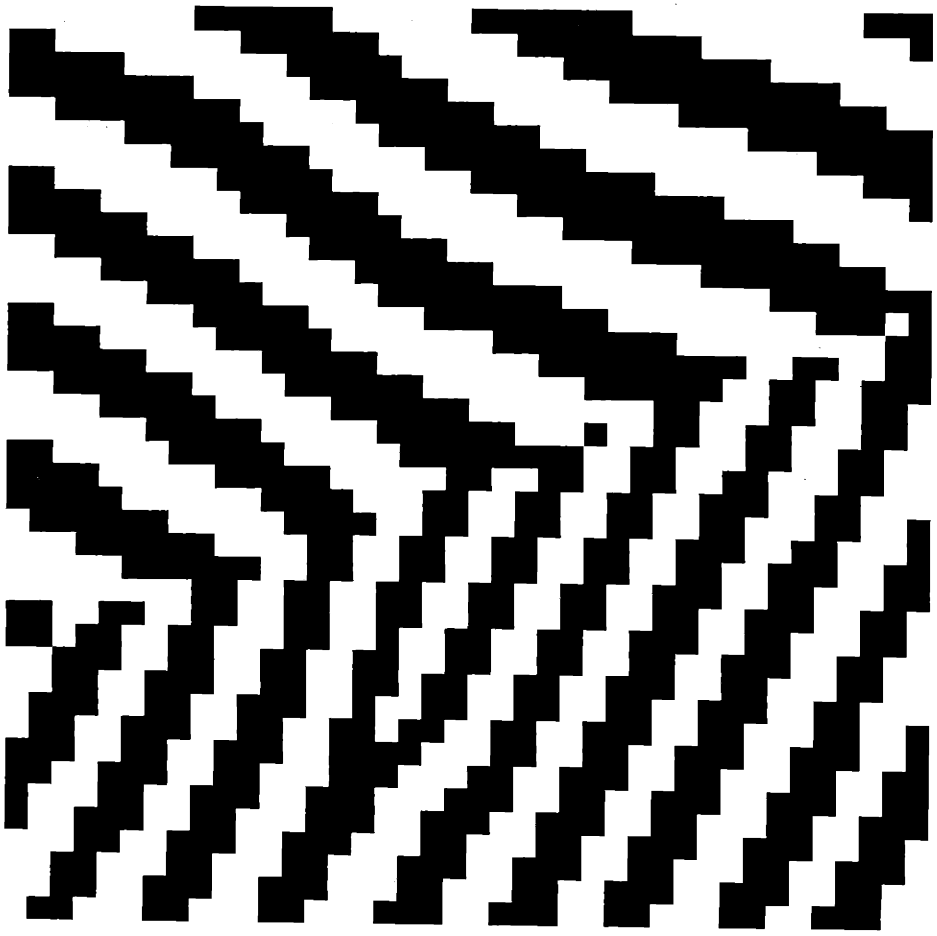

Figure 3.31: Sample Data Generated by Piece-Wise Constant $\underline{d}$ and $\underline{\lambda}$ fields

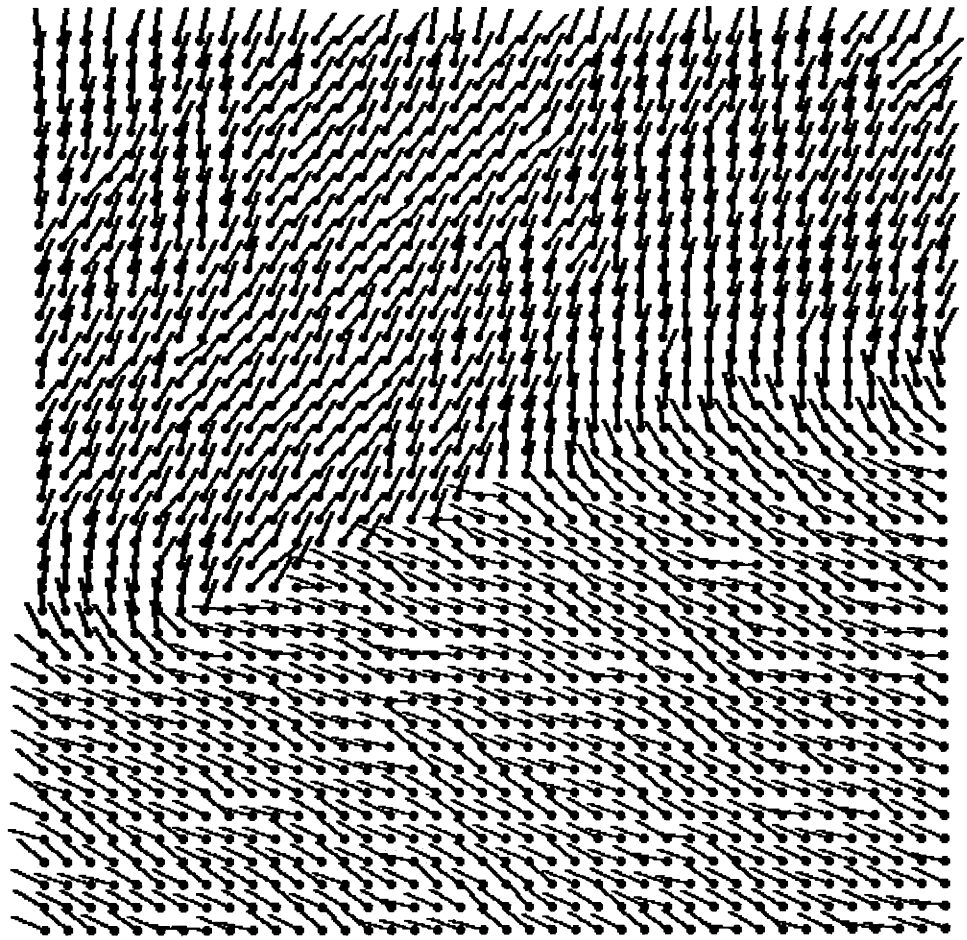

Figure 3.32: Estimate of $\underline{d}$ for $D = \Lambda = 10$ and $\gamma_d = \gamma_\lambda = 4$

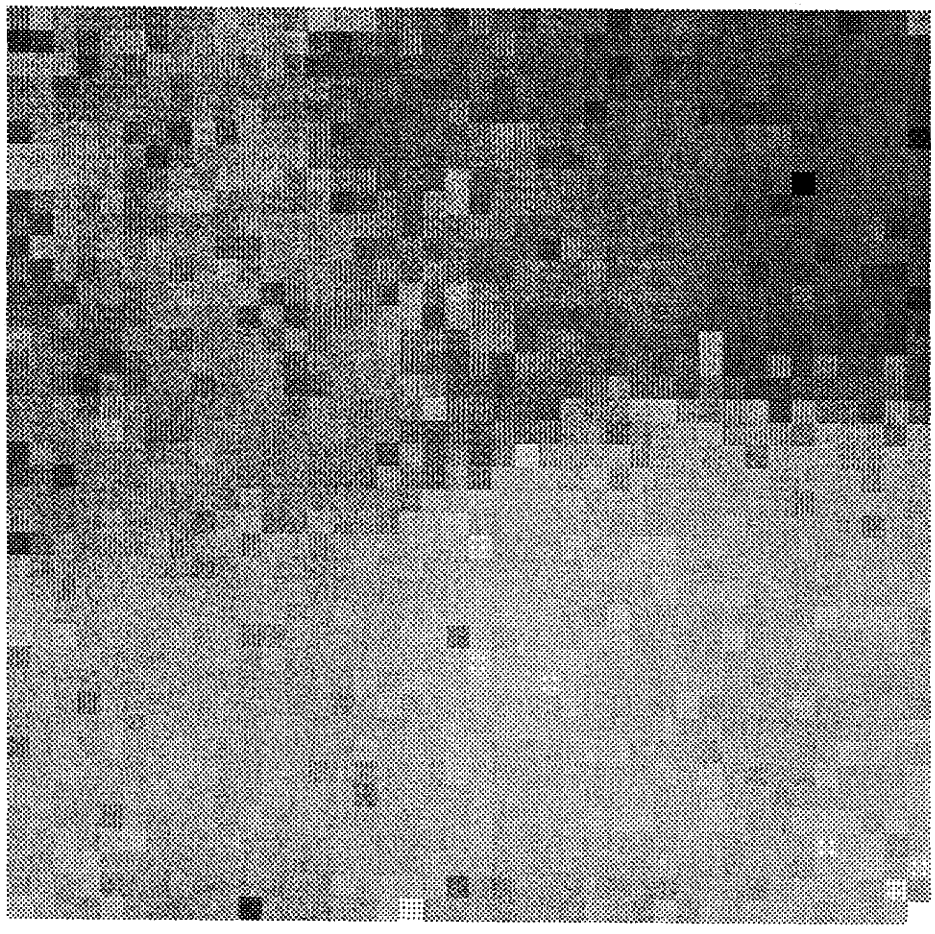

Figure 3.33: Estimate of $\underline{\lambda}$ for $D = \Lambda = 10$ and $\gamma_d = \gamma_\lambda = 4$

near the boundary which separates the two major regions of dip and bed frequency. This is not surprising considering that the values of the parameters D, Λ, γ_d, and γ_λ are now working simultaneously. Previously, either the D and Λ parameters were in effect or the parameters γ_d and γ_λ were in effect. These parameters were not used together. Consequently, the model is a bit oversmoothed.

Figures 3.34 and 3.35 verify the fact that the model is over-smoothed in Figures 3.32 and 3.33. Here the parameters take the values $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$. As can be observed the estimates of $\underline{d}$ and $\underline{\lambda}$ are greatly improved about the boundary between the major regions of dip and bed frequency.

Figures 3.37 and 3.38 show the results of estimating $\underline{d}$ and $\underline{\lambda}$ from the data set in Figure 3.36. As above the parameters are $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$. The estimates illustrated in Figures 3.37 and 3.38 are remarkably good. Comparison between these figures and Figures 3.12 and 3.13 shows that the SA algorithm estimates capture the nature and character of the data.

At this point it can be concluded that the model proposed in this chapter is useful as a tool for solving some inverse problems. These are inverse problems which attempt to determine the dip and bed frequency characteristics for binary layered data. A question of interest concerns the ability of the techniques discussed so far to handle non-binary data sets. To address this question, the following two examples illustrate attempts by the SA algorithm to estimate $\underline{d}$ and $\underline{\lambda}$ from non-binary data.

The first example is similar in nature to the binary data set already discussed. It is generated using the Metropolis algorithm and the specifications given in (3.14) and (3.15) (discussed in Section 3.3) with a modification. This modification simply removes the constraint that the data must be binary and allows the data to take a continuum of values within the range [-1,1]. The results of this change can be seen in Figure 3.39.

Figure 3.39 was generated using exactly the same $\underline{d}$ and $\underline{\lambda}$ fields as were used to generate Figure 3.9. The synthetic data in Figure 3.39 appears very similar in structure to the data in Figure 3.9. The difference is that the new data is non-binary and noisy looking. The apparent noise is a feature of the Metropolis algorithm which

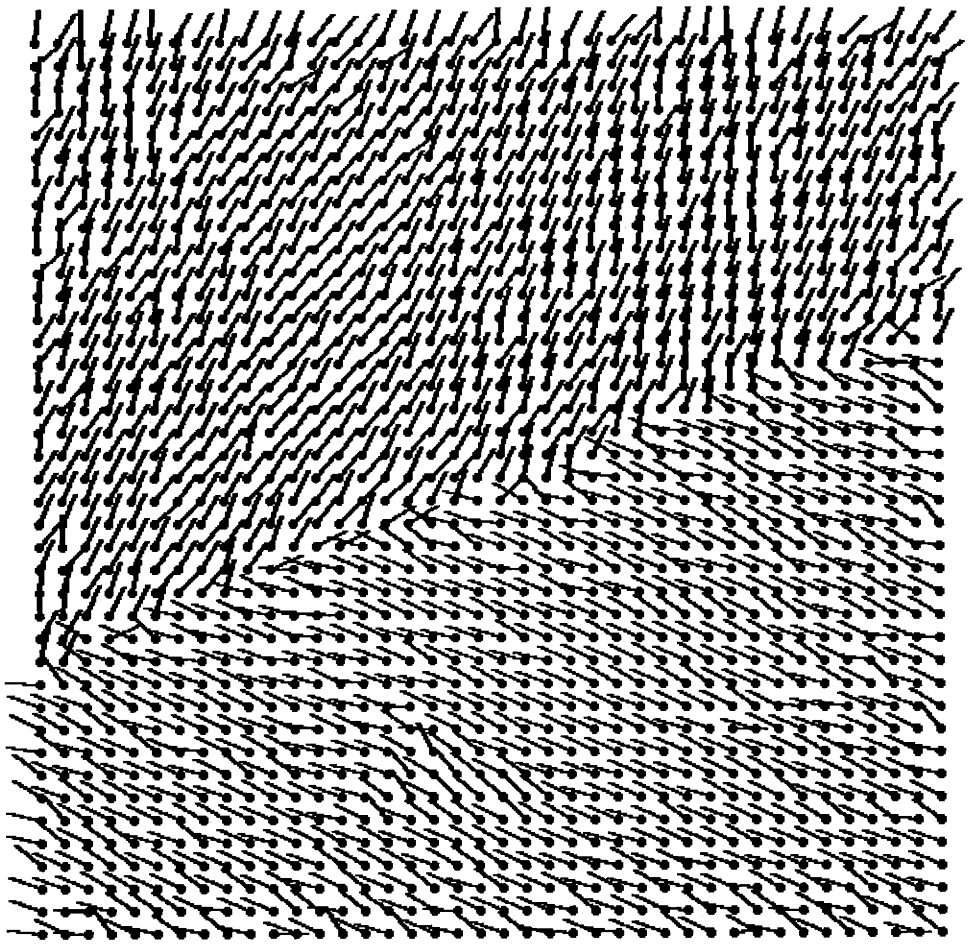

Figure 3.34: Estimate of $\underline{d}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

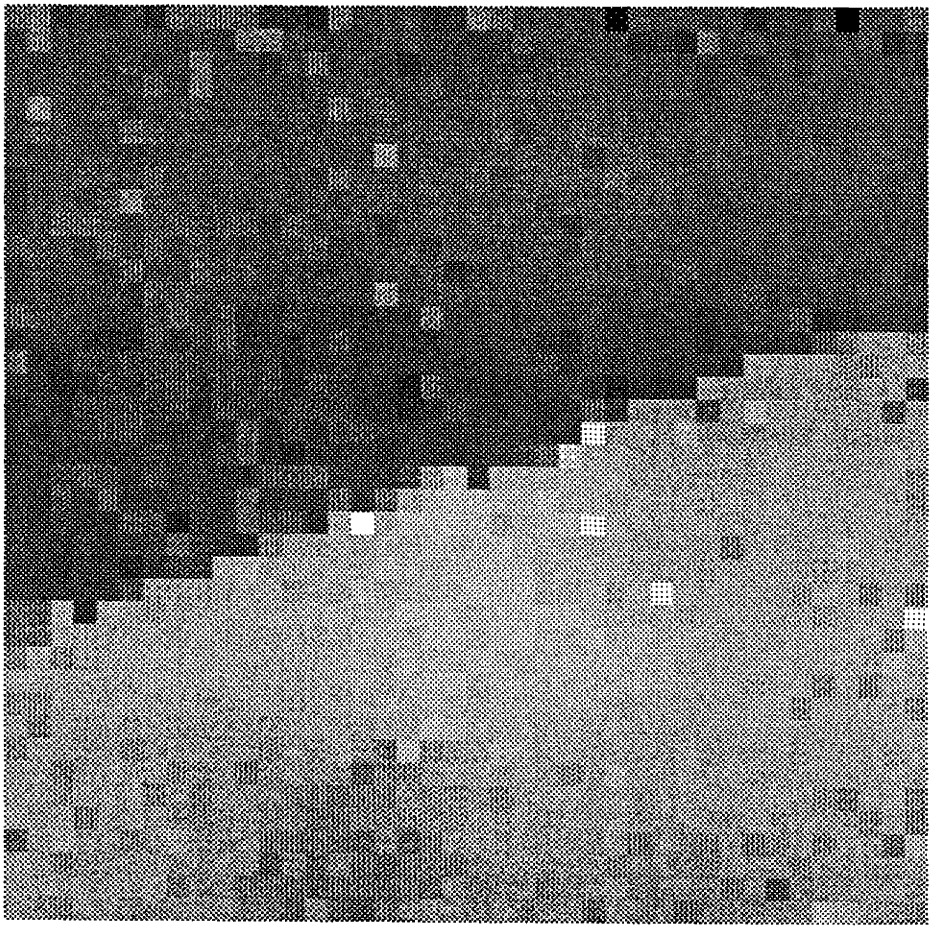

Figure 3.35: Estimate of $\underline{\lambda}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

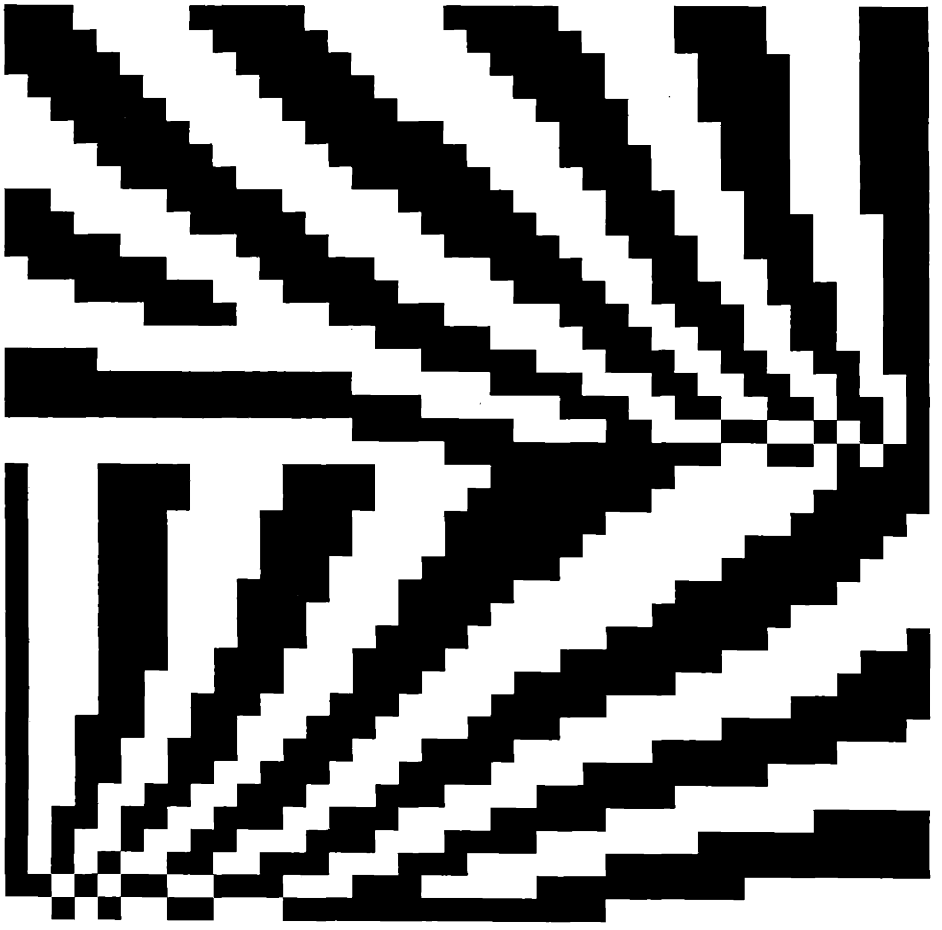

Figure 3.36: Sample Data Generated by Continuously Varying $\underline{d}$ and $\underline{\lambda}$ fields

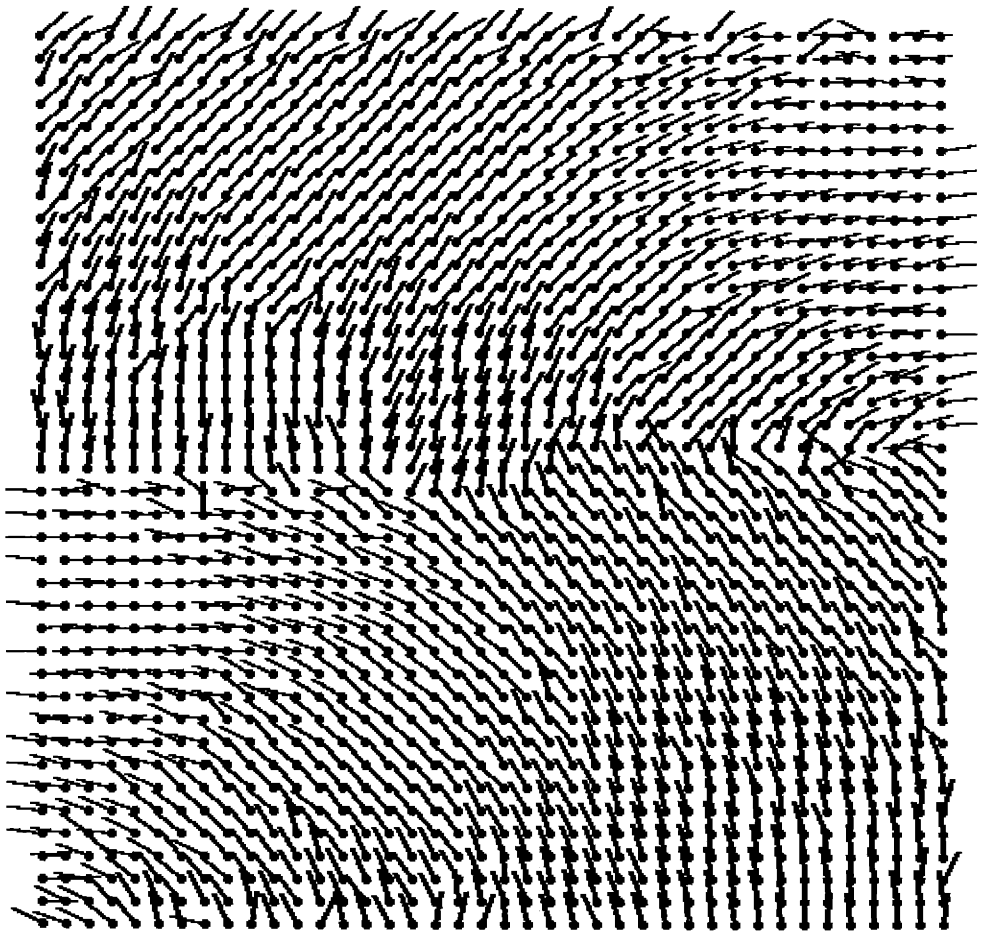

Figure 3.37: Estimate of $\underline{d}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

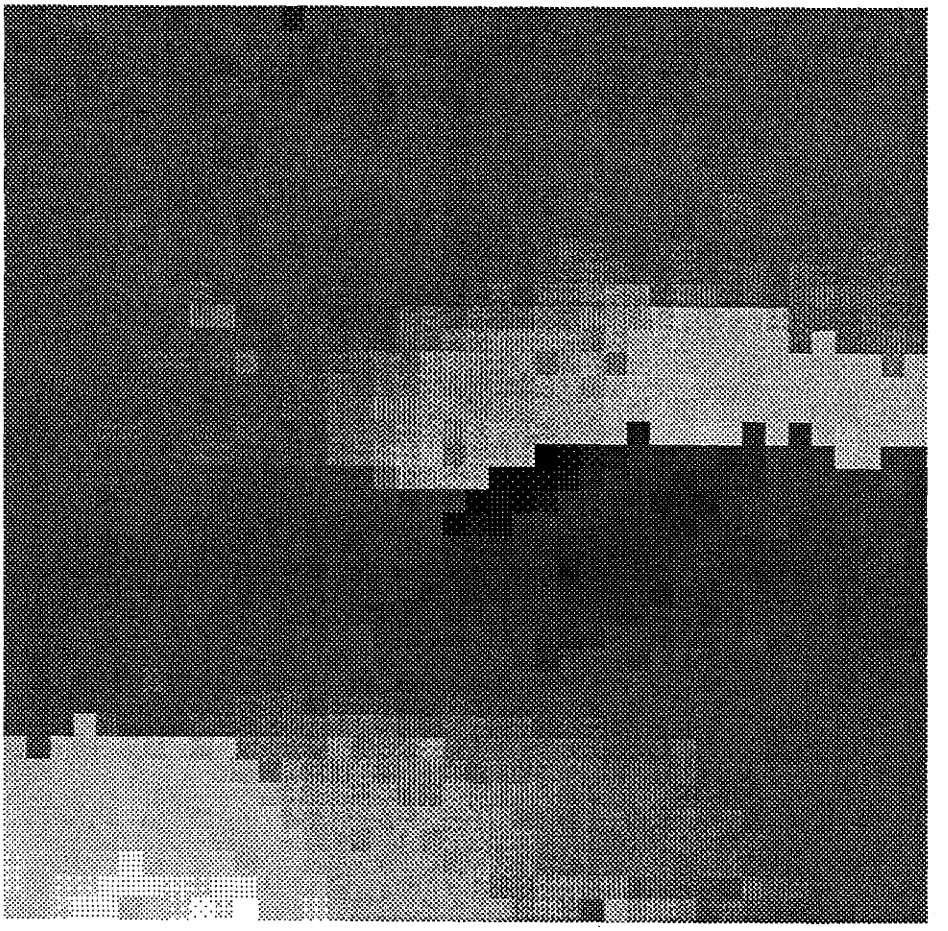

Figure 3.38: Estimate of $\underline{\lambda}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

produces sample functions from a distribution and, consequently, allows variability in the sample.

Figures 3.40 and 3.41 illustrate the estimates of $\underline{d}$ and $\underline{\lambda}$ based on the data in Figure 3.39. The choice of parameter values is $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$. As can be seen, the estimates are good, however, it is apparent that the estimate of $\underline{d}$ is slightly oversmoothed and the estimate of $\underline{\lambda}$ is slightly undersmoothed.

The final example is based on a non-binary real data set. The data is illustrated in Figure 3.42. Notice that the character of the data is similar to that of the synthetic data in Figure 3.39. That is, there is an abrupt jump in dip value across a boundary in the data. The bed frequency is variable throughout the data.

Figures 3.43 and 3.44 illustrate the estimates of $\underline{d}$ and $\underline{\lambda}$ based upon the data in Figure 3.42. The parameter values are, again $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$. As can be seen, these estimates do not capture the character of the data. The boundary between the piecewise-constant regions of dip does not appear in either estimate.

There exist two possibilities for the failure of the estimates in Figures 3.43 and 3.44. The first possibility is that the neighborhood structure (consisting of 7×7 blocks of data) is not large enough to capture the bed features in the data. The beds are thicker than 7 pixels in some regions of the data. However, careful examination of the data and estimates also reveals some thin beds in the data which are not captured by the estimates.

The second possibility has to do with the fact that the data illustrated in Figure 3.42 has beds which are adjacent but are composed of data which are not of opposite sign. This confuses the model presented in this chapter since it is a change in sign value which indicates a bed boundary.

3.6 Conclusions and Discussion

This chapter has presented a model which can be used for signal processing of binary image data consisting of layered structures. The signal processing is accomplished by calculating MAP estimates of two feature vectors $\underline{d}$ and $\underline{\lambda}$ which are integral components of a simple MRF model. Performance of the model depends on four

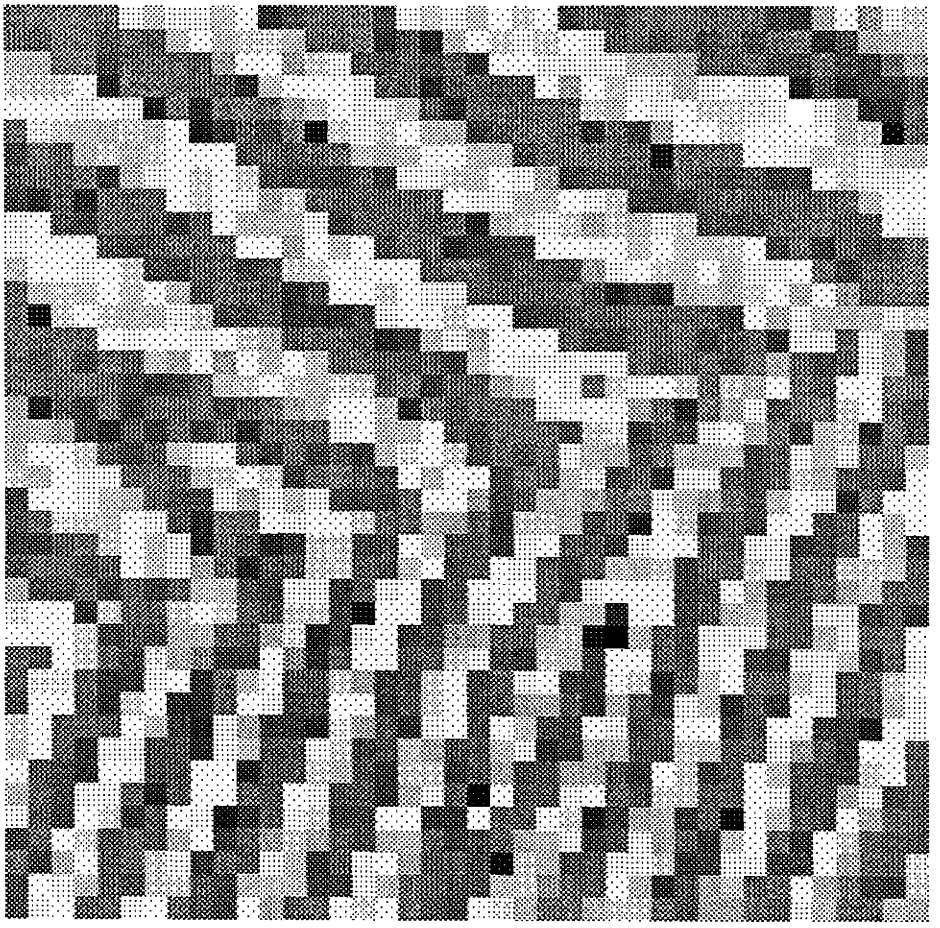

Figure 3.39: Non-Binary Sample Data From Piece-Wise Constant $\underline{d}$ and $\underline{\lambda}$ fields

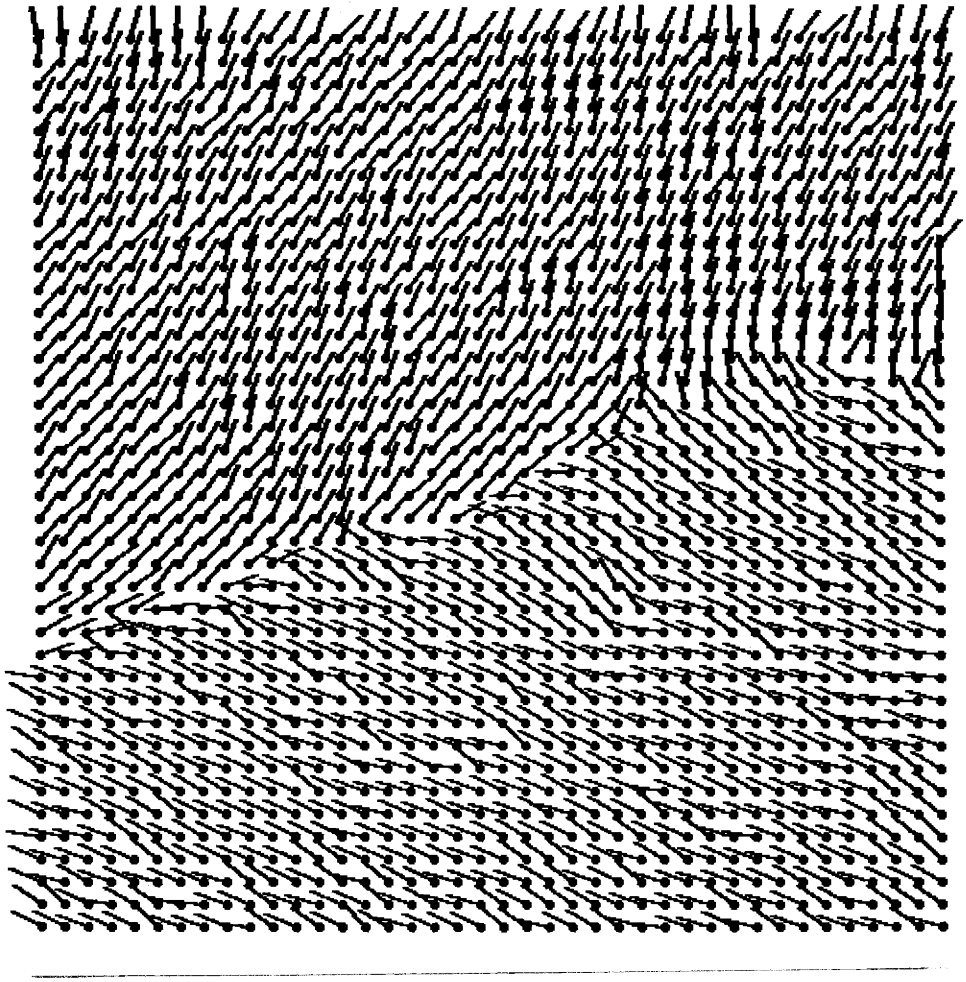

Figure 3.40: Estimate of $\underline{d}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

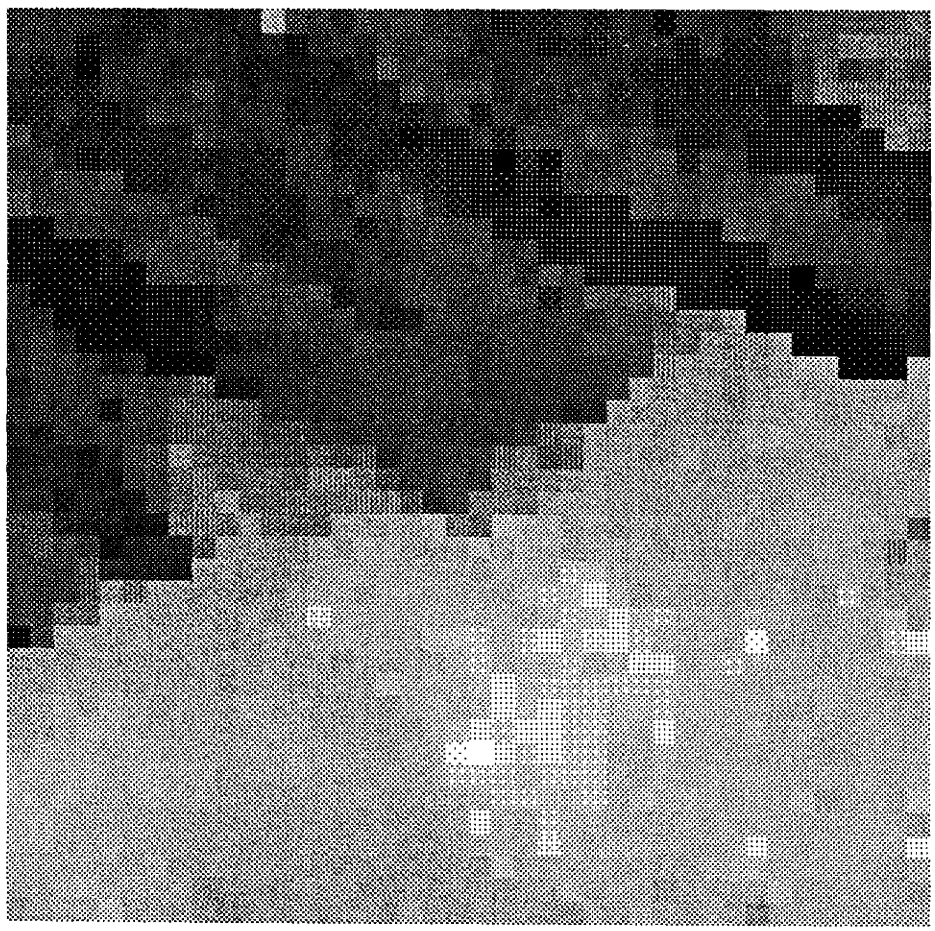

Figure 3.41: Estimate of $\underline{\lambda}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

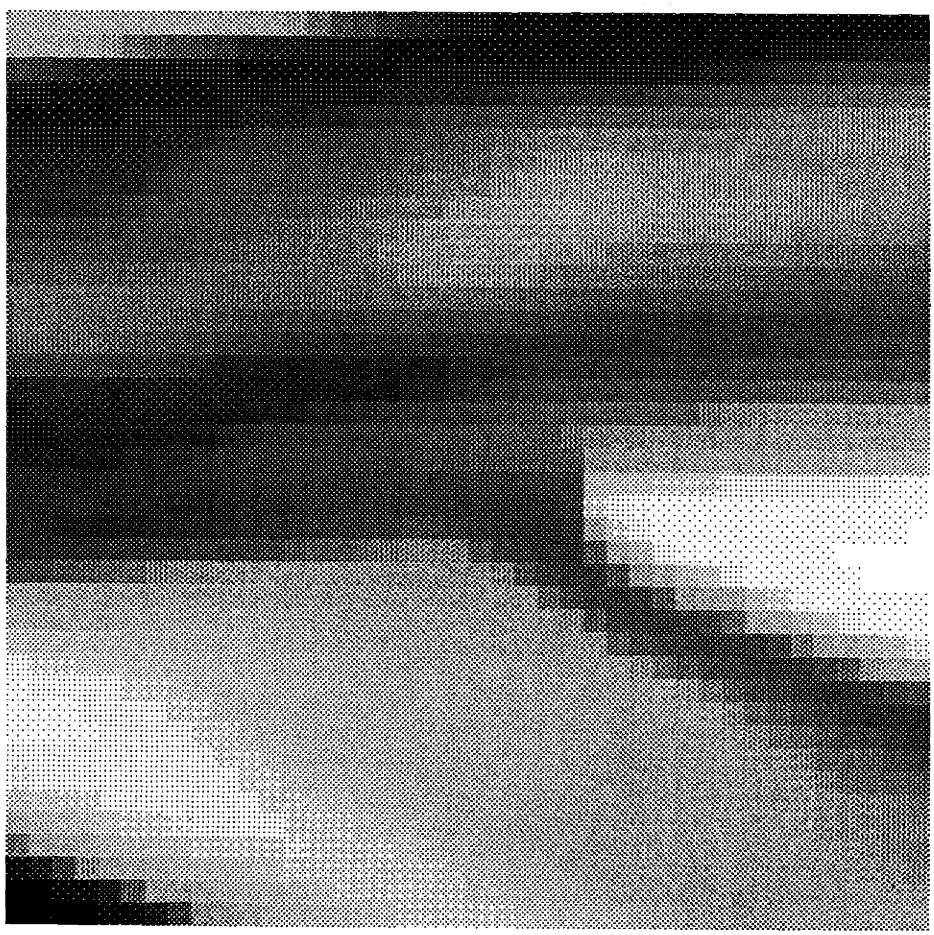

Figure 3.42: Real Non-Binary Data

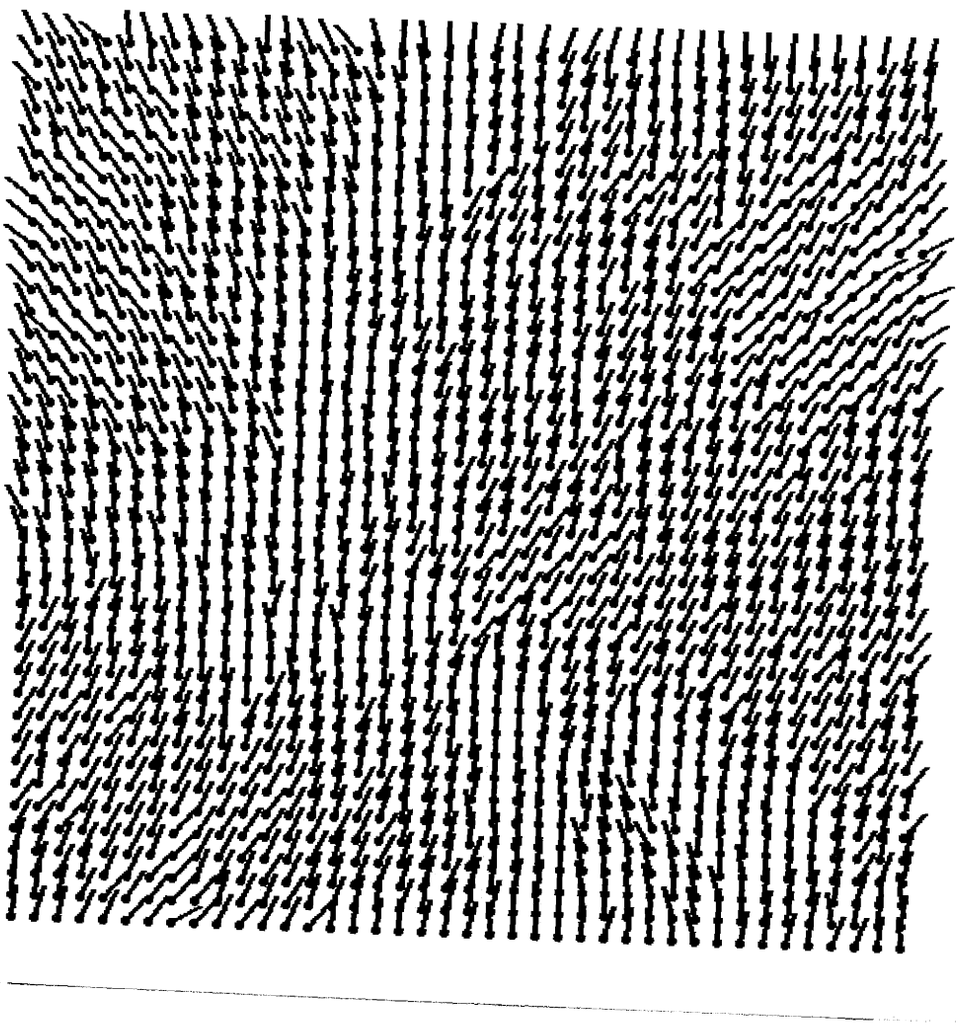

Figure 3.43: Estimate of $\underline{d}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

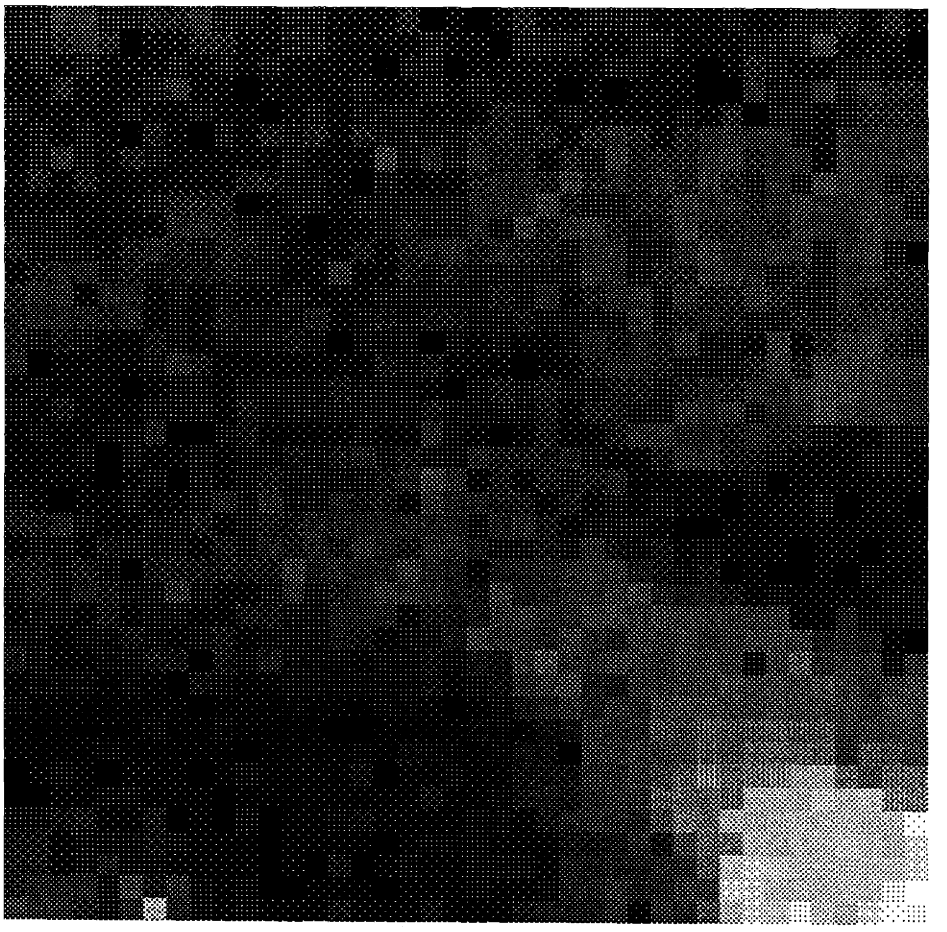

Figure 3.44: Estimate of $\underline{\lambda}$ for $D = \Lambda = 1$ and $\gamma_d = \gamma_\lambda = 2$

parameters, D, Λ, γ_d, and γ_λ. Examples presented in Section 3.5 illustrate the effects of these parameters on the estimates.

As was shown in Section 3.5, the signal processing algorithms discussed in this chapter are effective. However, the algorithms are limited to special types of data sets which consist of layers of alternating sign. The final example of Section 3.5 demonstrated that the estimation algorithms discussed in this chapter are not necessarily effective on general non-binary data.

Clearly, one of the reasons why the models of this chapter are ineffectual on general non-binary data is because of the limitation of the type of energy function used in the Gibb's distribution. The fact that the model relies on alternation in the sign of the data is clearly a drawback for data which is non-binary.

However, there is a deeper problem with the models in this chapter. The models here are partially dependent on distances between elements of the data. Furthermore, the models in this chapter assume data of an anisotropic structure. In the case of these models the anisotropy is due to a long correlation length orthogonal to dip and a short correlation length parallel to dip. Intuitively, data which is highly correlated is, in some sense, close (spatially) and data which is uncorrelated is likely to be distant.

Consequently, the associated intuition about correlation implies that distances which are traversed parallel to the dip vector are different than distances traversed orthogonal to the dip vector. The above discussion suggests that the neighborhood structures of the MRF models should directly incorporate or account for differences in correlation distances. That is, neighborhood structures should be functions of dip.

The next chapter discusses models which are designed to handle general layered non-binary data. The most important feature of the models are their incorporation of neighborhoods which depend locally on dip. The change in modeling of neighborhood structures, making them functions of dip leads to results circumventing the undesirable features of the models proposed in this chapter.

Chapter 4

Correlation Model for Layered Data

This chapter proposes several models which are designed to overcome the shortcomings of the binary data model of Chapter 3. These models are themselves MRF's and are intended for use on continuous-valued or binary-valued data containing layered features (as in Chapter 3).

In this chapter the MRF models do not depend on beds that alternate in sign. Rather, the models are designed to find directions of minimum (or maximum) correlation. This change in modeling philosophy precludes the need for estimating the bed frequency field, $\underline{\lambda}$. Since estimation of dip is associated with correlation and not the sign of the bed data the bed thickness (i.e. $1/\lambda$) is not necessary information. Consequently, only dip estimation is considered in this chapter.

The chapter is composed of the following section. Section 4.1 describes two MRF models for jointly distributed dip and data fields. The rationale for the models in Section 4.1 is discussed later in the chapter in Sections 4.3 and 4.4. Section 4.2 discusses MAP estimation of the dip field conditioned on observations of the data and using the model described in Section 4.1. This section also includes an example of dip estimation using the SA algorithm. Section 4.3 introduces the major ideas underlying the model proposed in Section 4.1 by introducing models that capture the correlation-oriented approach to dip estimation. A sequence of simplifications is then presented in Section 4.4. These simplifications form the conceptual bridge from the models introduced in Section 4.3 to those discussed in Section 4.1. Each successive model/algorithm captures the spirit of the model in Section 4.3, however,

with increasingly reduced computational complexity. Examples are given in Section 4.4 to allow comparison of the various algorithms. Section 4.5 contains further examples of our simplified algorithm, and conclusions and discussion are given in Section 4.6.

4.1 MRF Modeling of Data and Spatially Varying Dip

This section presents two MRF models which have continuously-valued data. Furthermore, the models have neighborhood structures which depend on the local value of dip. The objective of this section is to explain the structure of the models. The rationale for the models is left to Section 4.3 and 4.4.

As in Chapter 3, the models of this section are defined as Gibb's distributions. The models are second order MRF's, thus,

$$p(\underline{s},\underline{d}) = \frac{1}{Z}\exp\{-\sum_{i=1}^{M}\sum_{j=1}^{M} G_{ij}s_i s_j - \sum_{i=1}^{M}\sum_{j=1}^{M} D_{ij}|d_i \cdot d_j|\} \tag{4.1}$$

where Z is a normalization constant and G_{ij} and D_{ij}, when specified, complete the model description. The D_{ij} serve an identical function to the one they performed in Chapter 3. The neighborhood structure of the D_{ij} is also identical to that in Chapter 3. Not much more is said about the D_{ij} here except for the fact that we have taken them all to be equal to a constant value which serve to scale the smoothing of the dip field. The data is now considered to be continuously-valued. Furthermore, the data is arranged on an $N \times N$ square Cartesian grid where the location vector, p_i, of element i has integer components $p_i = (k,l)$ and $l = [\text{int}(i/N) - 1]$ where $k = [i - (l-1)N]$.

So far, the model in (4.1) is similar to the models proposed in Chapter 3 with the exception that the s_i are now continuously valued. There is a major conceptual difference, however, in the models that are now presented. The differences are in the way the G_{ij} are specified and how the neighborhoods are internally structured. To describe the function of the G_{ij} and the internal structure of the neighborhoods

associated with (4.1) we first recapitulate these for the models in Chapter 3.

The models in Chapter 3 describe the interaction between the data M-vector $\underline{s}$ and feature M-vectors $\underline{d}$ and $\underline{\lambda}$.[1] For each element of the data vector, s_i, there is an associated element, d_i, from the dip vector. Furthermore, there is also an associated neighborhood, N_i, which consists of a subset of the indices $\{1, 2, ..., M\} - i$. The model in Chapter 3 takes the neighborhood N_i to be the $n_1 \times n_2$ rectangular array of elements which have the element i as its center.

At a superficial level the structure of the models developed in this chapter are the same as that used in Chapter 3: the neighborhoods again consist of $n_1 \times n_2$ rectangles. However, there is an important difference. As depicted in Figure 4.1 we can think of the neighborhood structure used in Chapter 3 as having a natural internal ordering (the nearest neighbors are the four points closest to the center pixel) that doesn't depend on dip (although the G_{ij} values do depend on dip). In this chapter we will often find it useful to think of dip determining the internal ordering of the neighborhood. As illustrated in Figure 4.2 the "nearness" of neighbors is determined by how close they are after being projected onto the dip vector. This reflects the idea that points in the same bed, i.e. lying along a line perpendicular to the dip vector, are close to one another. This is reflected in our choice of G_{ij} in this chapter: these functions depend essentially only upon the nearness of pixels i and j as just defined.

To be a bit more precise, the neighborhood structure illustrated in Figure 4.2 affects the model through a re-ordering of the data elements. That is, the implementation of the effect of d_i on N_i is through the rearrangement of the data elements. Normally the data in the $n_1 \times n_2$ neighborhood N_i is ordered lexicographically into a data sub-vector. Referring to Figure 4.1, the $n_1 \times n_2$-vector $\underline{s}_{N_i}$ of the data elements in neighborhood N_i take the following ordering. The first element is the element of the lower left corner of Figure 4.1. The ensuing elements are taken from Figure 4.1 moving along rows from left to right. When a row is finished the next element comes from the next higher row starting, again, from the left side.

The ordering of data into the vector $\underline{s}_{N_i}$ when using the neighborhood structure

[1] The $\underline{\lambda}$ vector is incidental to the remaining discussion and is ignored.

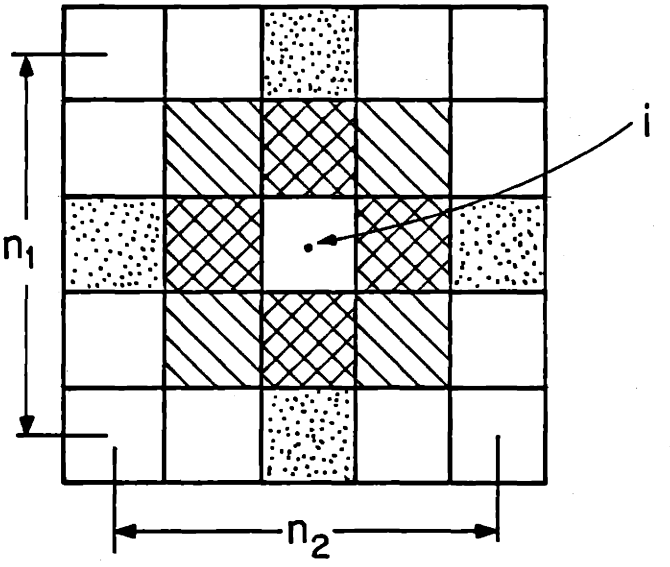

Figure 4.1: Neighborhoods Independent of d_i

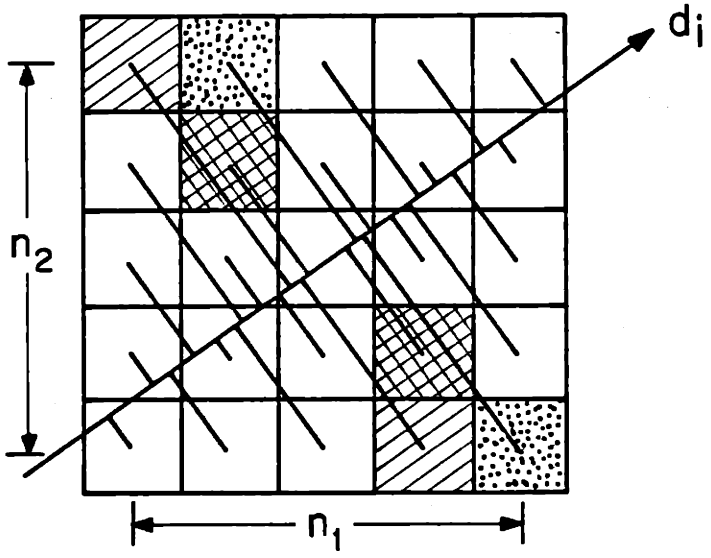

Figure 4.2: Neighborhoods Dependent on d_i

illustrated in Figure 4.2 is as follows. The projection of the element i onto the dip direction is taken to be the origin of the line. The element which has the most negative projection onto the line with respect to the origin is the first element in the vector $\underline{s}_{N_i}$. The ensuing elements are ordered in the vector from the most negative projection through the most positive projection. When ambiguity exists due to elements which project to the same point on the line, the element with the shortest perpendicular to the line is entered into the vector closest to the element i. If ambiguity still exists due to equal perpendicular distance then the ordering is taken randomly.

The above dip-directed ordering has a modeling interpretation. The idea is that data which are separated by a directed distance which is orthogonal to dip is highly correlated. The larger the component of separation parallel to dip becomes, the lower the correlation. Furthermore, data along a direction perpendicular to dip decreases in correlation, however, at a much longer scale than in the direction parallel to dip.

The data ordering as a function of dip can be represented by a permutation matrix, E. The permutation matrix consists of a single one in each of its rows and columns and zeroes everywhere else. Furthermore, E is an $(n_1 n_2) \times (n_1 n_2)$ matrix. Consequently, the ordered data vector $\underline{s}_{N_i}(d_i)$ is

$$\underline{s}_{N_i}(d_i) = E(d_i)\underline{s}_{N_i} \tag{4.2}$$

where $\underline{s}_{N_i}$ is the lexicographically ordered data vector.

Now that the neighborhood structure has been described, the G_{ij} can be specified. The explicit values of the G_{ij} are not given. Rather, an expression which is equivalent to the first double sum in (4.1) is described. Labeling this double sum as $V(\underline{s}, \underline{d})$,

$$\begin{aligned} V(\underline{s}, \underline{d}) &= \sum_{i=1}^{M}\sum_{j=1}^{M} G_{ij} s_i s_j \\ &= \sum_{i=1}^{M} \underline{s}_{N_i}^T E^T(d_i) K E(d_i) \underline{s}_{N_i} \\ &= \sum_{i=1}^{M} \underline{s}_{N_i}^T(d_i) K \underline{s}_{N_i}(d_i) \end{aligned} \tag{4.3}$$

where K is an $(n_1 n_2) \times (n_1 n_2)$ matrix to be specified. The interpretation of (4.3) is that $V(\underline{s}, \underline{d})$ is calculated by summing the contributions from each neighborhood. Each neighborhood orders the data according to the neighborhood dip and then calculates a quadratic product with constant matrix K.

The entries in matrix K take one of two forms. These two forms are differentiated by subscripts so that we use either K_ρ or $K_{\rho\gamma}$ and the meaning of the subscripts is explained in Section 4.3. The matrix K is defined by

$$K = \sum_{i=1}^{n_1 n_2 - \tau + 1} K_i \tag{4.4}$$

where K_i is an $(n_1 n_2 \times n_1 n_2)$ matrix and τ is an integer less than $n_1 n_2$. The matrix K_i is zero everywhere except for an embedded $\tau \times \tau$ matrix. This $\tau \times \tau$ matrix is denoted A_ρ or $A_{\rho\gamma}$ corresponding to the K_ρ or $K_{\rho\gamma}$ matrices. The location of A within K_i is illustrated in Figure 4.3. The $(1,1)$ element of A is at the (i,i) element of K_i, the $(1,2)$ element of A is at the $(i, i+1)$ element of K_i, etc. The matrix A takes one of the following forms

$$A_\rho = \begin{bmatrix} \frac{\tau-1}{\tau^2} & -\frac{1}{\tau^2} & \cdots & -\frac{1}{\tau^2} \\ -\frac{1}{\tau^2} & \frac{\tau-1}{\tau^2} & \cdots & -\frac{1}{\tau^2} \\ \vdots & \vdots & & \vdots \\ -\frac{1}{\tau^2} & -\frac{1}{\tau^2} & \cdots & \frac{\tau-1}{\tau^2} \end{bmatrix} \tag{4.5}$$

$$A_{\rho\gamma} = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.6}$$

The effect of taking the quadratic product of a τ-vector with the matrix in (4.5)

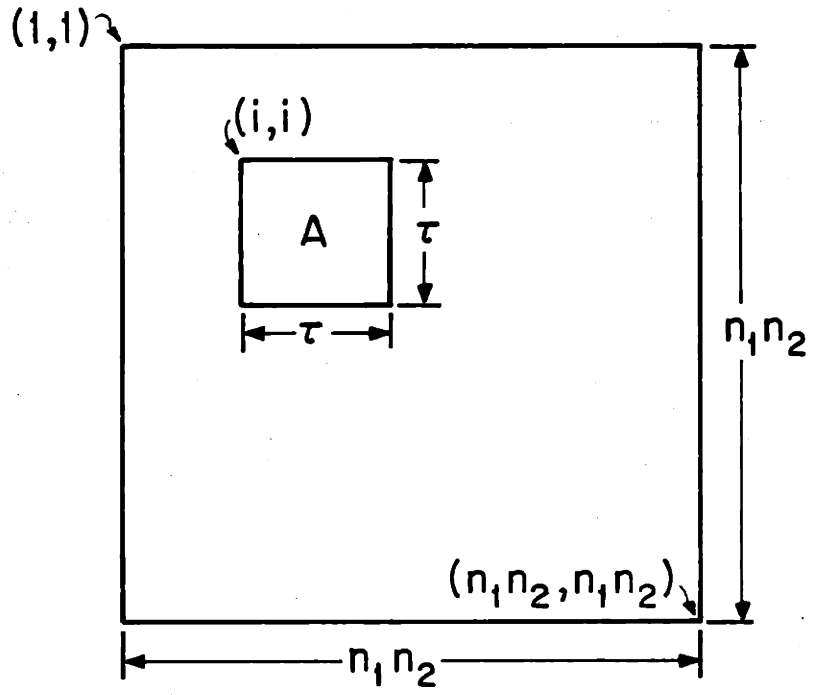

Figure 4.3: Matrix K_i

is to compute the vectors sample covariance. That is, for a τ-vector $\underline{x}$

$$\begin{aligned}\underline{x}^T A_\rho \underline{x} &= \sum_{i=1}^{\tau} \frac{\tau-1}{\tau^2} x_i{}^2 + \sum_{i=1}^{\tau}\sum_{j\neq i}(-\frac{1}{\tau^2}) x_i x_j \\ &= \frac{1}{\tau}\sum_{i=1}^{\tau} x_i{}^2 - \frac{1}{\tau^2}\sum_{i=1}^{\tau}\sum_{j=1}^{\tau} x_i x_j \\ &= \frac{1}{\tau}\sum_{i=1}^{\tau}[x_i - \frac{1}{\tau}\sum_{j=1}^{\tau} x_j]^2 \end{aligned} \tag{4.7}$$

The effect of the quadratic product with the matrix in (4.6) is to compute the sum of squared differences,

$$\begin{aligned}\underline{x}^T A_{\rho\gamma} \underline{x} &= x_1{}^2 + x_\tau{}^2 + \sum_{i=2}^{\tau-1} 2x_i{}^2 - 2\sum_{i=1}^{\tau-1} x_i x_{i+1} \\ &= \sum_{i=1}^{\tau-1}(x_i - x_{i+1})^2 \end{aligned} \tag{4.8}$$

Thus, in each neighborhood, the data is ordered according to the dip. Then a moving window computation is made. The window is of length τ and calculates either the sample variance or the sum of squared differences. The intuition is as follows. The data has been ordered so that elements which are highly correlated are close to each other. Consequently, computing a moving sample covariance or moving sum of squared differences should have a close to optimally small value, and, thus, a high likelihood in (4.1). The parameter τ is a correlation length parameter which regulates the distance within which the model believes data to be highly correlated.

The two models discussed here rely solely on orderings of the data. These orderings are the manner by which the dip interacts with the data. The next section discusses the procedure of MAP estimation of dip from observations of the data using the SA algorithm. Sections 4.3 and 4.4 give a detailed description and analysis of a sequence of models and algorithms which culminate in the one specified here.

4.2 MAP Estimation of Dip Using the SA Algorithm

Two models for continuously valued data were discussed in Section 4.1. These models relate the dip field and data field through an energy function as was the case in Chapter 3 for binary valued data. However, the relationship between dip and data is now conveyed through the neighborhood structure of the model. The dip interacts with the data by orderings of the data vector.

The purpose of this section is to describe the process of obtaining a MAP estimate of the dip vector, $\underline{d}$. As in Chapter 3 we use the SA algorithm to obtain this estimate which is referred to as $\hat{\underline{d}}_{SA}$. The notation $\hat{\underline{d}}_{SA}$ is used since the SA algorithm does not necessarily find the exact MAP estimate but does find an approximate MAP estimate.

As in Chapter 3 there are choices to be made concerning the values of the smoothing parameters, D_i. For the purposes of this chapter the smoothing parameters D_i are constant, that is $D_i = D$. The choice of value for D is found through some experimentation which is described in Section 4.5.

Again, as in Chapter 3, the neighborhoods, N_i, of this chapter consist of the 7×7 array of data elements with element i at the center. However, as described in Section 4.1 the internal structuring of the neighborhoods is different from the structures described in Chapter 3.

The neighborhood structures in this chapter are based upon orderings of the data along the hypothesized dip direction. There can only be a finite number of orderings for a 7×7 array of data along the dip direction. Consequently, only a finite resolution of dip may be had. This is no surprise given the model, and there is a benefit to be had from the explicit relationship between dip and orderings of the data. The benefit is that the orderings can be pre-computed since there is a finite number of them.

The speed of evaluating the likelihood function is greatly enhanced by pre-computation of the orderings of data. The remaining computation is a quadratic product of the ordered data with a simple matrix. The matrix is sparse and allows

for fast computations of the quadratic product. More about these fast computations is discussed in Section 4.4.

We close this section with an illustration of a dip estimate for the last data set of Section 3.5. More examples are discussed in Section 4.5. The data set is reproduced in Figure 4.4. Figure 4.5 illustrates the dip estimate obtained using the SA algorithm and the A_ρ matrix of (4.5). The value of the smoothing parameter was taken to be $D = .1$.

The data set in Figure 4.4 is carried through the sequence of algorithms presented in Section 4.4. This is for comparison purposes. There are two major regions in the data of Figure 4.4. These regions are separated by a line. The regions above and below the line are regions of relatively constant dip with a sharp difference in value across the line. This type of feature is often referred to as an "unconformity" in the geological literature [23].

As can be seen in Figure 4.5 there are three major regions of interest in the set of dip estimates. Two of the regions correspond well to two of the major regions of interest in the data. The third region is a small fantail of dips which are located at the juncture of the bend in the line separating the two data regions. This fantail is caused by the two high contrast bed boundaries just above and below the right bend portion of the line separating the data. The dip smoothly varies from the lower dip value to the upper dip value in this fantail.

The left bend portion of the line separating the Figure 4.4 data is higher than the corresponding dip line in Figure 4.5. This is due to the lack of high contrast bed boundaries below the data line. Consequently, the dip estimate lock onto the high contrast boundary at the data line. That is, in estimating dip at a pixel located slightly below the left-hand portion of the line in Figure 4.4, it is the high-contrast boundary at the upper edge of the pixel's neighborhood that determines the dip estimate. Clearly this would seem to indicate a built-in bias in determining boundaries in the data. However, in the next chapter we directly address the problem of boundary estimation following dip determination in a way that overcomes this apparent bias.

Figure 4.6 is an estimate of dip using the $A_{\rho\gamma}$ matrix of (4.6). The results of

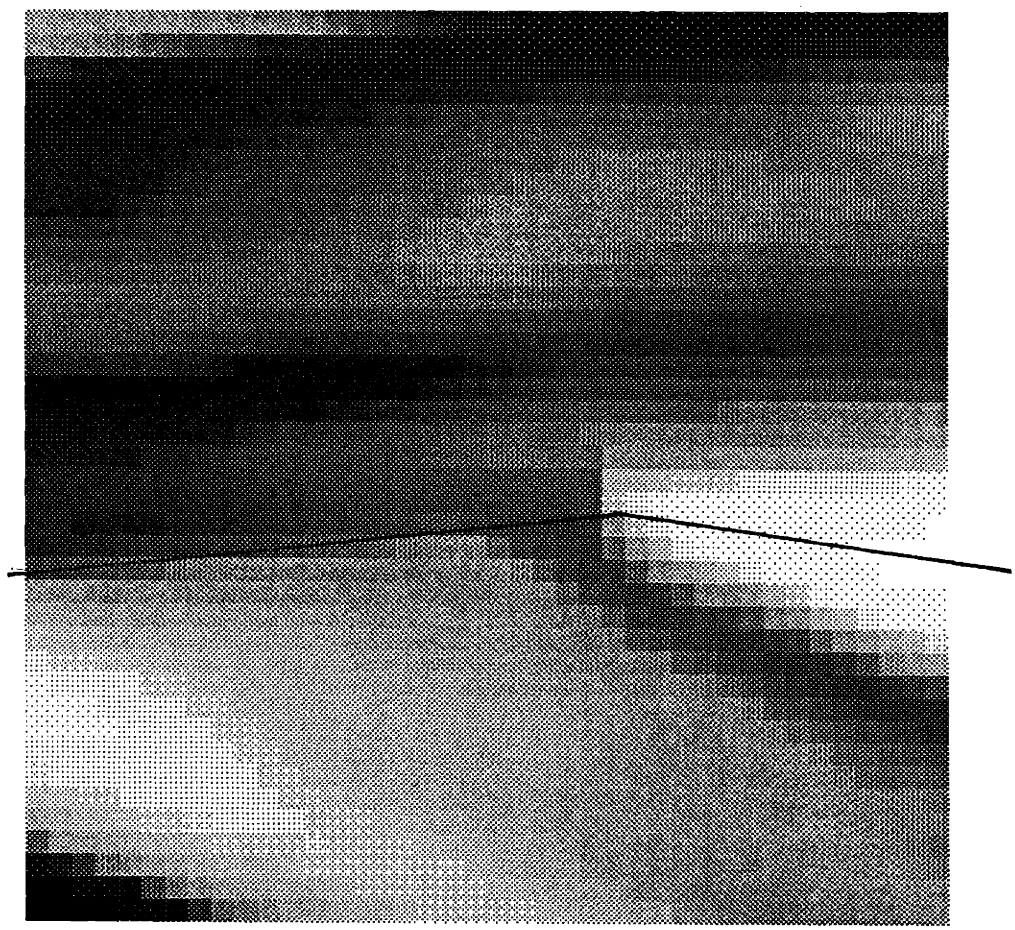

Figure 4.4: Non-Binary Data

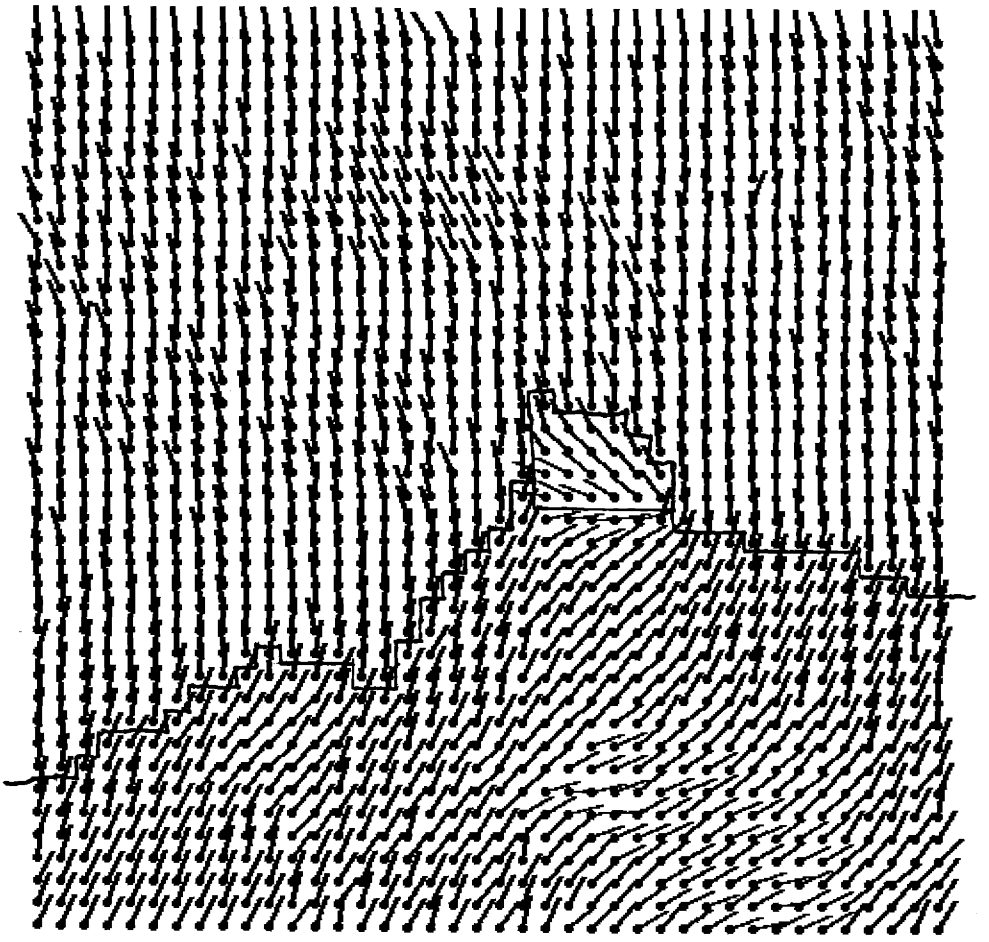

Figure 4.5: Dip Estimate From SA Algorithm with A_ρ Matrix and $D = .1$

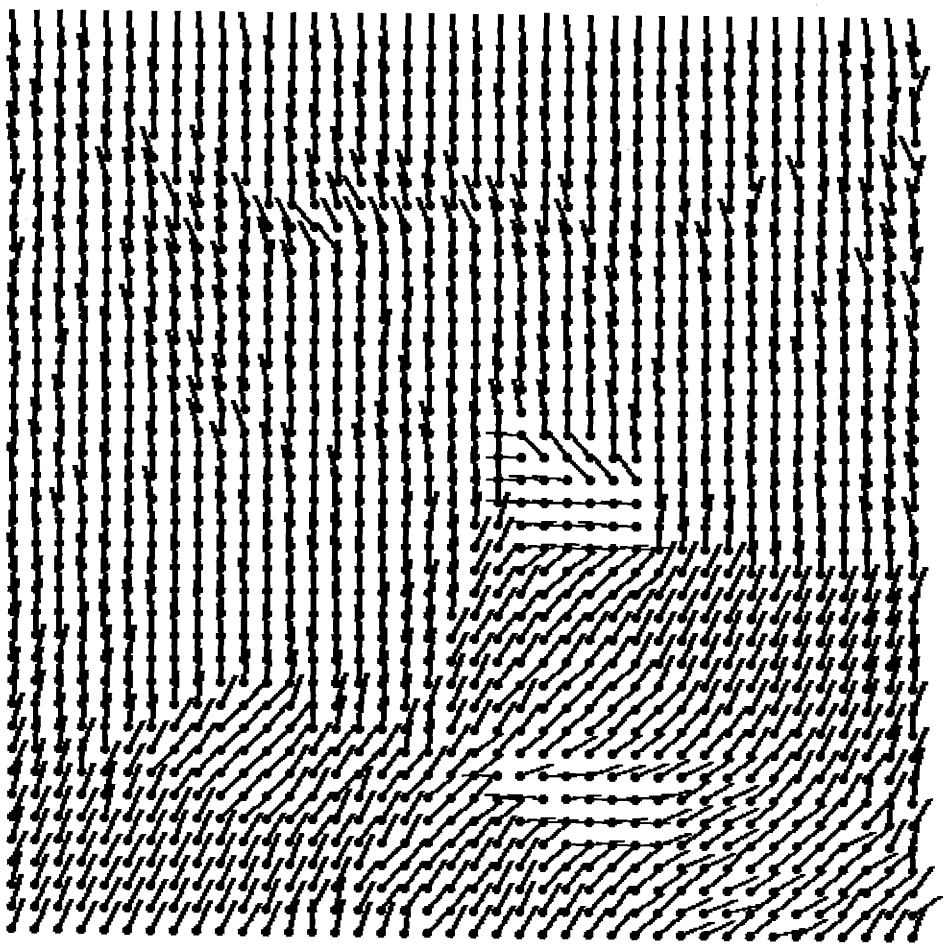

Figure 4.6: Dip Estimate From SA Algorithm with $A_{\rho\gamma}$ Matrix and $D = .1$

this figure are similar to those in Figure 4.5. The two models are contrasted more completely in Section 4.5.

4.3 Constant Dip Field Models

Section 4.1 presented a joint distribution for the data, $\underline{s}$, and the dip field, $\underline{d}$, where $\underline{d}$ was assumed to be a spatially-varying vector of dip values. This joint distribution is conceptually simple, however, it is extremely difficult to analyze. The difficulty stems from the space-varying, interacting dip values. This section and ensuing sections analytically explore the model in Section 4.1. The analysis proceeds by examining a single neighborhood of the model in Section 4.1 and, thus, only a single value of dip.

This section proposes several models for data which is available on both spatially continuous and discrete sets. The models proposed do not necessarily seem related to the MRF models proposed in Section 4.1. However, Section 4.4 reveals the connection between the models discussed here and the neighborhood structures and G_{ij} functions of Section 4.1. The models are parametrically dependent on a single value of dip which is denoted by d (no underscore). This contrasts the vector of dip values, $\underline{d}$, used in Section 4.1.

4.3.1 Two Models for Spatially-Continuous Data

Two models for spatially continuous data are discussed in this section. For both models assume that the data, $s(\underline{x})$, is available on a disk, D, centered at the origin and of radius R (see Figure 4.7). The data is assumed jointly Gaussian and zero mean. Specification of the covariance function completes the model. The covariance function is different for the two models.

For the first model the covariance function is denoted by $\Lambda_\rho(\underline{x},\underline{y})$ and

$$\Lambda_\rho(\underline{x},\underline{y}) = \rho^{|d\cdot(\underline{x}-\underline{y})|} + {\sigma_n}^2\delta(\underline{x}-\underline{y}) \tag{4.9}$$

where $0 \leq \rho < 1$ and ${\sigma_n}^2$ is the noise variance. The covariance function in (4.9) is called the *continuous ρ-model* and it can be understood as follows. In the direction

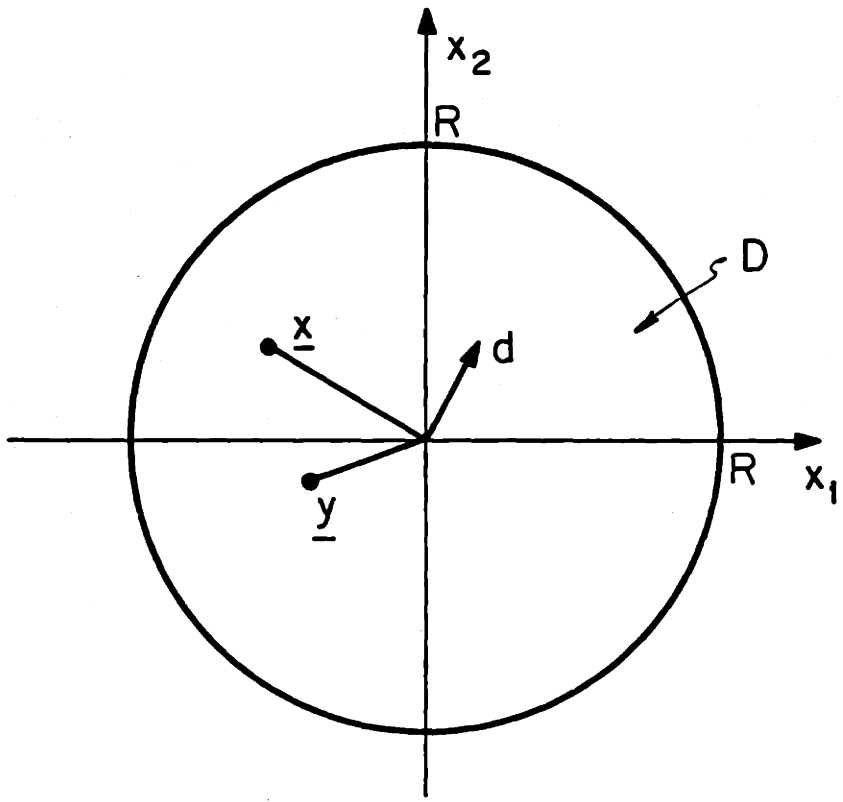

Figure 4.7: Support of GRF data

of the dip vector, d, the data looks like a first order Gauss-Markov process observed in white noise. In the direction orthogonal to the dip vector (denoted by the unit vector $d^{\perp}$) the data is constant and is observed in white noise. Figure 4.8 illustrates two cross-sections of Λ.

In the second model the covariance function is denoted by $\Lambda_{\rho\gamma}(\underline{x},\underline{y})$ and

$$\Lambda_{\rho\gamma}(\underline{x},\underline{y}) = \rho^{|d\cdot(\underline{x}-\underline{y})|}\gamma^{|d^{\perp}\cdot(\underline{x}-\underline{y})|} + {\sigma_n}^2\delta(\underline{x}-\underline{y}) \tag{4.10}$$

where $0 \leq \rho < \gamma < 1$, $d^{\perp}$ is the unit vector orthogonal to the vector d and ${\sigma_n}^2$ is again white noise. The covariance in (4.10) is termed the *continuous $\rho\gamma$-model* and is similar to that in (4.9) with the exception that the data is a first order Gauss-Markov random field observed in white noise in both the d direction and the $d^{\perp}$ direction. If γ were allowed to be unity, then (4.10) would reduce to (4.9).

4.3.2 Two Models For Spatially-Discrete Data

The discrete models are analogs of the continuous models. All of the practical signal processing algorithms of this chapter are for the discrete models. However, these practical models are based on optimal signal processing algorithms for the continuous data models.

The discrete models assume the data to be a jointly Gaussian random vector. The data is assumed to be arranged on an $n_1 \times n_2$ rectangular grid with the lower left hand corner element located at the grid point $(1,1)$ (see Figure 4.9). The vector of data is referred to by the notation $\underline{s}$ which should not be confused with the notation for the M-element vector of Chapter 3 and Section 4.1.

The $n_1 n_2$ components of $\underline{s}$ are referred to in several ways in the ensuing discussion. Primarily the data is denoted by the $n_1 n_2$-vector $\underline{s}$. The ordering of the components in $\underline{s}$ is from the lower left corner to the upper right corner of Figure 4.9 traveling along rows from left to right. Consequently, the first n_2 elements of $\underline{s}$ come from the lowest row of Figure 4.9, the left-most element being the first component of $\underline{s}$. The second lowest row of Figure 4.9 constitutes the next n_2 elements, etc.

Often the k^{th} row of $\underline{s}$ is denoted by the n_2-vector $\underline{s}_k$. This should be contrasted with the notation for the k^{th} element of $\underline{s}$, denoted s_k (no underscore).

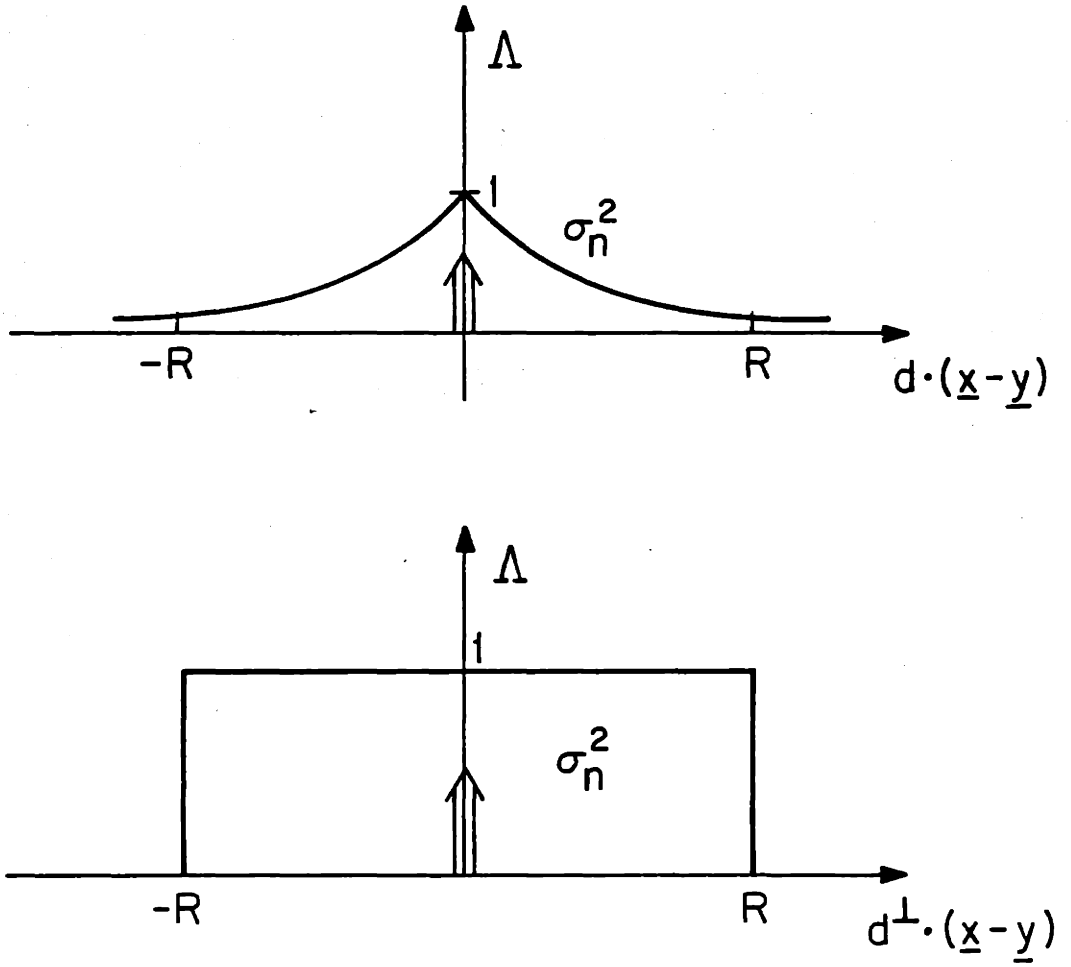

Figure 4.8: Cross-Sections of Λ

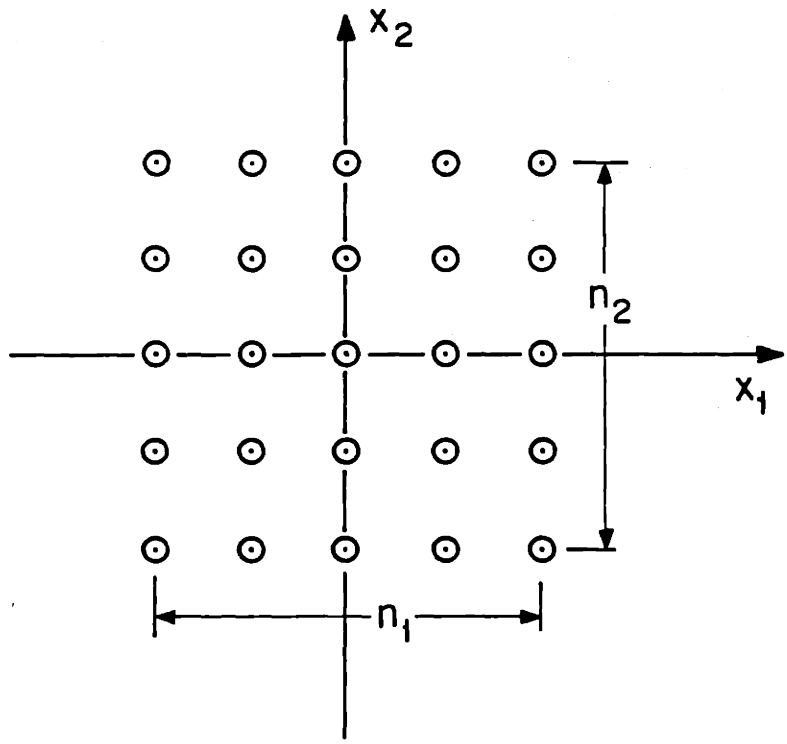

Figure 4.9: Sampling Geometry of Discrete Data Models

As for the continuous models the discrete models assume the data to be Gaussian and of zero mean. The covariance matrix of the first model is analogous to the spatially continuous model (4.9) and has elements

$$[\Lambda_\rho(\underline{x},\underline{y})]_{ij} = \rho^{|d\cdot(\underline{x}_i-\underline{x}_j)|} + {\sigma_n}^2\delta_{ij} \tag{4.11}$$

where $\underline{x}_i$ is the two-vector location of element s_i.

The covariance matrix of the second model is analogous to the spatially continuous model (4.10) and has elements

$$[\Lambda_{\rho\gamma}(\underline{x},\underline{y})]_{ij} = \rho^{|d\cdot(\underline{x}_i-\underline{x}_j)|}\gamma^{|d^\perp\cdot(\underline{x}_i-\underline{x}_j)|} + {\sigma_n}^2\delta_{ij} \tag{4.12}$$

This completes the initial modeling section of this chapter. The next section, Section 4.4, discusses a number of signal processing algorithms based on the models presented here. The sequence of algorithms culminates in a final algorithm which is the basis for the model in Section 4.4.

4.4 Signal Processing: ML Estimation of d

Section 4.1 presented models for spatially varying dip. However, the computations within each neighborhood of the models of Section 4.1 are based on a constant dip model. One objective of this section is to show that the constant-dip models in Section 4.3 are the basis for the constant neighborhood models in Section 4.1. A second is to demonstrate the utility of the types of signal processing algorithms we have proposed.

In this section we propose a sequence of increasingly simplified algorithms based on the models of Section 4.3. Each algorithm retains the character of the models while also becoming increasingly computationally simplified. The final algorithm which is discussed is the basis for the model used in constructing the models of Section 4.1.

Several fast algorithms for estimating the direction of minimum correlation are proposed in this section. There is a crucial modeling feature which is the crux of the fast algorithms. This feature is that the covariance models are ARC functions

(see Chapter 2). ARC functions are important modeling tools in image processing problems and this comes out in the ensuing sections.

4.4.1 ML Estimation of d

In this section the maximum likelihood (ML) estimate for d is described in both the continuous and discrete data cases. In the continuous data case the likelihood function is determined with the help of a complete orthonormal (CON) basis. In the discrete data case the likelihood function can be determined directly. In both cases the ML estimate is obtained by maximizing the joint distribution of the data

$$\hat{d}_{ML} = \arg\max_{d} p[s(\underline{x})] \tag{4.13}$$

for continuous data, and

$$\hat{d}_{ML} = \arg\max_{d} p(\underline{s}) \tag{4.14}$$

for discrete data.

For continuous data the joint distribution cannot be obtained directly. Rather, a limit is taken of a k coefficient expansion $s(\underline{x})$ using a CON basis. Following Van Trees [26] we assume that the covariance function of the data is positive definite so that the set of functions $\{\phi_i(\underline{x})\}$ is a CON basis where the elements of this set satisfy

$$\int_{\underline{y}\in D} K(\underline{x},\underline{y})\phi_i(\underline{y})d\underline{y} = \lambda_i\phi_i(\underline{x}) \tag{4.15}$$

where $\{\lambda_i\}$ are the eigenvalues and $\{\phi_i\}$ are the eigenfunctions of K. That K is a function of d is denoted by $K_d(\underline{x},\underline{y})$.

To find the likelihood function, the data is expanded in the CON basis obtained from $K(\underline{x},\underline{y})$

$$s(\underline{x}) = \sum_{k=1}^{\infty} s_i\phi_i(\underline{x}) \tag{4.16}$$

where

$$s_i = \int_{\underline{x}\in D} s(\underline{x})\phi_i(\underline{x})d\underline{x} \tag{4.17}$$

Since $s(\underline{x})$ is a GRF the s_i are independent JGRV's. The likelihood function of $s(\underline{x})$ can be obtained as the limit of the likelihood function of the s_i. Denoting the

likelihood function of the first K s_i as p_K we have

$$p_K = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi\lambda_i}} \exp\{-\frac{1}{2}\frac{{s_i}^2}{\lambda_i}\} \tag{4.18}$$

For the continuous data models of Section 4.3.1 the covariance models are ARC functions (see Chapter 2). Thus, for these models, the λ_i are independent of the value of d and, consequently, the $\sqrt{(2\pi\lambda_i)}$ term can be dropped from (4.18) yielding the equivalent likelihood function,

$$\begin{aligned} \tilde{p}_K &= \prod_{i=1}^{K} \exp\{-\frac{1}{2}\frac{{s_i}^2}{\lambda_i}\} \\ &= \exp\{-\frac{1}{2}\sum_{i=1}^{K}\frac{{s_i}^2}{\lambda_i}\} \\ &= \exp\{-\frac{1}{2}\int_{\underline{y}\in D}\int_{\underline{x}\in D} s(\underline{x})s(\underline{y})\sum_{i=1}^{K}\frac{\phi_i(\underline{x})\phi_i(\underline{y})}{\lambda_i}d\underline{x}d\underline{y}\} \end{aligned} \tag{4.19}$$

In the limit (4.19) converges to

$$\begin{aligned} p[s(\underline{x})] &= \lim_{k\to\infty}\tilde{p}_K \\ &= \exp\{-\frac{1}{2}\int_{\underline{x}\in D}\int_{\underline{y}\in D} s(\underline{x})Q(\underline{x},\underline{y})s(\underline{y})d\underline{x}d\underline{y}\} \end{aligned} \tag{4.20}$$

where $Q(\underline{x},\underline{y})$ is the inverse covariance function.

The likelihood function for the discrete data model is simpler. The data vector $\underline{s}$ is a JGRV so that the likelihood function can be written

$$p(\underline{s}) = \frac{1}{\sqrt{2\pi^M|\Lambda|}}\exp\{-\frac{1}{2}\underline{s}^T\Lambda^{-1}\underline{s}\} \tag{4.21}$$

where Λ is the covariance matrix of $\underline{s}$.

In general $Q(\underline{x},\underline{y})$ and Λ^{-1} cannot be analytically computed. Both are non-linear functions of d. The approach to calculating $\hat{d}_{ML}$ requires some sort of state space search in which (4.20) or (4.21) is calculated for each candidate value of d. The ML estimate is taken to be the value which maximizes (4.20) or (4.21).

4.4.2 Rotating the Data

The remainder of Section 4.4 addresses signal processing algorithms designed to obtain $\hat{d}_{ML}$ and approximations to $\hat{d}_{ML}$. Implementation of such algorithms is generally easier when considering discrete data because of the flexibility and availability of digital computers. Hence the emphasis is on algorithms for discrete data. However, when it suits the purpose of furthering development of, or conceptualizing new approaches to algorithms the continuous data model is utilized.

This section demonstrates the interchangeability of rotating d or rotating the data for the purpose of evaluating (4.20) of Section 4.4.1. This property holds true for the continuous data model and is only approximately true for the discrete data model. However, the assumption that this property holds exactly leads to tremendous gains in computational efficiency for algorithms which estimate d in the spatially discrete data case.

Equation (4.20) of Section 4.3.1 evaluates $p[s(\underline{x})]$ repeatedly, where

$$p[s(\underline{x})] = \exp\{-\frac{1}{2}\int\int s(\underline{x})Q(\underline{x},\underline{y})s(\underline{y})d\underline{x}d\underline{y}\} \tag{4.22}$$

Here, $Q(\underline{x},\underline{y}) = Q[d\cdot(\underline{x}-\underline{y})]$ and we are only interested in the properties of the integral

$$I = \int_{\underline{y}\in D}\int_{\underline{x}\in D} s(\underline{x})Q[d\cdot(\underline{x}-\underline{y})]s(\underline{y})d\underline{x}d\underline{y} \tag{4.23}$$

Our objective is to show that changing the value of d by a two-dimensional rotation, V, is equivalent to a rotation of the data by V^T. An alternative way of expressing this relation is that rotation of d by V while simultaneously rotating the data by V does not affect the value of I in (4.23).

Theorem 2 *Rotation of the support for the data, $\underline{s}$, by V has the identical effect on (4.23) as does rotation of the dip, d, by V^T. That is*

$$\int_{\underline{y}\in D}\int_{\underline{x}\in D} s(V\underline{x})Q[d\cdot(\underline{x}-\underline{y})]s(V\underline{y})d\underline{x}d\underline{y}$$

$$= \int_{\underline{y}\in D}\int_{\underline{x}\in D} s(\underline{x})Q[(V^Td)\cdot(\underline{x}-\underline{y})]s(\underline{y})d\underline{x}d\underline{y}$$

Proof 2 *The proof proceeds by showing that rotating the data by V while simultaneously rotating the dip vector by V^T is equivalent to having done neither rotation. Let $\tilde{\underline{x}} = V\underline{x}$ and $\tilde{\underline{y}} = V\underline{y}$ be a change of variables. Since $det(V) = det(V^T) = 1$ we have that $d\tilde{\underline{x}} = d\underline{x}$ and $d\tilde{\underline{y}} = d\underline{y}$. Thus,*

$$\int_{\underline{y}\in D}\int_{\underline{x}\in D} s(V\underline{x})Q[(V^T d)\cdot(\underline{x}-\underline{y})]s(V\underline{y})d\underline{x}d\underline{y}$$

$$= \int_{\underline{y}\in D}\int_{\underline{x}\in D} s(V\underline{x})Q[d\cdot(V\underline{x}-V\underline{y})]s(V\underline{y})d\underline{x}d\underline{y}$$
$$= \int_{\tilde{\underline{y}}\in D}\int_{\tilde{\underline{x}}\in D} s(\tilde{\underline{x}})Q[d\cdot(\tilde{\underline{x}}-\tilde{\underline{y}})]s(\tilde{\underline{y}})d\tilde{\underline{x}}d\tilde{\underline{y}}$$

where the first equality comes from the fact that rotations preserve angles between vectors. Q.E.D.

It should be noted that the validity of the above theorem is a direct consequence of the fact that Q is an ARC function. The value of the theorem comes when repeated evaluations of (4.23) are made with different values of d. This is required for calculation of $\hat{d}_{ML}$. For each value of d, calculating (4.23) requires evaluating K and then its inverse, Q. Obtaining Q is a computationally expensive operation.

An alternative procedure is to calculate Q for a single value of d, say d_0. For all ensuing values of $d = Vd_0$ we continue to use $Q(\underline{x},\underline{y}|d_0)$ and we rotate the data support by V^T to obtain the equivalent effect. Rotation of the data is much more computationally efficient than recalculation of Q for each value of d.

The above discussion suggests a simple, computationally efficient algorithm for finding $\hat{d}_{ML}$ in the discrete data case. The next sub-section discusses this algorithm and the computational savings achieved by using it.

4.4.3 The Rotate and Interpolate Algorithm

As is discussed in Section 4.4.2, multiple calculations of the quadratic integral (4.23) can be efficiently computed by rotation of the data. This property is valuable when searching for the ML estimate of d. This section presents an algorithm which searches for $\hat{d}_{ML}$ based on the results of Section 4.4.2.

The ML estimate of dip can be found by solving

$$\begin{aligned}\hat{d}_{ML} &= \arg\max_{d} \ln p[s(\underline{x})] \\ &= \arg\max_{d}\{-\int_{\underline{y}\in D}\int_{\underline{x}\in D} s(\underline{x})Q(\underline{x},\underline{y})s(\underline{y})d\underline{x}d\underline{y}\} \end{aligned} \tag{4.24}$$

(see (4.20) in Section 4.4.1). Equation (4.24) can be written so that

$$\hat{d}_{ML} = \arg\max_{d=Vd_0}\{-\int_{\underline{y}\in D}\int_{\underline{x}\in D} s(\underline{x})Q[d\cdot(\underline{x}-\underline{y})]s(\underline{y})d\underline{x}d\underline{y}\} \tag{4.25}$$

for some initial value of dip, d_0. From the results of Section 4.4.2 the expression in (4.25) is equivalent to

$$\hat{d}_{ML} = \arg\max_{d=Vd_0}\{-\int_{\underline{y}\in D}\int_{\underline{x}\in D} s(V^T\underline{x})Q[d_0\cdot(\underline{x}-\underline{y})]s(V\underline{y})d\underline{x}d\underline{y}\} \tag{4.26}$$

The rotate and interpolate (RI) algorithm is a method of searching for $\hat{d}_{ML}$ when the data is spatially discrete, not continuous. The RI algorithm is based on (4.26), however, since the data is spatially discrete the algorithm finds $\hat{d}_{RI}$ which is approximately equal to $\hat{d}_{ML}$. The remainder of this section details the RI algorithm and discusses its relationship to ML estimation.

The data used in the RI algorithm is spatially discrete. It is arranged on a rectangular grid and the following notation follows conventions proposed in Section 4.3.2. Since the data is spatially discrete the joint distribution is that of a Gaussian random vector. Consequently, the ML estimate of dip is found by solving

$$\hat{d}_{ML} = \arg\max_{d}\{-\ln|\Lambda| - \underline{s}^T\Lambda^{-1}\underline{s}\} \tag{4.27}$$

where Λ is the covariance matrix.[2]

Evaluation of (4.27) requires calculation of a matrix determinant and a matrix inverse for each value of d. For $\underline{s}$ an n_1n_2-vector these calculations require $o({n_1}^3{n_2}^3)$ multiplies. The RI algorithm requires $o({n_1}^2{n_2}^2)$ multiplies to obtain $\hat{d}_{RI}$ which is approximately $\hat{d}_{ML}$, and it accomplishes this by using the results of Section 4.4.2.

The RI estimate is obtained by solving

$$\hat{d}_{RI} = \arg\max_{d=Vd_0}\{-F[\underline{s}(V\underline{x})]^T\Lambda^{-1}(d_0)F[\underline{s}(V\underline{x})]\} \tag{4.28}$$

[2]Note: The ensuing discussion is identical for both the ρ and $\rho\gamma$ models (see (4.11) and (4.12)).

where F is an interpolation operator. The algorithm suggested by (4.28) is as follows: (1) Calculate the matrix inverse for a particular value of $d = d_0$, (2) rotate the grid the data is defined on by the rotation V, (3) interpolate the data onto the original grid using F, (4) Evaluate the expression in (4.28).

The interpolation operator, F performs the following operations. For each point on the rotated grid, the four closest points on the unrotated grid are found. One of the four points is discarded and the remaining three are used to fit a plane to the values of data at these points. The interpolated data value at the rotated grid point is the value on the plane associated with the rotated grid position. The discarded point is chosen by selecting the three out of four grid points which have minimum covariance. This choice ensures that the interpolated value retains the character of the data in its vicinity.

The RI algorithm is the discrete counterpart to the computation of the ML estimate for spatially continuous data. There are two reasons why (4.28) does not yield $\hat{d}_{ML}$ exactly. First, the data lies on a grid which when rotated does not superimpose on itself (except at dip angles $-\frac{\pi}{2}, 0, \frac{\pi}{2}$). This explains the purpose of the interpolator, F, which obtains values of data on the unrotated grid from values of data on the rotated grid. The interpolation is a model-independent approximation of the data on the unrotated grid which is not equivalent to rotation of the spatially continuous data. Second, the calculation of the determinant specified by (4.27) is ignored in (4.28). This determinant is independent of d for spatially continuous data (see Section 2.5). For spatially discrete data, however, the dip is only approximately independent of the determinant.

The advantage of computing the RI estimate over the straight-forward ML estimate is computational speed. As previously mentioned, the ML estimate requires $o({n_1}^3 {n_2}^3)$ multiplies for each value of d. The RI estimate calculates a matrix inverse only once. For each ensuing value of dip there is a rotation ($o(n_1 n_2)$ multiplies), an interpolation ($o(n_1 n_2)$ multiplies), and a quadratic product ($o({n_1}^2 {n_2}^2)$ multiplies). The computation for the RI algorithm is dominated by the $o({n_1}^2 {n_2}^2)$ calculations which makes it far faster than the algorithm for computing the ML estimate.

At this point we would like to illustrate the use of the RI algorithm on the real

data of Figure 4.4. To apply a constant dip model such as the one used for the RI algorithm to the data of Figure 4.4 requires an assumption. The data displays varying dip in Figure 4.4, however, we assume that it varies slowly. If the dip varies slowly enough then within small windows it can be assumed that the dip is constant. Consequently, to apply the RI algorithm to the data in Figure 4.4 we use a moving window. At each location, the data within the window is assumed to be a constant function of dip. The RI algorithm is used to calculate the estimate of dip at that window location.

In the following two examples the window is a 7×7 array of elements. Thus, according to the previous discussion $n_1 = n_2 = 7$. The window is centered at each of the data positions in Figure 4.4. For each position an estimate for dip is calculated based on the data within the window. When the window extends beyond the boundaries or the data the RI algorithm is adjusted to account for the decreased amount of data (that is, n_1 and/or n_2 are decreased in value).

Figure 4.10 illustrates the use of the RI algorithm when the ρ-model of Section 4.3.2 is used as the model for the data. Figure 4.11 illustrates the algorithm when the $\rho\gamma$-model is used. In Figure 4.10 the model takes the values $\rho = .125$, ${\sigma_n}^2 = .1$. For Figure 4.11 the model uses the value $\rho = .099$, $\gamma = .99$, and ${\sigma_n}^2 = .1$. Approximately 45 minutes of CPU time on a Data General MV10000 computer was required to obtain each of the estimates. This contrasts with the approximately 4 hours required to obtain the dip estimates using the SA algorithm in Figures 4.5 and 4.6.

The estimate in Figures 4.10 and 4.11 appear very similar to the SA estimates in Figures 4.5 and 4.6. There are, however, some spurious local dip estimates along the boundaries in Figures 4.5 and 4.6. In several locations the local dip estimate points in a markedly different direction than its neighbors. This behavior is due to the difficulty in rotating and interpolating data near data boundaries. The points near a boundary have no reasonable interpolated values when they are rotated outside of the boundaries. Consequently, it is not surprising to see a few poor estimates near boundaries nor to see superior estimates in the SA algorithm which includes a dip smoothing term.

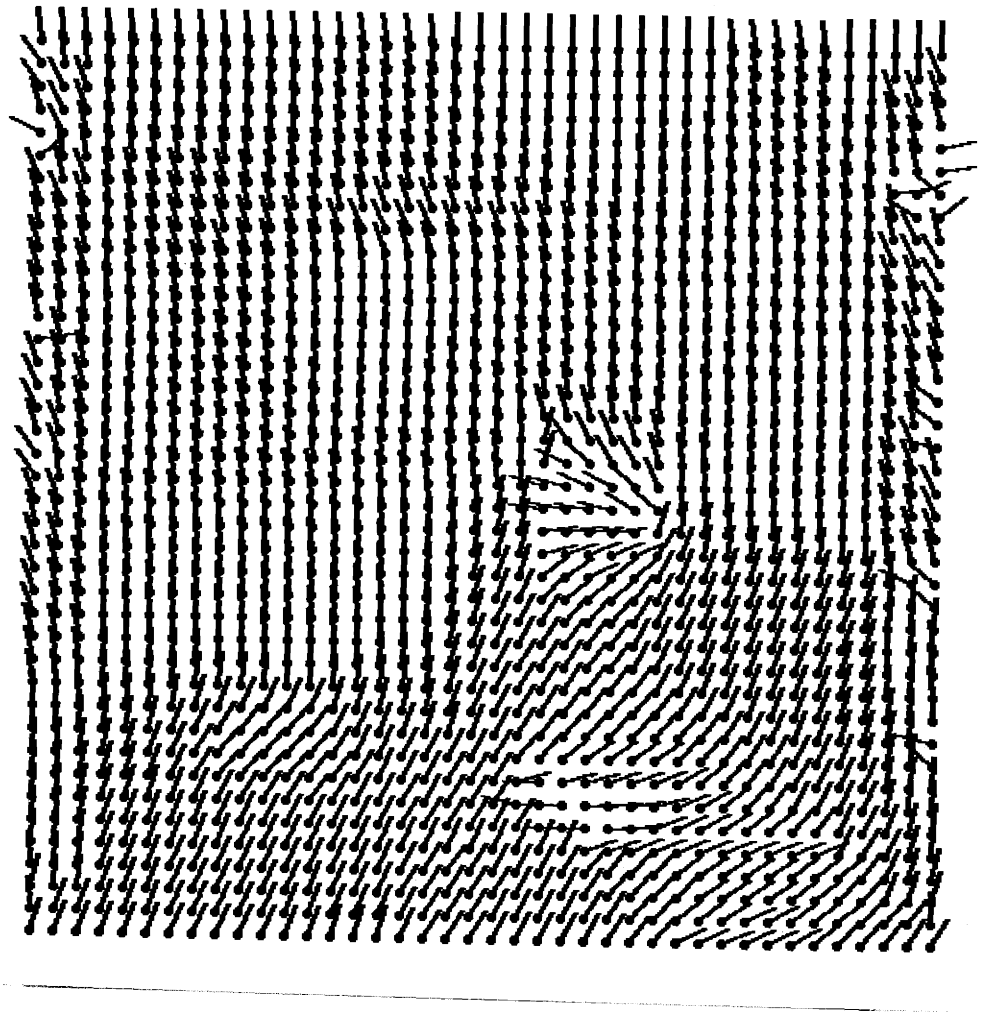

Figure 4.10: Estimate of dip for $\rho = .125$ and $\sigma_n{}^2 = .1$

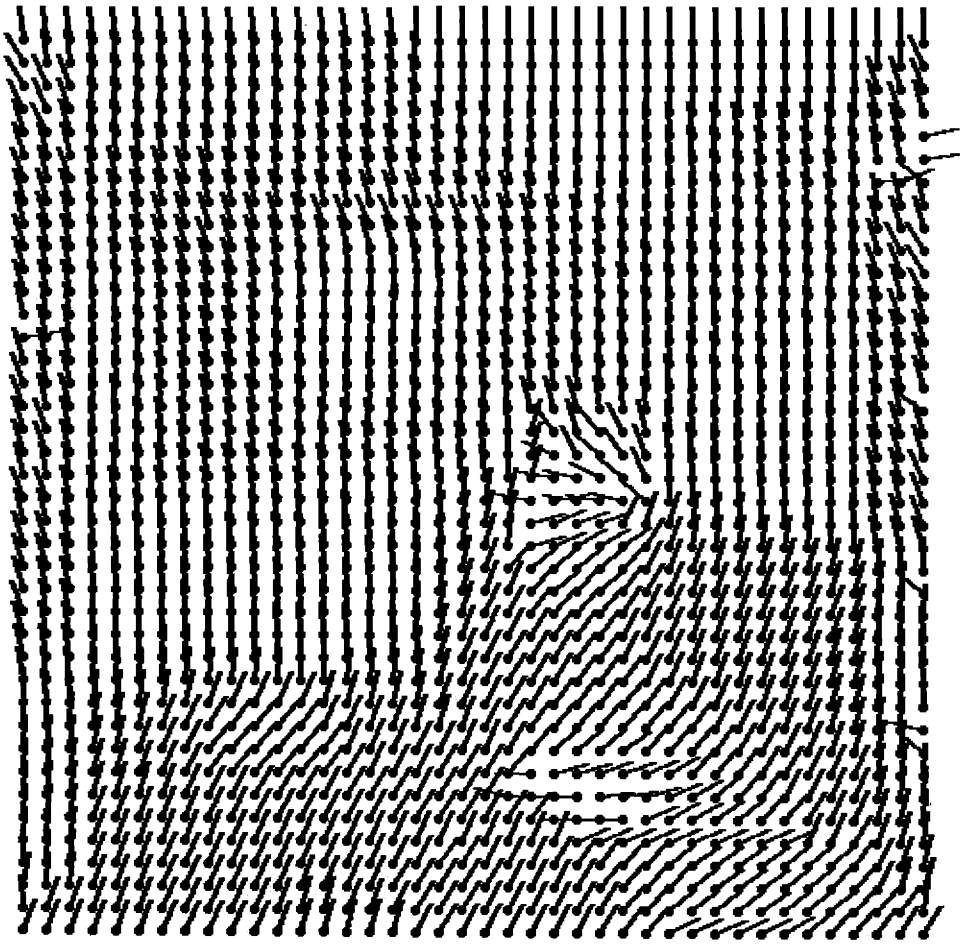

Figure 4.11: Estimate of dip for $\rho = .099$, $\gamma = .99$, and $\sigma_n{}^2 = .1$

The next two sub-sections examine the structure of Λ and Λ^{-1} for a particular value of dip. The new insights obtained from this examination lead to a faster algorithm for estimating dip.

4.4.4 Structure of Covariance for Vertical Dip

The structure of Λ is special since its entries are derived from an ARC function. The RI algorithm of Section 4.4.3 relies on the inverse of Λ at a particular value of d. As is demonstrated in the next sub-section, much can be inferred about the inverse of Λ when $d = [0\ 1]^T$. This section discusses the structure of Λ for this value of dip.

Section 4.3.2 discusses two discrete data models. In each of the models the data vector, $\underline{s}$, represents a two-dimensional arrangement of the data. The first n_2 elements of $\underline{s}$ represent the lowest row of the image, the next n_2 elements represent the next row, etc. When d is chosen to be orthogonal to the rows of the image (i.e. $d = [0\ 1]^T$), the ij^{th} element of the covariance matrix has the following structure. Let l_i and l_j be the column numbers and k_i and k_j be the row numbers of element i and j, respectively, then

$$[\Lambda_\rho]_{ij} = \rho^{|k_i - k_j|} + {\sigma_n}^2 \delta_{ij} \tag{4.29}$$

for the ρ-model in (4.11), and

$$[\Lambda_{\rho\gamma}]_{ij} = \rho^{|k_i - k_j|}\gamma^{|l_i - l_j|} + {\sigma_n}^2 \delta_{ij} \tag{4.30}$$

for the $\rho\gamma$-model in (4.12) (see Section 4.3.2).

From (4.29) and (4.30) it is seen that the covariance matrices are of special forms. The structure of Λ_ρ is block Toeplitz plus a constant times an identity matrix

$$\Lambda = \begin{bmatrix} R_0 & R_1 & \cdots & R_{n_1-1} \\ R_1 & R_0 & \cdots & R_{n_1-2} \\ \vdots & \vdots & & \vdots \\ R_{n_1-1} & R_{n_1-2} & \cdots & R_0 \end{bmatrix} + {\sigma_n}^2 I \tag{4.31}$$

where I is $(n_1 n_2) \times (n_1 n_2)$, R_k is $n_2 \times n_2$, and R_k is the constant matrix

$$R_k = \begin{bmatrix} \rho^k & \rho^k & \cdots & \rho^k \\ \rho^k & \rho^k & \cdots & \rho^k \\ \vdots & \vdots & & \vdots \\ \rho^k & \rho^k & \cdots & \rho^k \end{bmatrix} \tag{4.32}$$

The expression in (4.31) is compactly expressed as a Kronecker product

$$\Lambda_\rho = (R \otimes U) + {\sigma_n}^2 I \tag{4.33}$$

where R is the $n_1 \times n_1$ matrix

$$R = \begin{bmatrix} 1 & \rho & \cdots & \rho^{n_1-1} \\ \rho & 1 & \cdots & \rho^{n_1-2} \\ \vdots & \vdots & & \vdots \\ \rho^{n_1-1} & \rho^{n_1-2} & \cdots & 1 \end{bmatrix} \tag{4.34}$$

and U is the $n_2 \times n_2$ matrix

$$U = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{4.35}$$

The structure of $\Lambda_{\rho\gamma}$ is similar to Λ_ρ in that it is simply represented using a Kronecker product

$$\Lambda_{\rho\gamma} = (R \otimes G) + {\sigma_n}^2 I \tag{4.36}$$

where R is the $n_1 \times n_1$ matrix in (4.34) and G is the $n_2 \times n_2$ matrix

$$G = \begin{bmatrix} 1 & \gamma & \cdots & \gamma^{n_2-1} \\ \gamma & 1 & \cdots & \gamma^{n_2-2} \\ \vdots & \vdots & & \vdots \\ \gamma^{n_2-1} & \gamma^{n_2-2} & \cdots & 1 \end{bmatrix} \tag{4.37}$$

The matrix $\Lambda_{\rho\gamma}$ is block Toeplitz. Furthermore, each of the blocks of $\Lambda_{\rho\gamma}$ are Toeplitz (i.e., $\rho^k G$).

The next section looks at the structures of the inverses of Λ_ρ and $\Lambda_{\rho\gamma}$. The interpretation of these structures provides much insight into the models and suggests fast alternatives to the RI algorithm.

4.4.5 Structure of the Inverse Covariance Matrices for $d = [0\ 1]^T$

Both of the covariances in Section 4.4.4 are highly structured and a great deal can be said about the inverse covariance matrices. This section finds the inverse covariance matrices by using power series expansions. An interpretation of each matrix inverse is given utilizing the most significant terms of the power series.

Structure and Interpretation of Λ_ρ^{-1}

As discussed in Section 4.4.4 the covariance matrix, Λ_ρ, has,the representation

$$\Lambda_\rho = (R \otimes U) + \epsilon I \tag{4.38}$$

where R and U are as in (4.34)-(4.35) and $\epsilon = {\sigma_n}^2$. The inverse of Λ_ρ is found as a power series expansion

$$\begin{aligned} \Lambda_\rho^{-1} &= \frac{1}{\epsilon} S_{-1} + S_0 + \epsilon S_1 + \epsilon^2 S_2 + \cdots \\ &= \sum_{k=-1}^{\infty} \epsilon^k S_k \end{aligned} \tag{4.39}$$

Taking Λ_ρ to be

$$\Lambda_\rho = \mathcal{R} + \epsilon I \tag{4.40}$$

where $\mathcal{R} = R \otimes U$, we have that

$$\begin{aligned} I &= \Lambda_\rho \Lambda_\rho^{-1} \\ &= (\mathcal{R} + \epsilon I)(\sum_{k=-1}^{\infty} \epsilon^k S_k) \\ &= \frac{1}{\epsilon} \mathcal{R} S_{-1} + \sum_{k=0}^{\infty} \epsilon^k (\mathcal{R} S_k + S_{k-1}) \end{aligned} \tag{4.41}$$

Setting the ϵ^0 term in (4.41) to the identity and insisting that all the remaining terms equal zero yields the matrices S_k,

$$S_k = \begin{cases} I_{n_1n_2 \times n_1n_2} - \frac{1}{n_2}(I_{n_1 \times n_1} \otimes U) & k = -1 \\ \frac{(-1)^k}{{n_2}^{k+2}}(R^{-(k+1)} \otimes U) & k = 0, 1, 2, \cdots \end{cases} \tag{4.42}$$

The proof that (4.42) satisfies (4.41) is found in the appendix to this chapter.

Substituting (4.42) into (4.39) yields Λ_ρ^{-1},

$$\begin{aligned} \Lambda_\rho^{-1} &= \frac{1}{\epsilon}(I_{n_1n_2 \times n_1n_2} - \frac{1}{n_2}(I_{n_1 \times n_1} \otimes U)) + \sum_{k=0}^{\infty} \frac{(-\epsilon)^k}{{n_2}^{k+2}}[R^{-k+1} \otimes U] \\ &= \frac{1}{\epsilon}(I_{n_1n_2 \times n_1n_2} - \frac{1}{n_2}(I_{n_1 \times n_1} \otimes U)) + \frac{1}{{n_2}^2}\sum_{k=0}^{\infty}(-\frac{\epsilon}{n_2})^k[R^{-k+1} \otimes U] \\ &= \frac{1}{\epsilon}(I_{n_1n_2 \times n_1n_2} - \frac{1}{n_2}(I_{n_1 \times n_1} \otimes U)) + \frac{1}{{n_2}^2}[R^{-1}\sum_{k=0}^{\infty}(-\frac{\epsilon}{n_2})^k R^{-k}] \otimes U \end{aligned} \tag{4.43}$$

For $|\lambda(\frac{\epsilon}{n_2}R^{-1})| < 1$, (4.43) reduces to

$$\begin{aligned} \Lambda_\rho^{-1} &= \frac{1}{\epsilon}[I_{n_1n_2 \times n_1n_2} - \frac{1}{n_2}(I_{n_1 \times n_1} \otimes U)] + \frac{1}{{n_2}^2}[R^{-1}(I_{n_1 \times n_1} + \frac{\epsilon}{n_2}R^{-1})^{-1}] \otimes U \\ &= \frac{1}{\epsilon}[I_{n_1n_2 \times n_1n_2} - \frac{1}{n_2}(I_{n_1 \times n_1} \otimes U)] + \frac{1}{{n_2}^2}[R + \frac{\epsilon}{n_2}I_{n_1 \times n_1}]^{-1} \otimes U \end{aligned} \tag{4.44}$$

For ϵ small enough (4.44) becomes

$$\Lambda_\rho^{-1} = \frac{1}{\epsilon}[I_{n_1n_2 \times n_1n_2} - \frac{1}{n_2}(I_{n_1 \times n_1} \otimes U)] \tag{4.45}$$

This approximation lends a special interpretation to the inverse covariance matrix. To see the interpretation requires explicit evaluation of (4.45) which looks like

$$\Lambda_\rho^{-1} = \begin{bmatrix} U_{n_2} & 0 & \cdots & 0 \\ 0 & U_{n_2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & U_{n_2} \end{bmatrix} \tag{4.46}$$

where

$$U_{n_2} = \begin{bmatrix} \frac{n_2-1}{n_2} & -\frac{1}{n_2} & \cdots & -\frac{1}{n_2} \\ \frac{-1}{n_2} & \frac{n_2-1}{n_2} & \cdots & \frac{-1}{n_2} \\ \vdots & \vdots & & \vdots \\ \frac{-1}{n_2} & \frac{-1}{n_2} & \cdots & \frac{n_2-1}{n_2} \end{bmatrix} \tag{4.47}$$

Computation of the quadratic product

$$\begin{aligned}
\underline{s}^T \Lambda_\rho^{-1} \underline{s} &\approx \frac{1}{\epsilon} \sum_{k=1}^{n_1} \underline{s}_k{}^T U_{n_2} \underline{s}_k \\
&= \frac{1}{\epsilon} \sum_{k=1}^{n_1} \frac{n_2 - 1}{n_2} \sum_{i=(k-1)n_2+1}^{kn_2} s_i{}^2 - \frac{1}{n_2} \sum_{i=(k-1)n_2+1}^{kn_2} \sum_{j=(k-1)n_2+1,\ j\neq i}^{kn_2} s_i s_j \\
&= \frac{1}{\epsilon} \sum_{k=1}^{n_1} \{ \sum_{i=(k-1)n_2}^{kn_2} s_i{}^2 - \frac{1}{n_2} (\sum_{i=(k-1)n_2}^{kn_2} s_i)^2 \} \\
&= \frac{1}{\epsilon} \sum_{k=1}^{n_1} n_2 \hat{\sigma}_k^2 \qquad (4.48)
\end{aligned}$$

where $\underline{s}_k$ is the k^{th} row of data and $\hat{\sigma}_k^2$ is the sample covariance of the k^{th} row of data.

Thus, (4.48) states that the estimate $\hat{d}_{RI}$ is approximately the value of dip which minimizes the sum of the row data sample covariances. The row data is obtained by rotating the data grid by $d = V d_0$ and interpolating the data onto the unrotated grid. This interpretation is intuitively pleasing. Furthermore, computation of the sum of the row sample covariances requires only $o(n_1 n_2)$ multiplies, a substantial savings over the exact calculation of $\hat{d}_{RI}$.

The result in (4.48) is obtained by approximating the inverse covariance matrix Λ_ρ by the first term in (4.39). As already shown, the first term of the expansion results in a block diagonal matrix with zero blocks off the main diagonal. The effect of including the second term in (4.39) results in a matrix which is block tri-diagonal with zero blocks off the three principal diagonals. Consequently, when the second order term is included, the interaction between adjacent rows of data is included in the computation of the quadratic product with the data. As more terms in the expansion are included, rows of data which are farther apart are included in the calculation of the quadratic product.

Before proceeding to a similar analysis for the inverse structure of $\Lambda_{\rho\gamma}$ it should be noted that there is a striking similarity between (4.48) and beamforming (see [8]). Beamforming is the synthetic steering of a fixed antenna array for the purpose of maximizing the power of an incoming plane wave. The steering is accomplished by appending a linearly increasing set of delays on the antenna array. This is followed

by performing sums of inner products on the incoming data. The direction which maximizes the sum of inner products is the direction from which the plane wave emanates.

The linearly increasing set of delays in the beamforming problem is analogous to the rotate and interpolate operation proposed by the RI algorithm. The RI algorithm then proceeds to approximately form the sum of covariances which is equivalent to taking a sum of inner products. Thus, there is a strong relationship between beamforming and the ρ-model of Section 4.3.2.

Structure and Interpretation of $\Lambda_{\rho\gamma}^{-1}$

The covariance matrix $\Lambda_{\rho\gamma}$ can be expressed as

$$\Lambda_{\rho\gamma} = (R \otimes G) + \epsilon I \tag{4.49}$$

where R and G are as in (4.34) and (4.37) and $\epsilon = {\sigma_n}^2$. Again the inverse covariance matrix is represented by a power series expansion,

$$\Lambda_{\rho\gamma}^{-1} = \sum_{k=0}^{\infty} \epsilon^k S_k \tag{4.50}$$

Taking $\Lambda_{\rho\gamma}$ to be

$$\Lambda_{\rho\gamma} = \mathcal{R} + \epsilon I \tag{4.51}$$

where $\mathcal{R} = R \otimes G$, we have that

$$\begin{aligned} I &= \Lambda_{\rho\gamma}\Lambda_{\rho\gamma}^{-1} \\ &= (\mathcal{R} + \epsilon I)(\sum_{k=0}^{\infty} \epsilon^k S_k) \\ &= \mathcal{R}S_0 + \sum_{k=0}^{\infty} \epsilon^k (\mathcal{R}S_{k+1} + S_k) \end{aligned} \tag{4.52}$$

Setting the first term in (4.52) to the identity and the remaining terms to zero yields

$$S_k = \begin{cases} \mathcal{R}^{-1} & k = 0 \\ (-1)^k \mathcal{R}^{-(k+1)} & k = 1, 2, \ldots \end{cases} \tag{4.53}$$

Thus,

$$\begin{aligned}\Lambda_{\rho\gamma}^{-1} &= \sum_{k=0}^{\infty} \epsilon^k (-1)^k \mathcal{R}^{-(k+1)} \\ &= \mathcal{R}^{-1} \sum_{k=0}^{\infty} (-\epsilon \mathcal{R}^{-1})^k \end{aligned} \tag{4.54}$$

When $|\lambda(\epsilon R^{-1}| < 1$ the expression in (4.54) may be written

$$\begin{aligned}\Lambda_{\rho\gamma}^{-1} &= \mathcal{R}^{-1}(I + \epsilon \mathcal{R}^{-1})^{-1} \\ &= (\mathcal{R} + \epsilon I)^{-1} \end{aligned} \tag{4.55}$$

For ϵ small a good approximation to (4.55) is

$$\begin{aligned}\Lambda_{\rho\gamma}^{-1} &\approx \mathcal{R}^{-1} \\ &= (R \otimes G)^{-1} \\ &= R^{-1} \otimes G^{-1} \end{aligned} \tag{4.56}$$

where

$$R^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 \\ -\rho & (1+\rho^2) & -\rho & \cdots & 0 \\ 0 & -\rho & (1+\rho^2) & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.57}$$

and

$$G^{-1} = \frac{1}{1-\gamma^2} \begin{bmatrix} 1 & -\gamma & 0 & \cdots & 0 \\ -\gamma & (1+\gamma^2) & -\gamma & \cdots & 0 \\ 0 & -\gamma & (1+\gamma^2) & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.58}$$

Consequently, (4.56) looks like

$$\Lambda_{\rho\gamma}^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} G^{-1} & -\rho G^{-1} & \underline{0} & \cdots & \underline{0} \\ -\rho G^{-1} & (1+\rho^2)G^{-1} & -\rho G^{-1} & \cdots & \underline{0} \\ \underline{0} & -\rho G^{-1} & (1+\rho^2)G^{-1} & \cdots & \underline{0} \\ \vdots & \vdots & \vdots & & \vdots \\ \underline{0} & \underline{0} & \underline{0} & \cdots & G^{-1} \end{bmatrix} \tag{4.59}$$

The quadratic product of the data with (4.59) yields

$$\begin{aligned}\underline{s}^T\Lambda_{\rho\gamma}^{-1}\underline{s} \approx \quad & (\frac{1}{1-\rho^2})\{\underline{s}_1{}^T G^{-1}\underline{s}_1 + \underline{s}_{n_1}{}^T G^{-1}\underline{s}_{n_1} \\ & + \sum_{k=2}^{n_1-1}(1+\rho^2)\underline{s}_k{}^T G^{-1}\underline{s}_k - 2\rho\sum_{k=1}^{n_1-1} s_k{}^T G^{-1}\underline{s}_{k+1}\} \end{aligned} \tag{4.60}$$

Now assuming $\rho \approx 0$ and $\gamma \approx 1$ makes (4.60) look like

$$\begin{aligned}\underline{s}^T\Lambda_{\rho\gamma}^{-1}\underline{s} &\approx \underline{s}_1{}^T G^{-1}\underline{s}_1 + \underline{s}_{n_1}{}^T G^{-1}\underline{s}_{n_1} + \sum_{k=2}^{n_1-1}\underline{s}_k{}^T G^{-1}\underline{s}_k \\ &= \sum_{k=1}^{n_1}\underline{s}_k{}^T G^{-1}\underline{s}_k \\ &= \sum_{k=1}^{n_1}\{(\frac{1}{1-\gamma^2})[\sum_{i=(k-1)n_2+2}^{kn_2-1} 2s_i{}^2 - 2\sum_{i=(k-1)n_2+1}^{kn_2-1} s_i s_{i+1} + s_{(k-1)n_2+1}{}^2 + s_{kn_2}{}^2]\} \\ &= \sum_{k=1}^{n_1}\{(\frac{1}{1-\gamma^2})\sum_{i=(k-1)n_2+1}^{kn_2-1}(s_i - s_{i+1})^2\} \end{aligned} \tag{4.61}$$

The inner sum in (4.61) is the sum of squared differences of adjacent data elements for row k. Thus, for the case where the covariance of the data can be modeled by $\Lambda_{\rho\gamma}$ when $d = [0\ 1]^T$ the estimate $\hat{d}_{RI}$ is approximately as in (4.61). That is, $\hat{d}_{RI}$ is the value of d which after rotating and interpolating the data minimizes the sum of the sum of squared differences along rows. This, too, is intuitively pleasing since it is expected that the variation in data along rows is smallest when the data has been rotated by the best value of dip. As in the case of the ρ-model, the more terms of the expansion that are included in the calculation of the inverse covariance matrix the more interaction there is between rows of data which are farther apart from each other.

The next section proposes an alternative to the RI algorithm. This new algorithm is faster than the RI algorithm and it achieves its improved computational speed using the results discussed in this section.

4.4.6 The Approximate Rotate and Interpolate Algorithm

The examination of the structure for Λ_ρ and $\Lambda_{\rho\gamma}$ in Section 4.4.5 gave much insight into the nature of the RI algorithm. After certain approximations are made the

computational complexity of the RI algorithm can be decreased from $o(n_1{}^2 n_2{}^2)$ multiplies to $o(n_1 n_2)$ multiplies. The purpose of this section is to present the Approximate Rotate and Interpolate (ARI) algorithm which is based upon the analysis of Section 4.4.5.

The ARI algorithm finds $\hat{d}_{ARI}$ which is approximately equal to $\hat{d}_{RI}$. The algorithm is very similar to the RI algorithm and it proceeds as follows,

$$\hat{d}_{ARI} = \arg \max_{d=Vd_0} \{-F[\underline{s}(V\underline{x})]^T K F[\underline{s}(V\underline{x})]\} \tag{4.62}$$

where K is an $n_1 n_2 \times n_1 n_2$ block diagonal matrix

$$K = \begin{bmatrix} A & 0 & \cdots & 0 \\ 0 & A & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & A \end{bmatrix} \tag{4.63}$$

and A is an $n_2 \times n_2$ matrix.

In the case of the ρ-model covariance given in (4.11) we take

$$A = \begin{bmatrix} \frac{n_2-1}{n_2{}^2} & -\frac{1}{n_2{}^2} & \cdots & -\frac{1}{n_2{}^2} \\ -\frac{1}{n_2{}^2} & \frac{n_2-1}{n_2{}^2} & \cdots & -\frac{1}{n_2{}^2} \\ \vdots & \vdots & & \vdots \\ -\frac{1}{n_2{}^2} & -\frac{1}{n_2{}^2} & \cdots & \frac{n_2-1}{n_2{}^2} \end{bmatrix} \tag{4.64}$$

Combining (4.64) with (4.63) specifies the ARI algorithm to search for $\hat{d}_{ARI}$ by: (1) Rotating the data grid by V, (2) interpolating the data onto the old grid using F, (3) evaluate the sum of the row sample covariances (see Section 4.4.5 (4.48)). The value of $d = Vd_0$ which minimizes the sum of the row sample covariances is $\hat{d}_{ARI}$.

The procedure of calculating $\hat{d}_{ARI}$ is computationally efficient. Calculation of the rotation of the grid is $o(n_1 n_2)$ multiplies, the interpolation is $o(n_1 n_2)$ multiplies, and the quadratic product of the data with K is $o(n_1{}^{\frac{3}{2}} n_2{}^{\frac{3}{2}})$. However, since K has the effect of calculating sample covariances along rows of the data the computation can be made in $o(n_1 n_2)$ multiplies. This can be seen by expanding the computation

of the k^{th} row, $\underline{s}_k$, with the matrix A

$$\begin{aligned}
\underline{s}_k{}^T A \underline{s}_k &= \sum_{i=(k-1)n_2+1}^{kn_2} \frac{n_2-1}{{n_2}^2} {s_i}^2 - \sum_{i=(k-1)n_2+1}^{kn_2} \sum_{j\neq i} \frac{1}{{n_2}^2} s_i s_j \\
&= \sum_{i=(k-1)n_2+1}^{kn_2} \frac{1}{{n_2}^2} {s_i}^2 - \sum_{i=(k-1)n_2+1}^{kn_2} \sum_{j=(k-1)n_2+1}^{kn_2} \frac{1}{{n_2}^2} s_i s_j \\
&= \sum_{i=(k-1)n_2+1}^{kn_2} \frac{1}{{n_2}^2} {s_i}^2 - \left(\frac{1}{n_2} \sum_{i=(k-1)n_2+1}^{kn_2} s_i\right)^2 \qquad (4.65)
\end{aligned}$$

Calculation of (4.65) requires n_2 multiplies for the first term and one multiply for the second term. Since (4.65) is calculated once for each of the n_1 rows there are $n_1 n_2 + n_2$ multiplies.

In the case of the covariance model in (4.12) we take

$$A = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \qquad (4.66)$$

When (4.66) is used in the ARI algorithm, rotation and interpolation of the data is followed by calculating the sum of squared differences of the data along rows (see Section 4.4.3). Thus,

$$\begin{aligned}
\underline{s}_k{}^T A \underline{s}_k &= {s_{(k-1)n_2+1}}^2 + {s_{kn_2}}^2 + 2\left(\sum_{i=(k-1)n_2+2}^{kn_2-1} {s_i}^2 - \sum_{i=(k-1)n_2+1}^{kn_2-1} s_i s_{i+1}\right) \\
&= \sum_{i=(k-1)n_2+1}^{kn_2-1} (s_i - s_{i+1})^2 \qquad (4.67)
\end{aligned}$$

which is of $o(n_2)$ multiplies. Thus, for A as in (4.66) the ARI algorithm is also of $o(n_1 n_2)$ multiplies in computational complexity.

As in Section 4.4.3 we illustrate the use of the ARI algorithm on the real data set in Figure 4.4. The moving window is a 7×7 array of elements and there are no parameters to set since the approximations which result in the ARI algorithm require the parameters to have special values. Figures 4.12 and 4.13 illustrate the

ARI algorithm for the A matrix as in (4.64) and (4.66), respectively. As can be seen by comparison with the examples in Section 4.4.3 there is only nominal difference between the RI and ARI estimates of dip on this data set. The amount of CPU time to obtain each of the estimates was approximately 15 minutes. This is in comparison to the 45 minutes needed for the estimates computed using the RI algorithm.

The next sub-section describes a modification of the ARI algorithm. The modification is an approximation of rotation and interpolation. The resulting estimation procedure is faster than the ARI algorithm.

4.4.7 The Projection Algorithm

The signal processing algorithms in this section have relied on rotating and interpolating the data. These two operations are computationally expensive and data dependent. This sub-section describes a modification of the ARI algorithm. The new algorithm replaces rotation and interpolation of the data with projections of the data locations along the dip direction. Consequently, the new procedure is termed the PROJ algorithm.

The PROJ algorithm calculates an estimate of dip

$$\hat{d}_{PROJ} = \arg\max_{E_d}\{-(E_d\underline{s})^T K(E_d\underline{s})\} \tag{4.68}$$

where E_d is the permutation matrix appearing in (4.2) of Section 4.1 and K is one of two matrices in Section 4.4.6 (see (4.62), (4.63), (4.64), and (4.66)). The novelty of the PROJ algorithm is that the data is re-ordered according to projections along the dip direction. This re-ordering is an approximation to the rotate and interpolate procedure performed by the RI and ARI algorithms.

To understand why the permutation matrix, E_d, can be considered an approximation to rotation and interpolation, consider the problem of determining which data points in the unrotated, uninterpolated data would be closest in value to the data values in the rotated interpolated data. Figure 4.14a shows how the rotated data grid looks and Figure 4.14b shows the unrotated data grid. Figure 4.14 also shows a correspondence between the data locations in the two grids illustrating

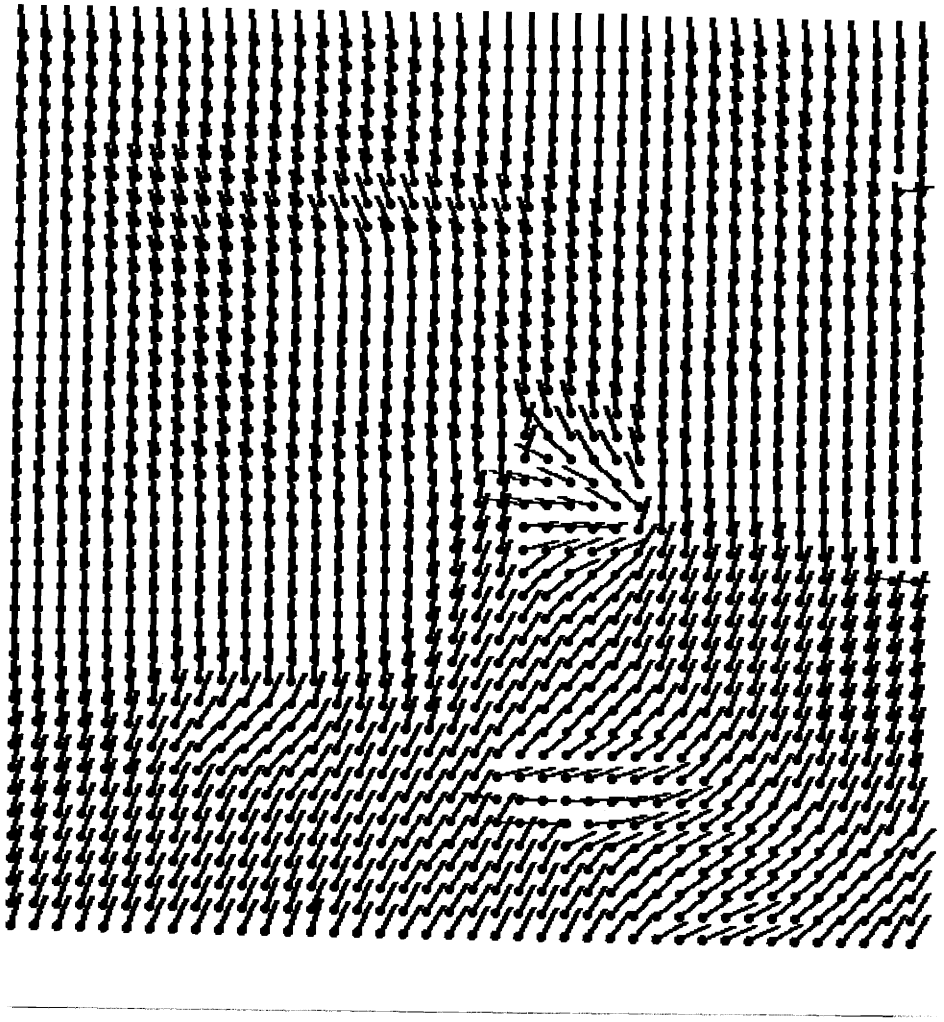

Figure 4.12: Estimate of dip using the ARI algorithm and the A matrix of the ρ-model

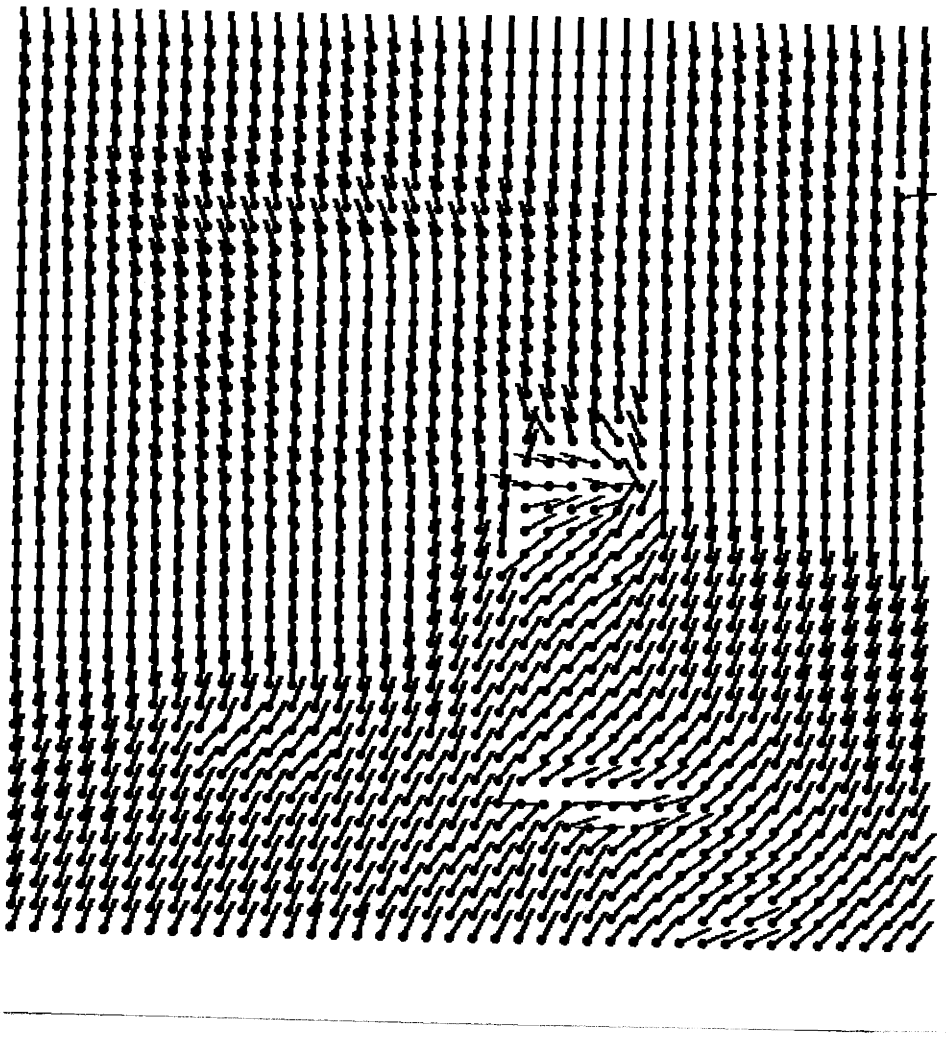

Figure 4.13: Estimate of dip using the ARI algorithm and the A matrix of the $\rho\gamma$-model

a solution to the above question. Projection is an approximation to rotation and interpolation.

Note that since the orderings of the data are independent of data values they can all be precomputed for each value of dip. Clearly, there are only a finite number of orderings of the data due to projection and, consequently, only a finite number of dip values can be resolved.

Once again the data of Figure 4.4 is employed this time to illustrate the use of the PROJ algorithm. Figures 4.15 and 4.16 illustrate the dip estimates from the PROJ algorithm based on the ρ and $\rho\gamma$-models, respectively. The main difference between the PROJ estimates and the ARI estimates is seen in Figure 4.16. There is quite a bit of degradation in the quality of the estimate in Figure 4.16 compared to that in Figure 4.13. This is a result of the fact that the PROJ algorithm is twice removed from the RI algorithm by approximations.

The result in Figure 4.15, however, looks similar to its counterpart from the ARI algorithm in Figure 4.12. The ρ-model of the PROJ algorithm employs much more lateral smoothing within each neighborhood than does the $\rho\gamma$-model. This accounts for the superior performance as the estimate of dip at each point is based in effect on looking at a wider field of data.

One advantage that the PROJ algorithm has over the ARI algorithm is that estimates of dip near boundaries are less likely to be biased. This is because the PROJ algorithm only projects data within the boundaries. The ARI and RI algorithms attempt to interpolate data points outside of the data boundaries. The second advantage that the PROJ algorithm has over the ARI algorithm is speed. The estimates in Figures 4.15 and 4.16 took approximately 5 minutes of CPU time each. This compares with 15 minutes required by the ARI algorithm to calculate the estimates in Figures 4.12 and 4.13.

The next sub-section presents the final algorithm of this section. This last algorithm improves the performance of the PROJ algorithm and serves as a final link in explaining the relationship between the models in Section 4.1 and the models in Section 4.3.

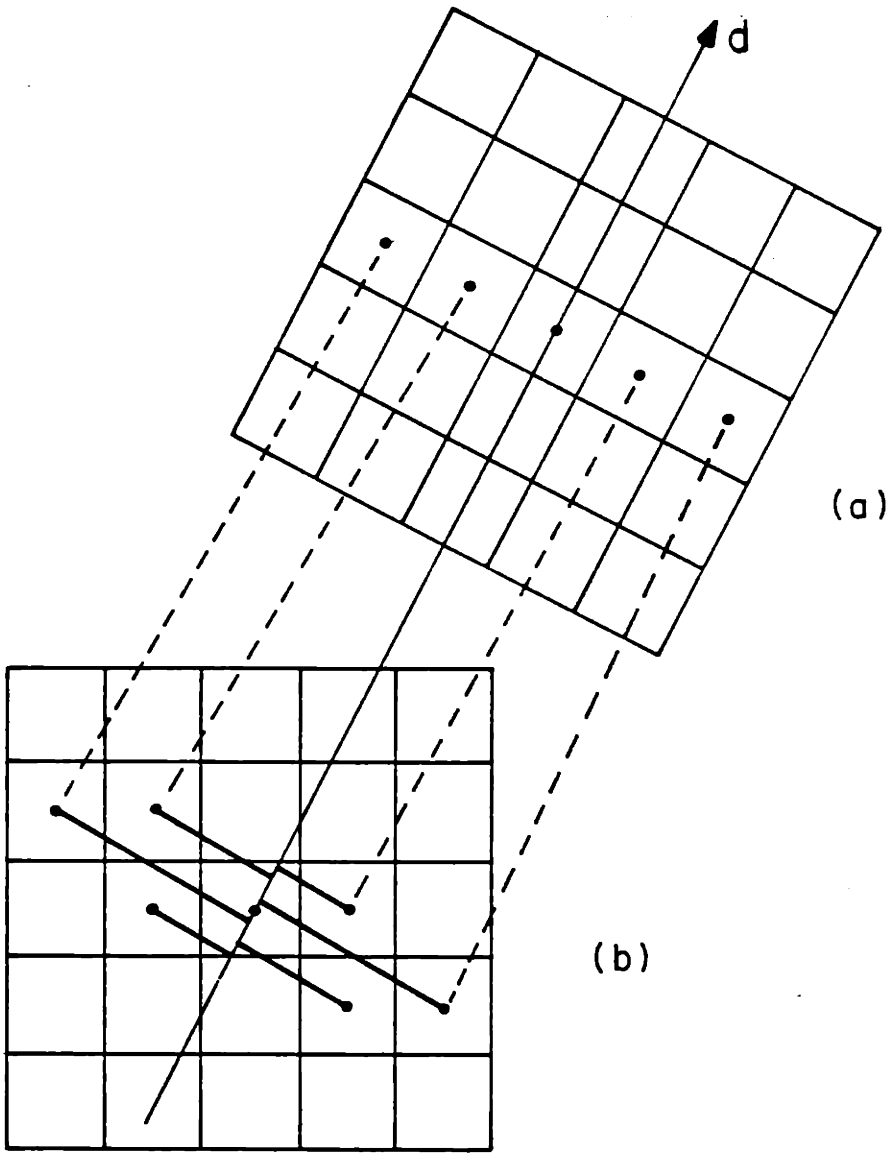

Figure 4.14: Approximation of Rotation and Interpolation

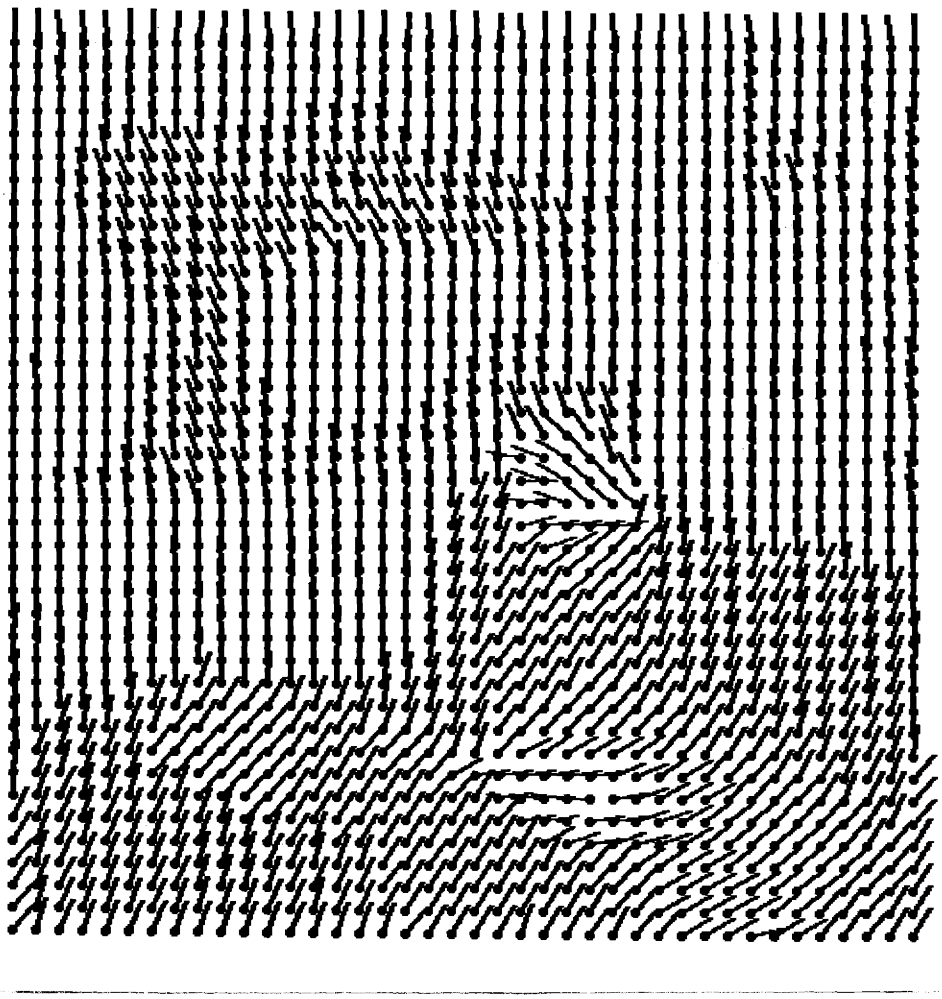

Figure 4.15: Estimate of dip using PROJ algorithm with the ρ-model

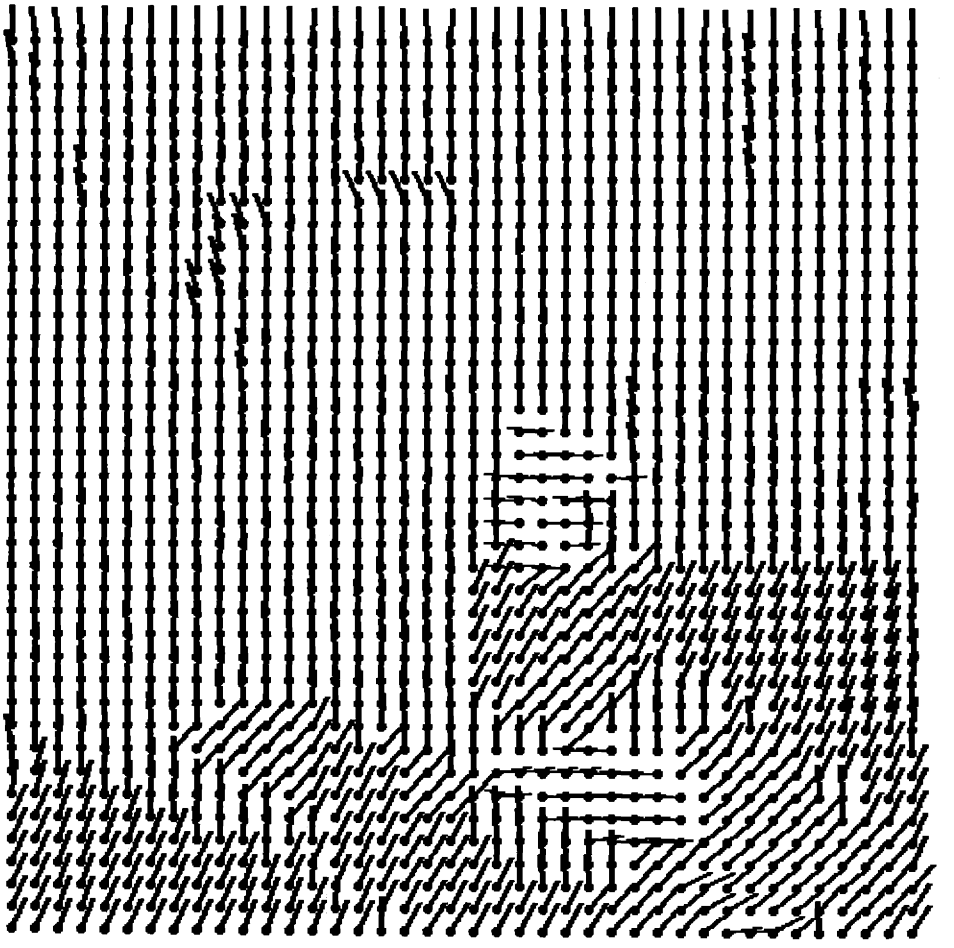

Figure 4.16: Estimate of dip using PROJ algorithm with the $\rho\gamma$-model

4.4.8 The Smoothed Projection Algorithm

The last section discussed the PROJ algorithm which was based on the ARI algorithm and an approximation of the rotate and interpolate procedure. A comparison of the dip estimates in Figure 4.16 using the PROJ algorithm and Figure 4.11 using the RI algorithm shows a substantial loss of quality. This is not surprising since the PROJ algorithm is twice removed from the RI algorithm by approximation methods. This section proposes a new algorithm which improves the estimate quality of the PROJ algorithm without significantly increasing the computational burden.

The new algorithm is termed the Smoothed Projection (SPROJ) algorithm. The SPROJ algorithm modifies the PROJ algorithm by using a slightly different matrix for the quadratic product computation. Consequently, referring to (4.68)

$$\hat{d}_{SPROJ} = \arg\max_{E_d}\{-(E_d\underline{s})^T\tilde{K}(E_d\underline{s})\} \tag{4.69}$$

where $\tilde{K}$ is related to K in the following manner. The matrix K is a block diagonal matrix (see (4.63)) composed of zero blocks on the off-diagonal and the matrix A on the diagonal. For each row of data, the A matrix computes either the sample variance or sum of squared differences with the data. The $\tilde{K}$ matrix treats each consecutive n_2 elements of the data as a row, computes the sample variance or sum of squared differences of these n_2 elements, and takes the sum of each of these.

Thus, $\tilde{K}$ can be written as

$$\tilde{K} = \sum_{i=1}^{n_2n_1-n_2+1} K_i \tag{4.70}$$

where each K_i has a single A matrix embedded in it and zeros everywhere else. The matrix K_i has the $(1,1)$ element of the A matrix at its (i,i) position and the $(1,2)$ element at its $(i,i+1)$ position, etc. Thus,

$$[K_i]_{kl} = \begin{cases} [A]_{k-i+1,l-i+1} & i \le k,l \le i+n_2-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.71}$$

As was shown in Section 4.4.6, the quadratic product of the data with the A matrix requires either n_2 or n_2+1 multiplies. This means that the SPROJ algorithm

requires no more than $(n_2+1)(n_1n_2-n_2+1)$ multiplies. This is an increase in computation over the PROJ algorithm which requires no more than $(n_1n_2+n_2)$ multiplies. However, the SPROJ algorithm is still faster than the ARI algorithm due to the precomputation of the orderings of the data.

The point of this section is that the SPROJ algorithm serves as the basis for the model described in Section 4.1. This is the model that was used in calculating the SA estimate of dip. The SA algorithm calculates the function within the braces in (4.69) for each neighborhood. In addition the SA algorithm also calculates a smoothing term which evaluates a cost associated with adjacent dip values. The model in Section 4.1 is a joint model for data and dip. This distribution is easily written as the product of a distribution for the data conditioned on dip times a prior dip distribution

$$\begin{aligned} p(\underline{s},\underline{d}) &= p(\underline{s}|\underline{d})p(\underline{d}) \\ &= \frac{1}{Z_sZ_d}\exp\{-\sum\sum G_{ij}s_is_j\}\exp\{-\sum\sum D_{ij}|d_i\cdot d_j|\} \end{aligned} \tag{4.72}$$

where Z_s and Z_d are appropriate scale factors. The SPROJ estimate of dip is a collection of independent ML estimates. The SA estimate of dip specifies a prior dip field distribution, and, consequently, yields an MAP estimate of dip.

Figures 4.17 and 4.18 show the SPROJ estimates of dip based on the ρ and $\rho\gamma$-models, respectively. The estimate in Figure 4.18 is much closer to the RI estimate in Figure 4.13 than is the ARI estimate in Figure 4.16. The estimates in Figures 4.17 and 4.18 took 10 minutes of CPU time compared to the 5 minutes used by the PROJ algorithm and 15 minutes used by the ARI algorithm.

The next section does a detailed analysis of the algorithms presented so far in this chapter.

4.5 Examples

Most of the examples in this section are concerned with the use of the RI algorithm of Section 4.4.3. The reason for this is that the RI algorithm depends on several parameters, namely ρ, γ, and ${\sigma_n}^2$, and one of the objectives of this section is to

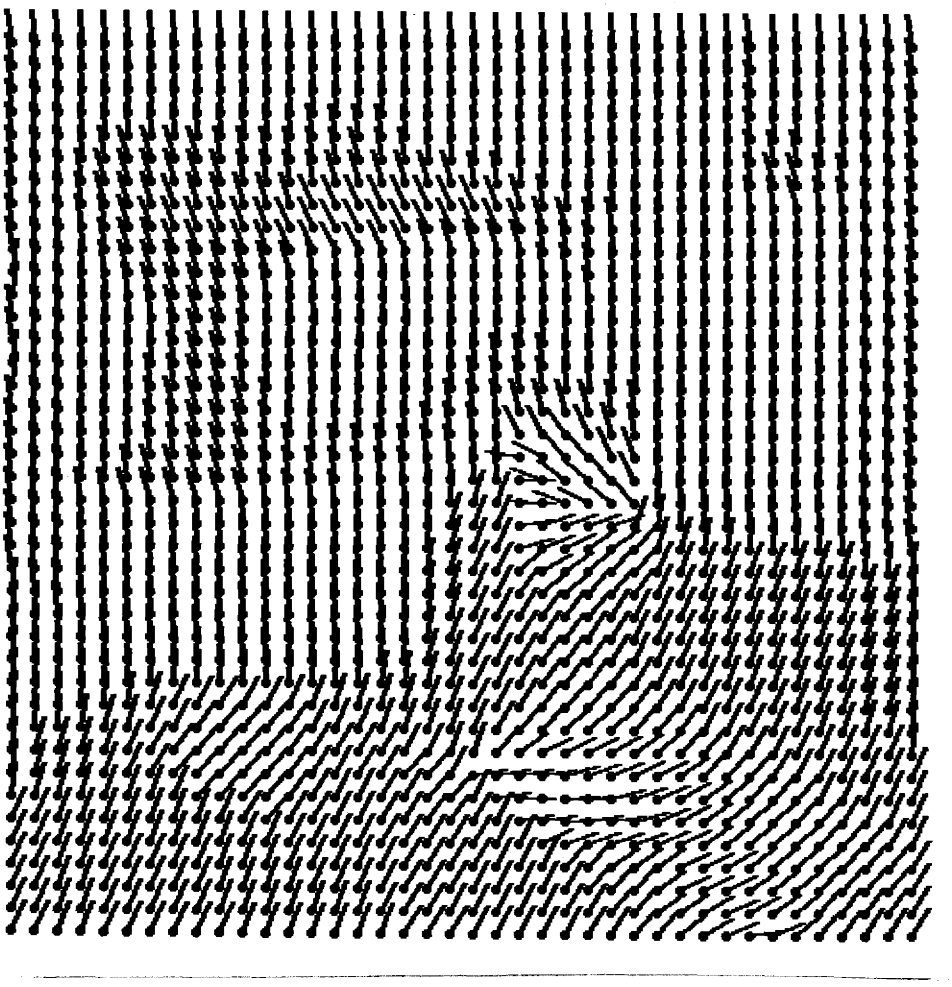

Figure 4.17: Smoothed projection algorithm estimate of dip based on ρ-model

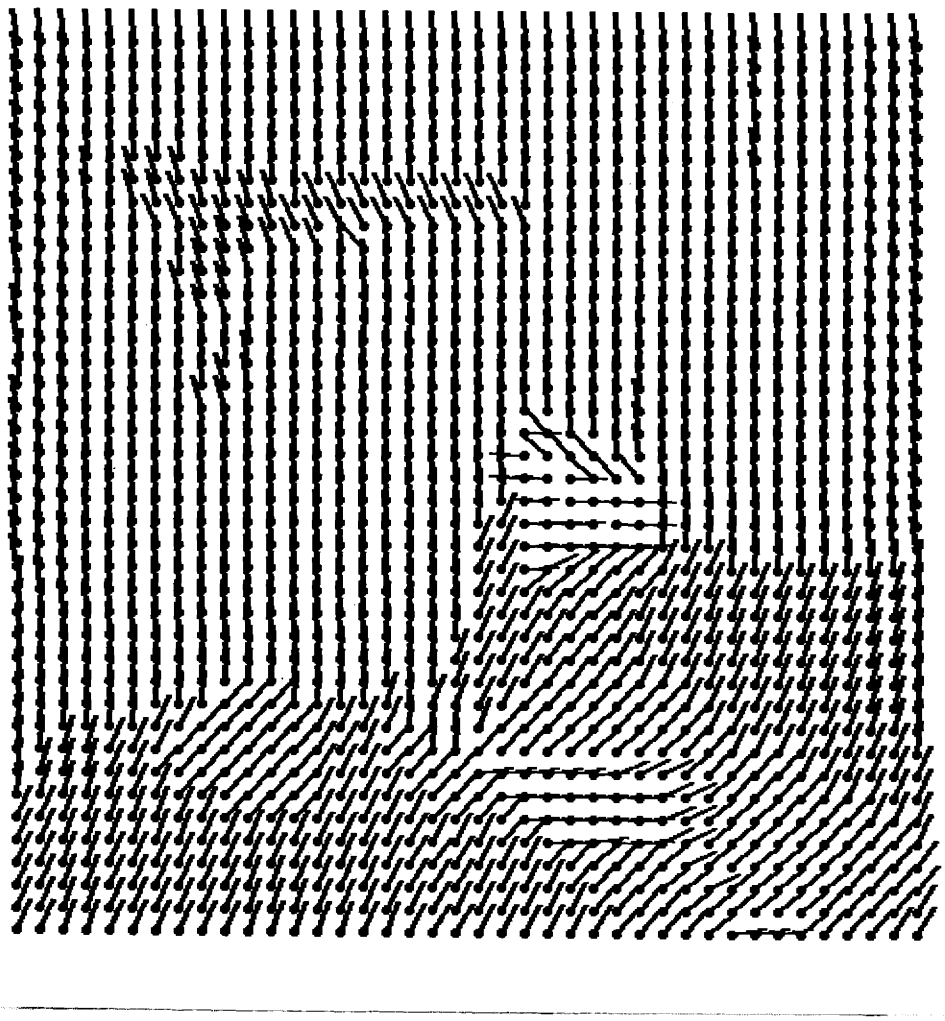

Figure 4.18: Smoothed projection algorithm estimate of dip based on $\rho\gamma$-model

illustrate the effect these parameters have on the RI algorithm. Examples are also presented which comparatively illustrate the performance of the ARI, PROJ, SPROJ, and SA algorithms.

As in the examples section of Chapter 3, the data set used to illustrate the following examples is a constant dip field. The data is illustrated in Figure 4.19. This data was deterministically generated so that the bed boundaries have a slope of $\frac{1}{2}$ and display various contrasts between adjacent beds. The bed boundaries are vertically separated by five pixels. Consequently, the beds are 4.47 pixels thick in the dip direction.

This section is composed of three major sets of examples. The first set illustrates the effect of the parameters ρ and ${\sigma_n}^2$ on the RI algorithm based on the ρ-model in (4.11). The second set of examples illustrates the effect of parameters ρ, γ, and ${\sigma_n}^2$ on the RI algorithm based on the $\rho\gamma$-model in (4.12). The final set compares results of the ARI, PROJ, SPROJ, and SA algorithms to those of the RI algorithm.

4.5.1 Examples Using the RI Algorithm with the ρ Model

There are six examples in this section. The examples are comprised of two subsets. The first subset illustrates the RI algorithm using the ρ-model in (4.11) with a small value for ${\sigma_n}^2$. The second subset uses a large value for ${\sigma_n}^2$. In each of the two subsets the value of ρ takes on large, small, and intermediate values.

The first three examples take ${\sigma_n}^2 = .01$. The values of ρ are .01, .25, and .9, respectively. The examples are illustrated in Figures 4.20-4.22. As can be seen, the quality of the estimates in these figures are good except near the boundaries. The left and right boundaries show some individual dip estimates which are extremely poor. The reason for spurious dip estimates near the boundaries is due to the rotation and interpolation part of the algorithm. It is difficult to obtain good interpolations in these regions since rotation of the data moves grid points outside of the image boundaries. Consequently, there are no data points from which to interpolate.

It is interesting to note that the estimates in Figures 4.20-4.22 are seemingly independent of the value for ρ. This is not surprising if one examines the power

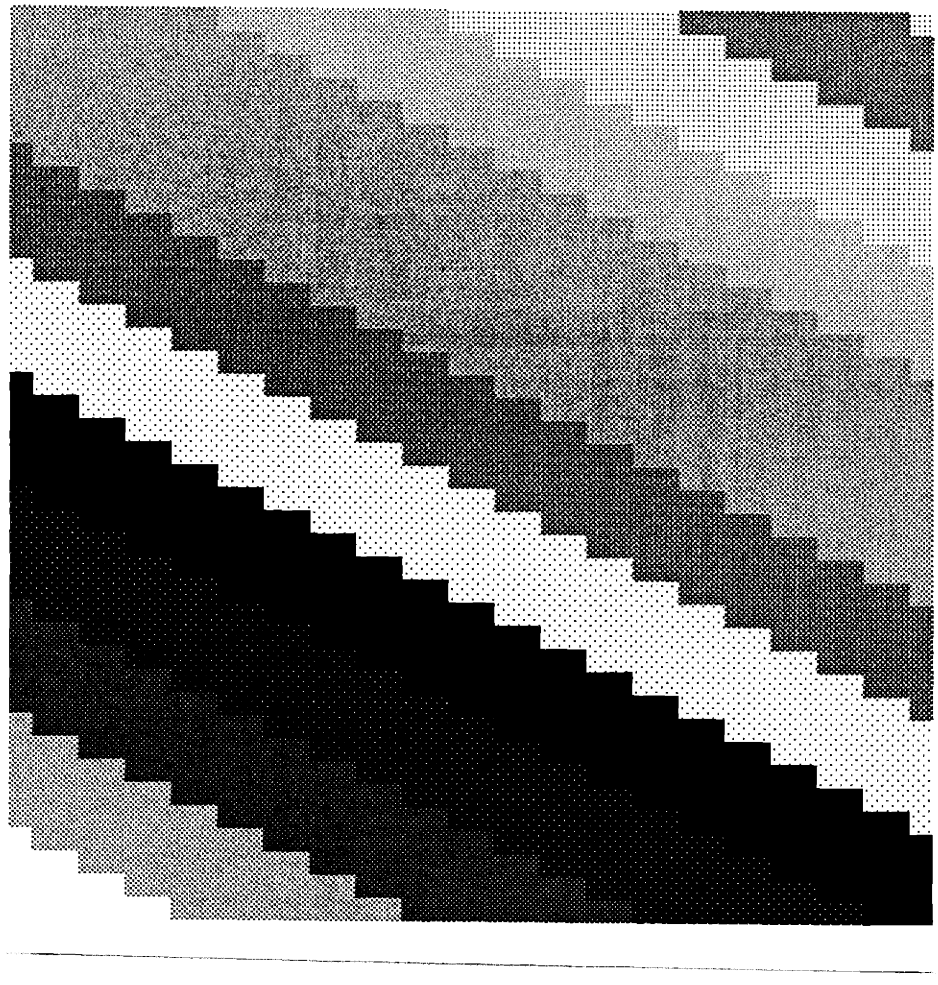

Figure 4.19: Synthetic, non-binary, constant dip data with $d = [1\ 2]^T$ and with beds 4.47 pixels thick in the dip direction

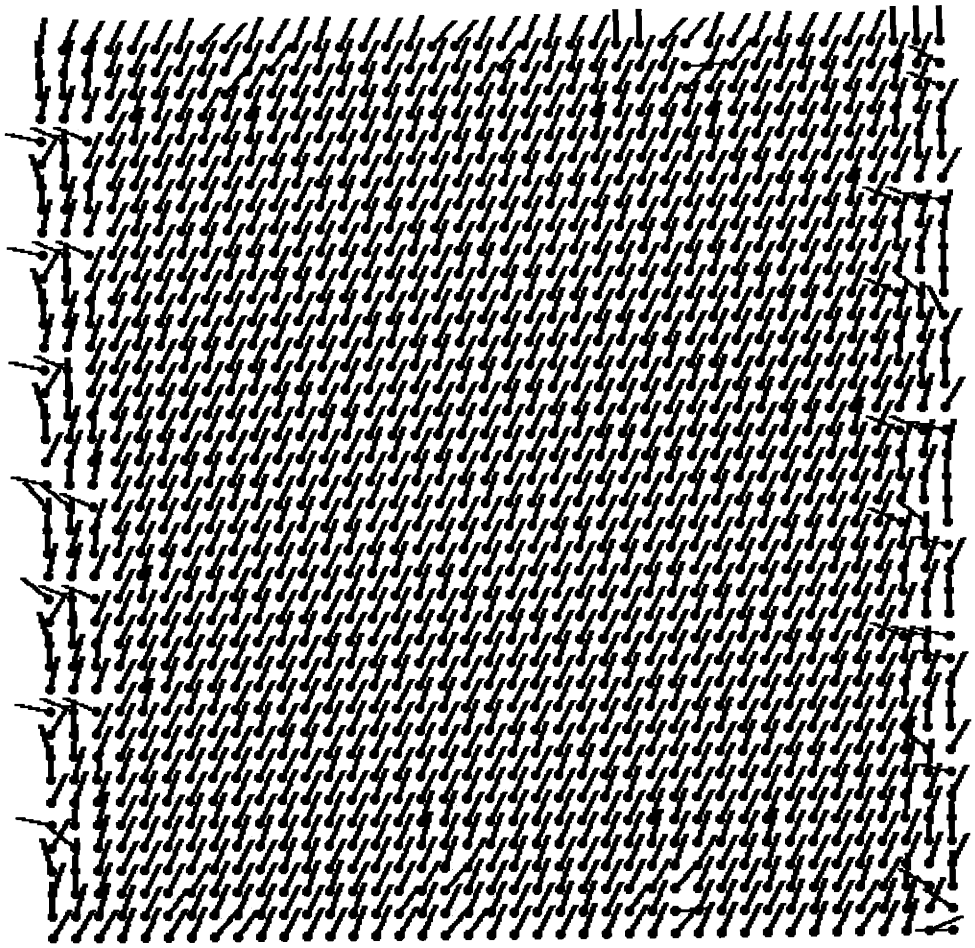

Figure 4.20: RI algorithm using the ρ-model with parameters $\rho = .01$, ${\sigma_n}^2 = .01$

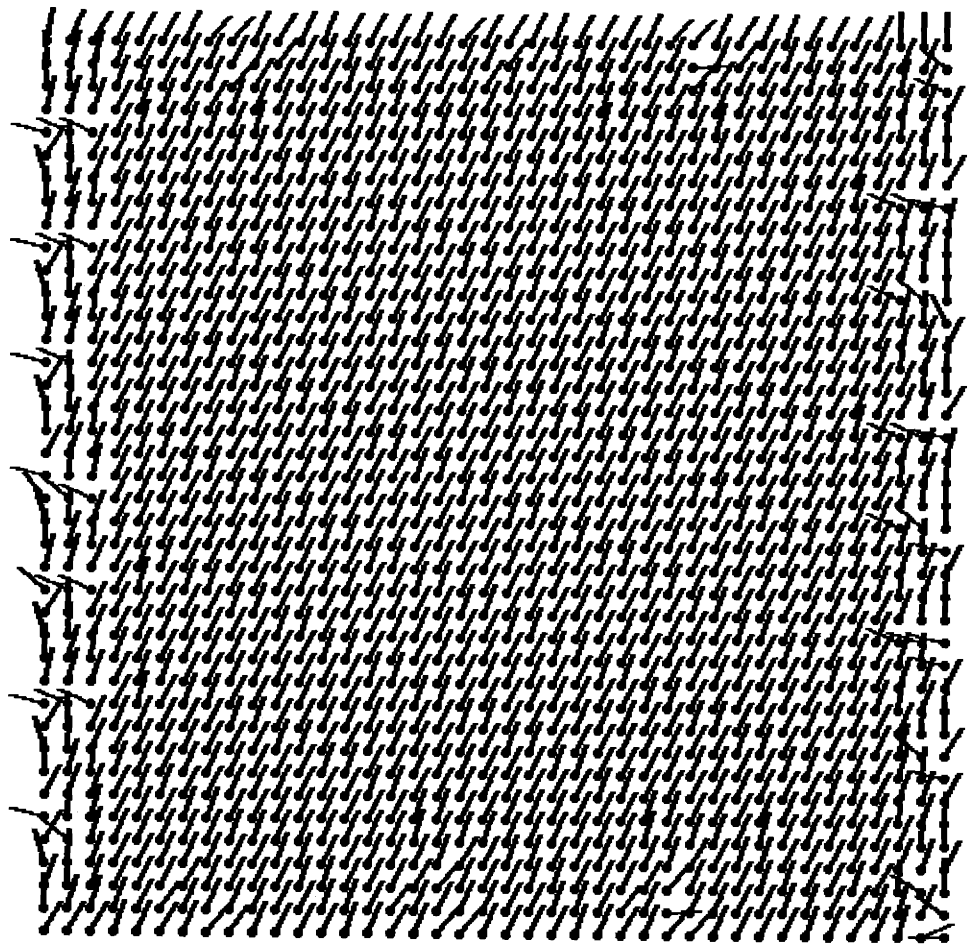

Figure 4.21: RI algorithm using the ρ-model with parameters $\rho = .25$, $\sigma_n{}^2 = .01$

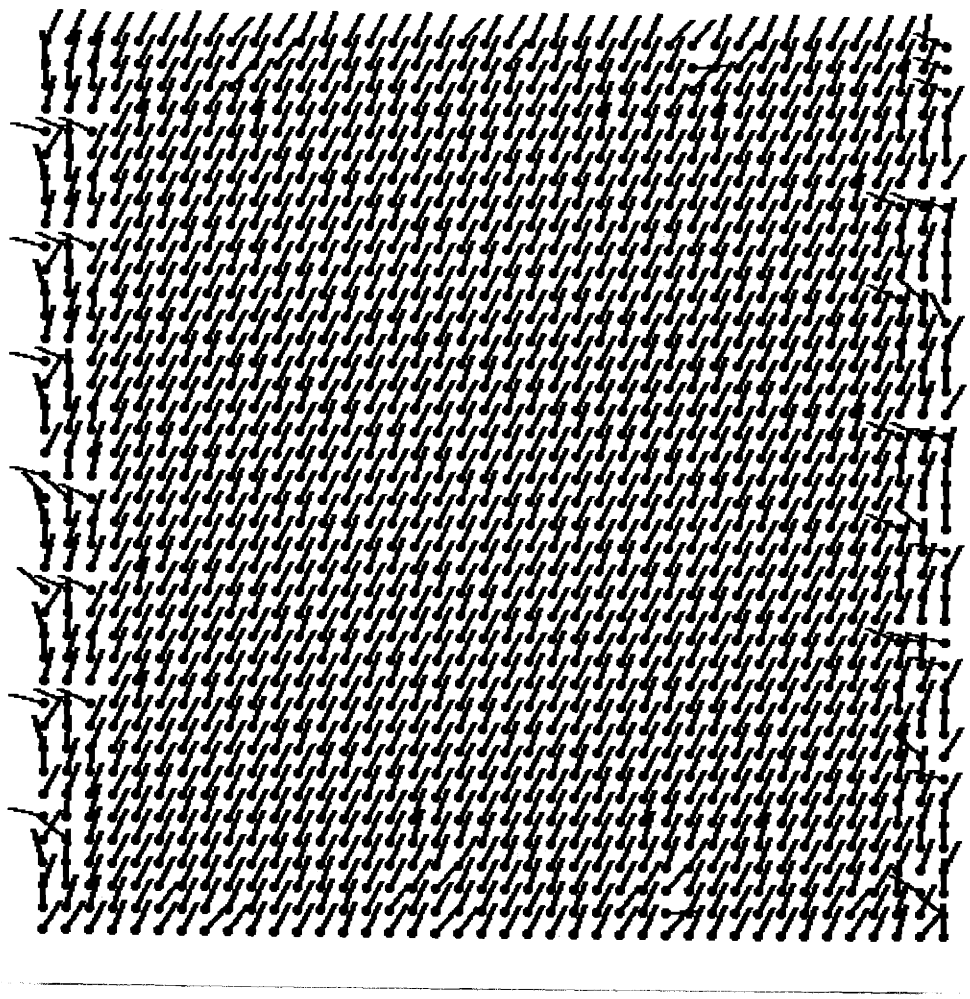

Figure 4.22: RI algorithm using the ρ-model with parameters $\rho = .90$, $\sigma_n{}^2 = .01$

series expansion for Λ^{-1} in (4.43). As can be seen the ϵ^{-1} (where $\epsilon = {\sigma_n}^2$) term in the power series expansion is independent of the value for ρ. For small values of ϵ this term dominates. Consequently, we would not expect ρ to affect the estimates for small ϵ.

The second three examples are illustrated in Figures 4.23-4.25. These three figures all show dip estimates using the RI algorithm based on the ρ model with a large value for ${\sigma_n}^2$. The value of ρ is .01, .25, and .90, respectively, while ${\sigma_n}^2 = 1$.

The estimates in Figures 4.23-4.25 still display spurious dips near the boundaries due to the difficulty with interpolation there. Furthermore, in all three figures there is a coherent degradation of the dip along lines which coincide with bed boundaries in the data (Figure 4.19). This degradation is due to the increased strength of the noise parameter ${\sigma_n}^2$. Careful examination of the dip estimates reveals that the coherent degradation occurs at bed boundaries of low contrast. The degradation at low contrast bed boundaries decreases as the value of ρ increases.

To explain these observations we refer to Figure 4.26. Figure 4.26 shows a 7×7 window of data where the center of the window is near a bed boundary. There are three different types of locations the center of the 7×7 window can occupy near the bed boundary. These three locations are illustrated in Figure 4.26(a-c). As can be seen, the window in Figure 4.26a contains elements from only two beds. The windows in Figures 4.26b and 4.26c have one and two elements from a third bed, respectively.

Figure 4.23 shows the dip estimate with $\rho = .01$ and ${\sigma_n}^2 = 1$. The figure also highlights three individual dips with the labels a, b, and c. These labels correspond to the window configurations in Figure 4.26a, b, and c, respectively. The dip computed using the window in Figure 4.26a is a good estimate. This is due to the fact that only two beds are within the boundaries of the window. The dips computed using windows in Figures 4.26b and c, however, are highly biased. The bias in these estimates is due to the high value for the noise and the low contrast of the two major beds inside the window. The elements from the third bed are more heavily weighted since they contrast more highly than the two major beds do. Since there are only one or two elements from the third bed the RI algorithm

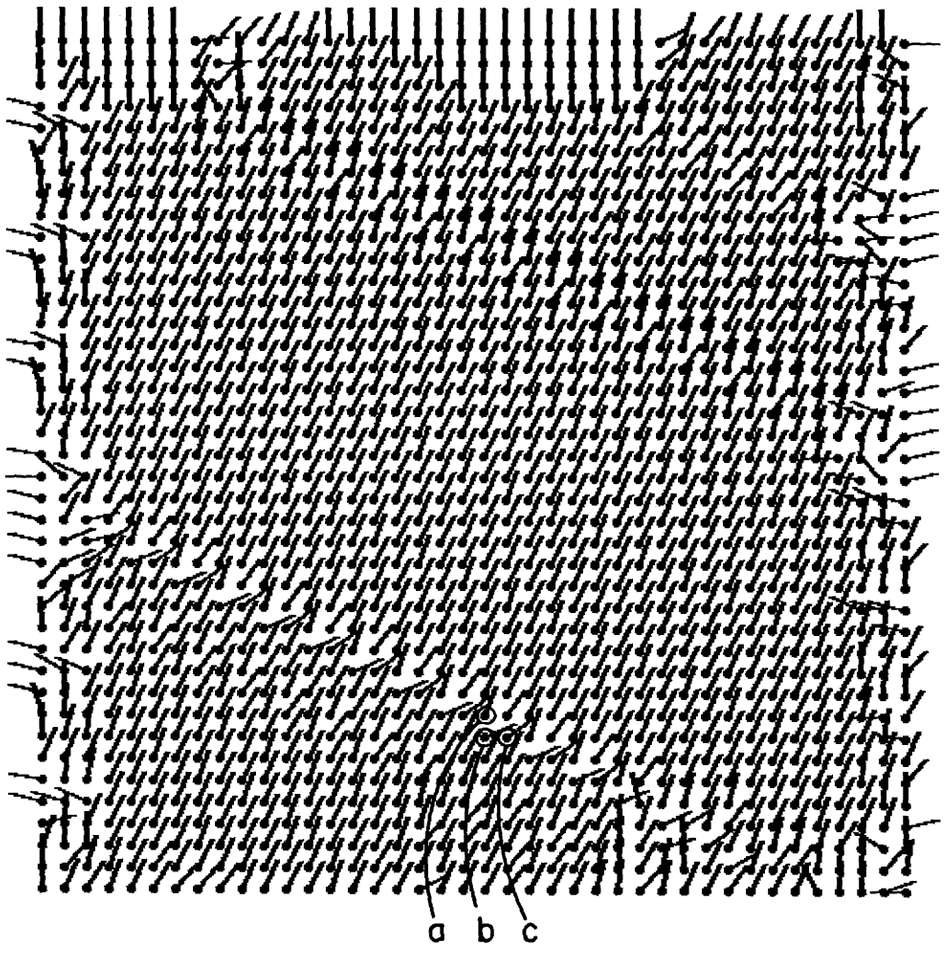

Figure 4.23: RI algorithm using the ρ-model with parameters $\rho = .01$, $\sigma_n{}^2 = 1.0$

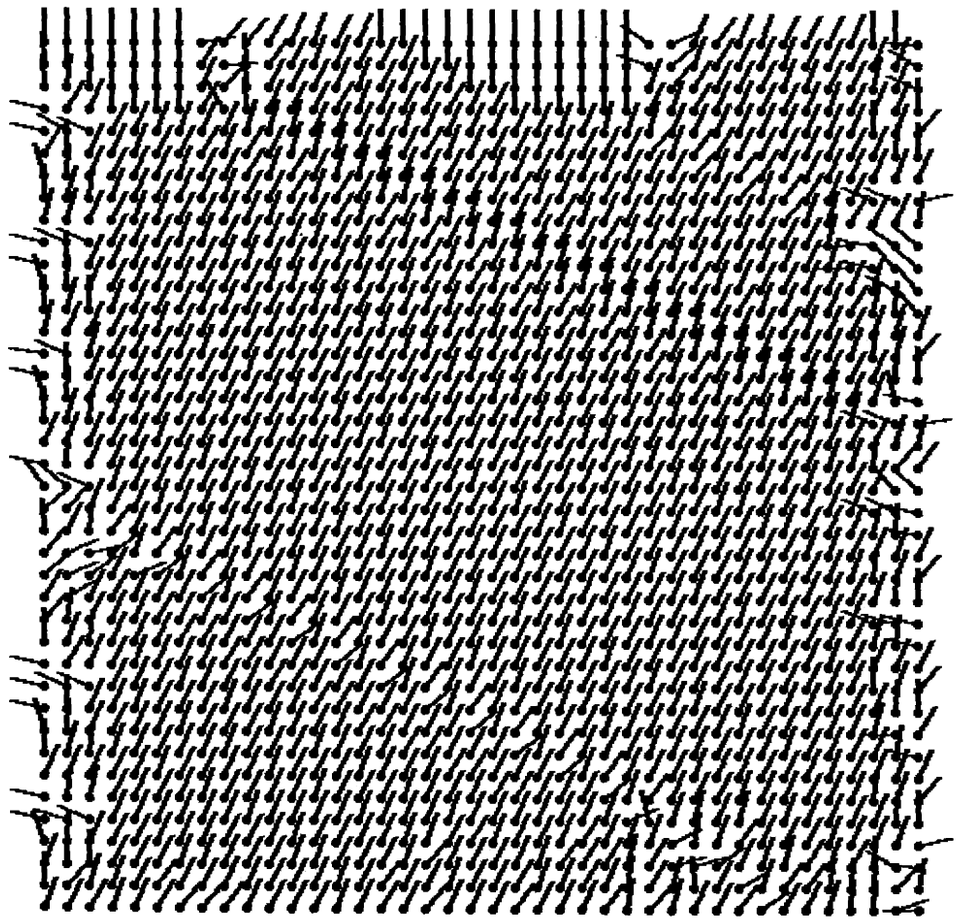

Figure 4.24: RI algorithm using the ρ-model with parameters $\rho = .25$, ${\sigma_n}^2 = 1.0$

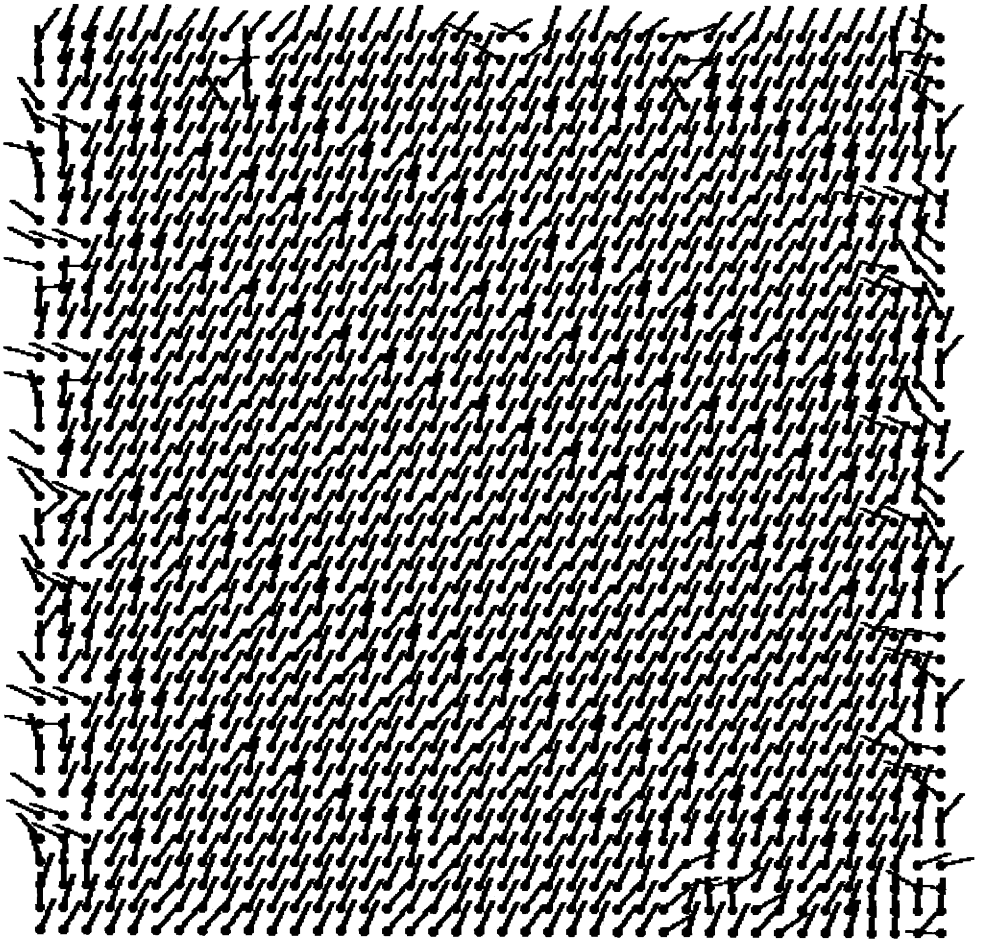

Figure 4.25: RI algorithm using the ρ-model with parameters $\rho = .90$, $\sigma_n{}^2 = 1.0$

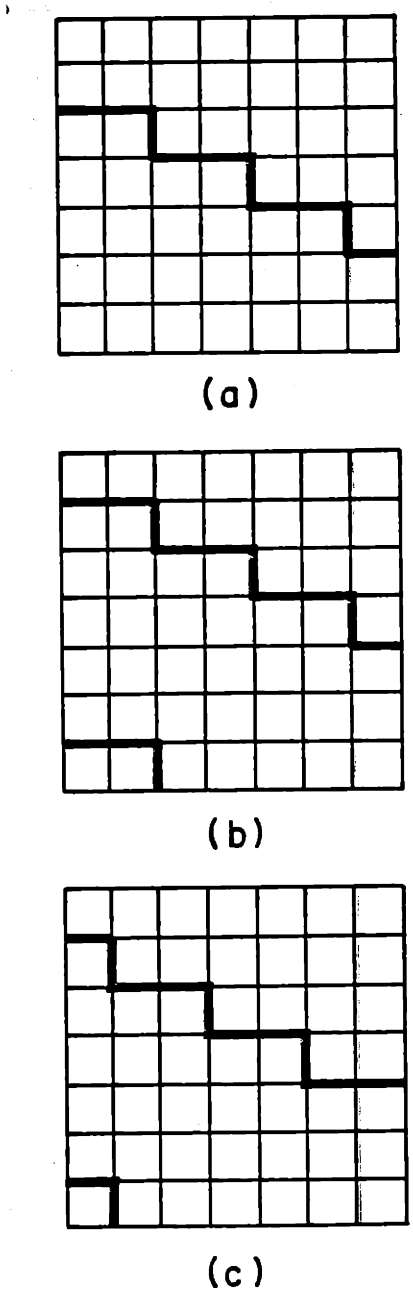

Figure 4.26: Three window locations

cannot accurately identify the correct dip.

The estimates in Figures 4.24 and 4.25 increasingly improve as the value for ρ increases. This is because the RI algorithm performs more and more smoothing in the direction of the dip as the value of ρ approaches unity. Consequently, the one or two additional elements from the third bed in Figure 4.27b and c have less effect on the resulting estimates.

We draw the following conclusions from these examples. When noise is small the value of ρ is of little consequence. However, as the noise increases, increasing the value of ρ improves the performance of the RI algorithm. This improvement is due the extra smoothing which is provided by the higher order terms in the expansion of the inverse covariance matrix in (4.43).

4.5.2 Examples Using the RI Algorithm with the $\rho\gamma$-Model

This section illustrates the relationship between ρ and γ when the RI algorithm is used in conjunction with the $\rho\gamma$-model in (4.12). There are three examples in this section. These examples show the effects of the RI algorithm when ρ is small and γ is large. The examples also illustrate the estimates of dip when ρ and γ are close in value.

In all three examples presented here the parameter $\sigma_n{}^2 = .1$. The examples are illustrated in Figures 4.27- 4.29. In these figures ρ and γ take the values $\rho = .01$ $\gamma = .99$, $\rho = .2$ $\gamma = .8$, and $\rho = .4$ $\gamma = .6$, respectively.

In the three figures it can be seen that as ρ and γ become closer in value, the estimates become more disturbed. In particular, the data regions of low bed contrast show dip estimates which are highly biased. This phenomenon is not surprising since ρ and γ specify the anisotropic character of the data model. As ρ and γ become more similar in value, the RI algorithm is less capable of distinguishing the major correlation axes.

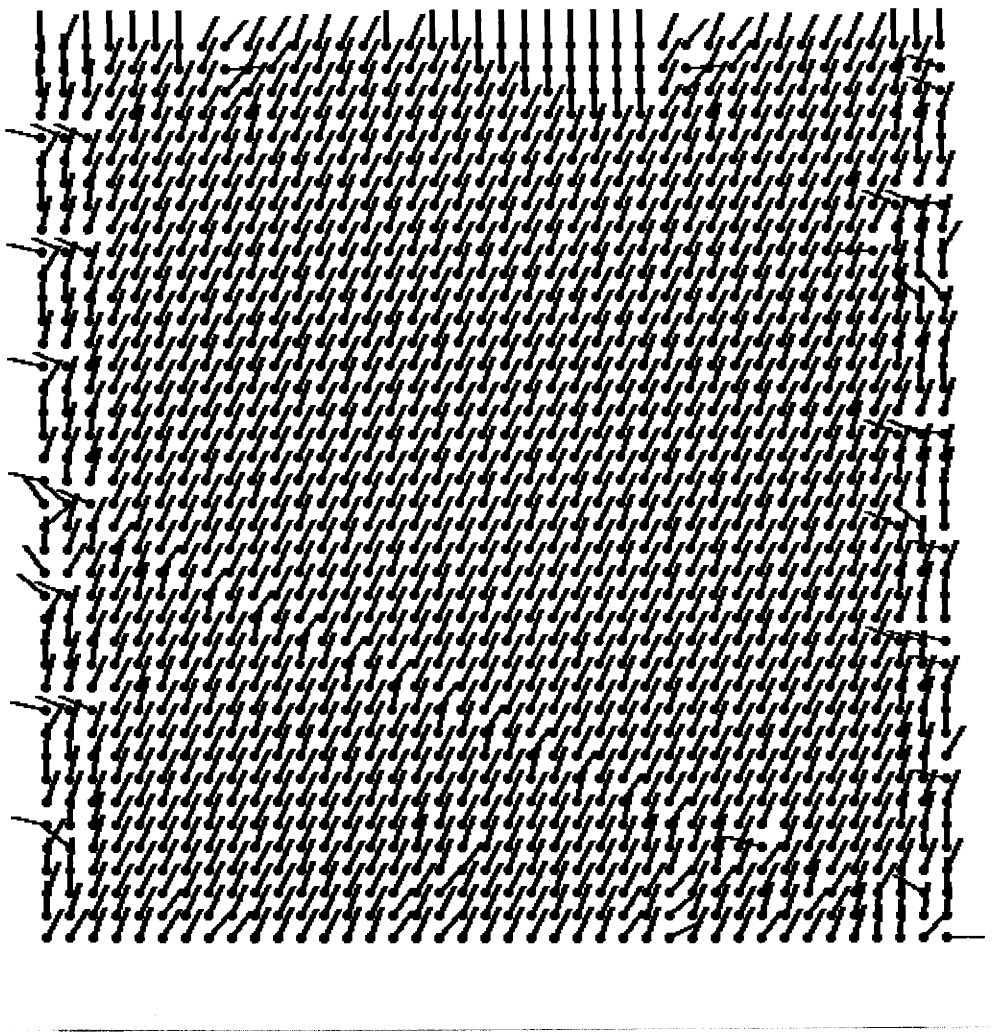

Figure 4.27: RI algorithm using the $\rho\gamma$-model with parameters $\rho = .01$, $\gamma = .99$, ${\sigma_n}^2 = .1$

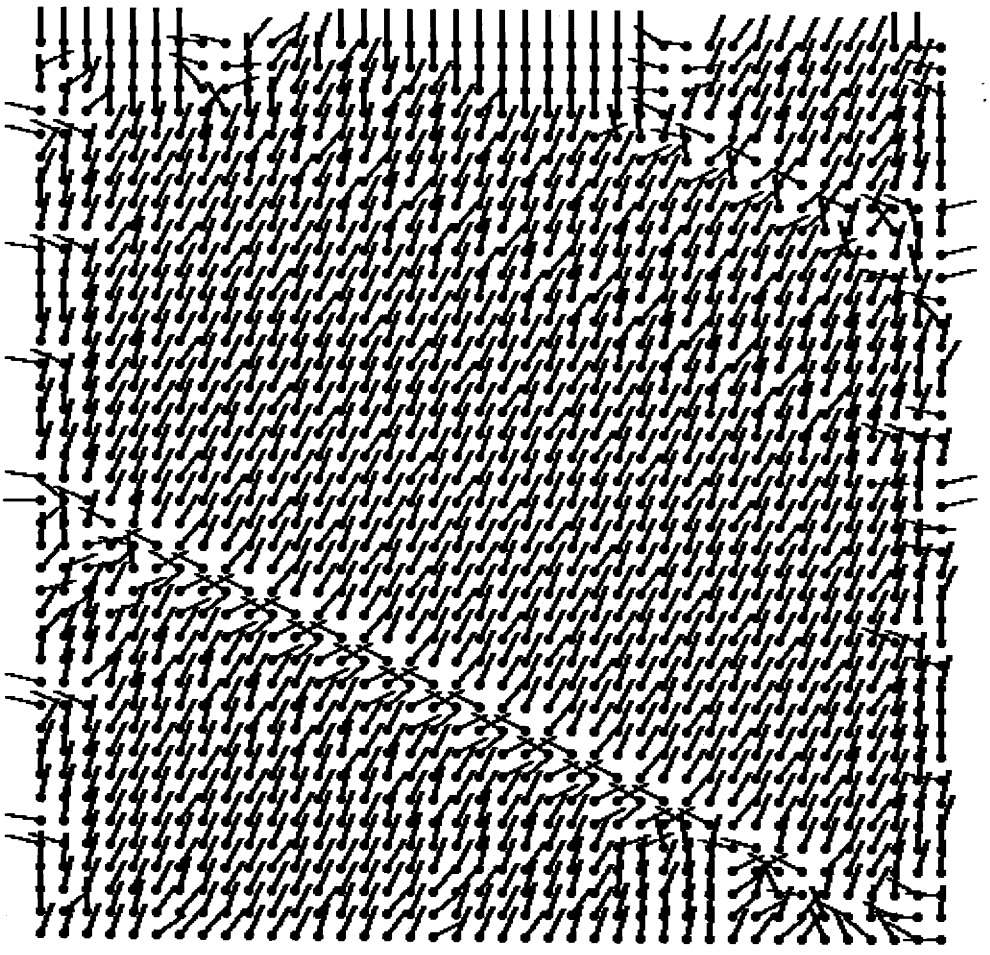

Figure 4.28: RI algorithm using the $\rho\gamma$-model with parameters $\rho = .2$, $\gamma = .8$, ${\sigma_n}^2 = .1$

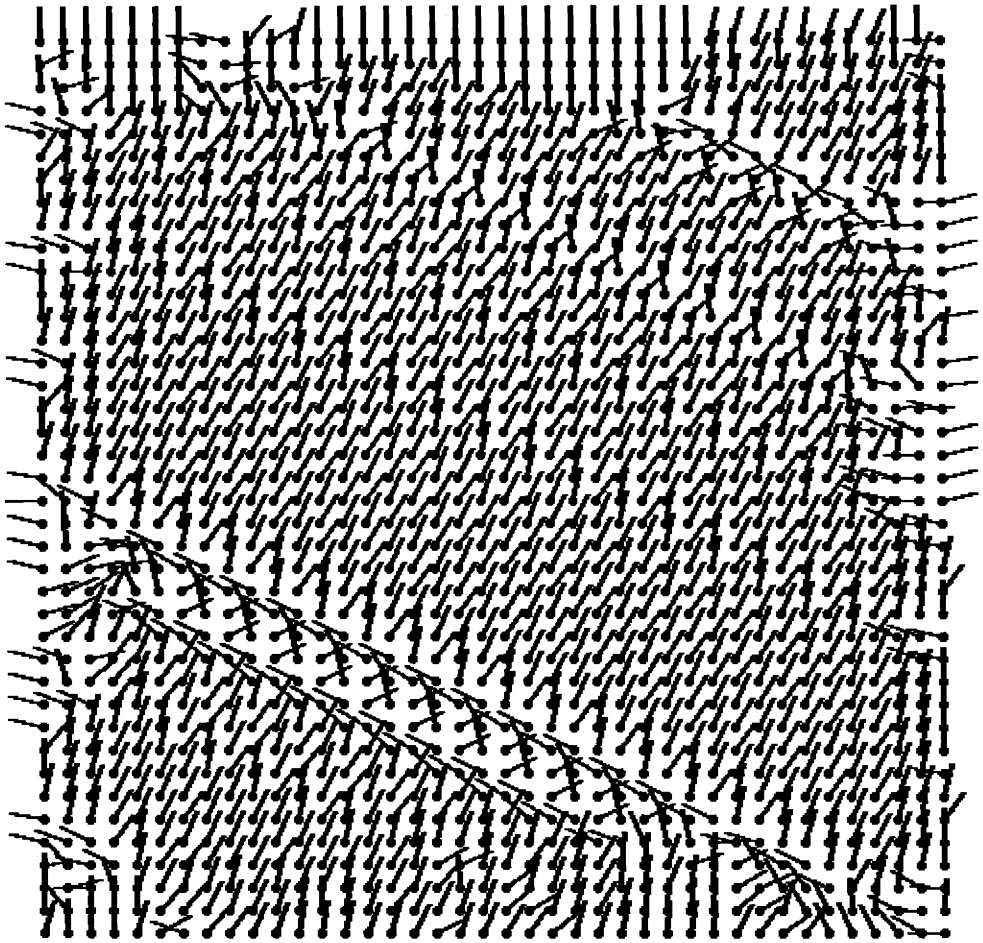

Figure 4.29: RI algorithm using the $\rho\gamma$-model with parameters $\rho = .4$, $\gamma = .6$, ${\sigma_n}^2 = .1$

4.5.3 Example Using the ARI, PROJ, SPROJ, and SA Algorithms

This section contains ten examples. These examples are composed of two estimates for each of the algorithms ARI, PROJ, SPROJ, and SA. The first estimate is based on the ρ-model in (4.11). There are two additional estimates which compare the SPROJ and SA algorithms. The results of this section are compared to those in Sections 4.5.1 and 4.5.2.

The first pair of examples are illustrated in Figures 4.30-4.31. These two figures illustrate the use of the ARI algorithm in estimating the dip field associated with the data in Figure 4.19. Figure 4.30 is obtained using the ρ-model and Figure 4.31 is obtained using the $\rho\gamma$-model.

The estimates still suffer from spurious dips near the boundaries. This is due to the attempt to interpolate data outside the data boundaries. The estimate based on the ρ-model is better than the one based on the $\rho\gamma$-model. As discussed in Section 4.4.5 the $\rho\gamma$-model leads to an algorithm which performs less smoothing in the direction orthogonal to dip than does the ρ model. This accounts for the decreased performance in Figure 4.31.

The illustrations in Figures 4.32-4.33 are dip estimates using the PROJ algorithm based on the ρ and $\rho\gamma$-models, respectively. Notice that since the PROJ algorithm does not use rotation and interpolation, the dip estimates near boundaries are much better than those obtained using the RI or ARI algorithms.[3] However, since the PROJ algorithm uses an approximation to rotation and interpolation it also seems to be more sensitive to problems with low contrast data.

The results of the PROJ algorithm are greatly improved when extra smoothing is used as in the SPROJ algorithm. The results in Figures 4.34-4.35 illustrate the SPROJ algorithm when using the ρ and $\rho\gamma$-models. Both estimates are extremely good.

The illustrations in Figures 4.36-4.37 show the results of using the SA algorithm

[3]Note: The region affected by trying to interpolate data outside of the boundaries of the data is the region consisting of dips within four elements of the boundary. This is due to the use of a 7×7 window.

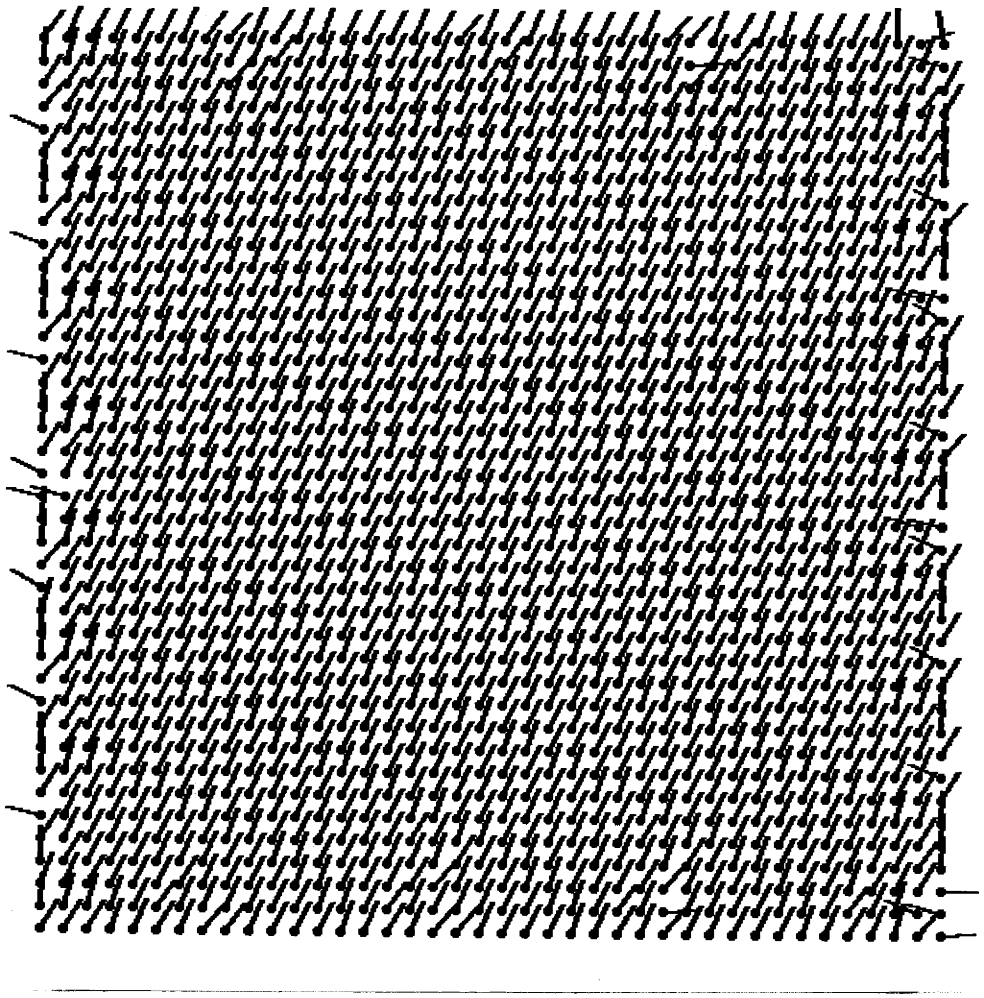

Figure 4.30: ARI algorithm using the ρ-model

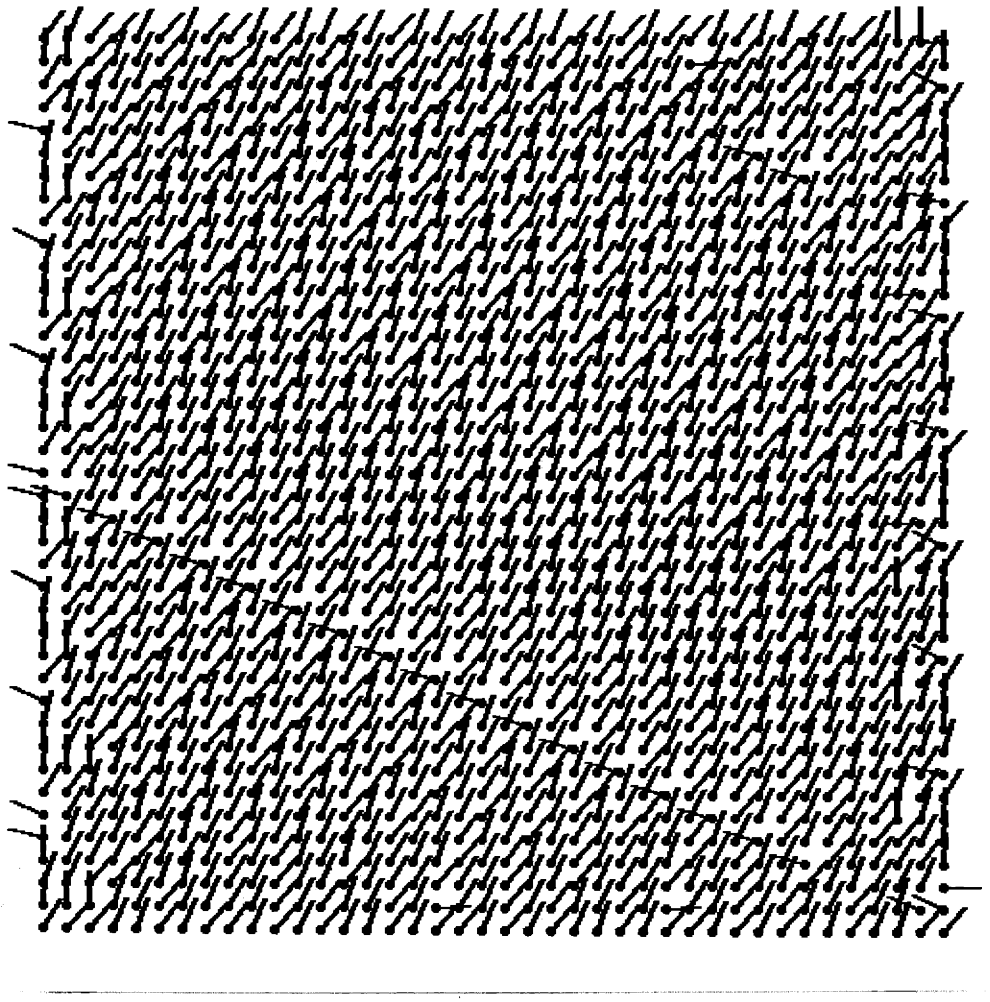

Figure 4.31: ARI algorithm using the $\rho\gamma$-model

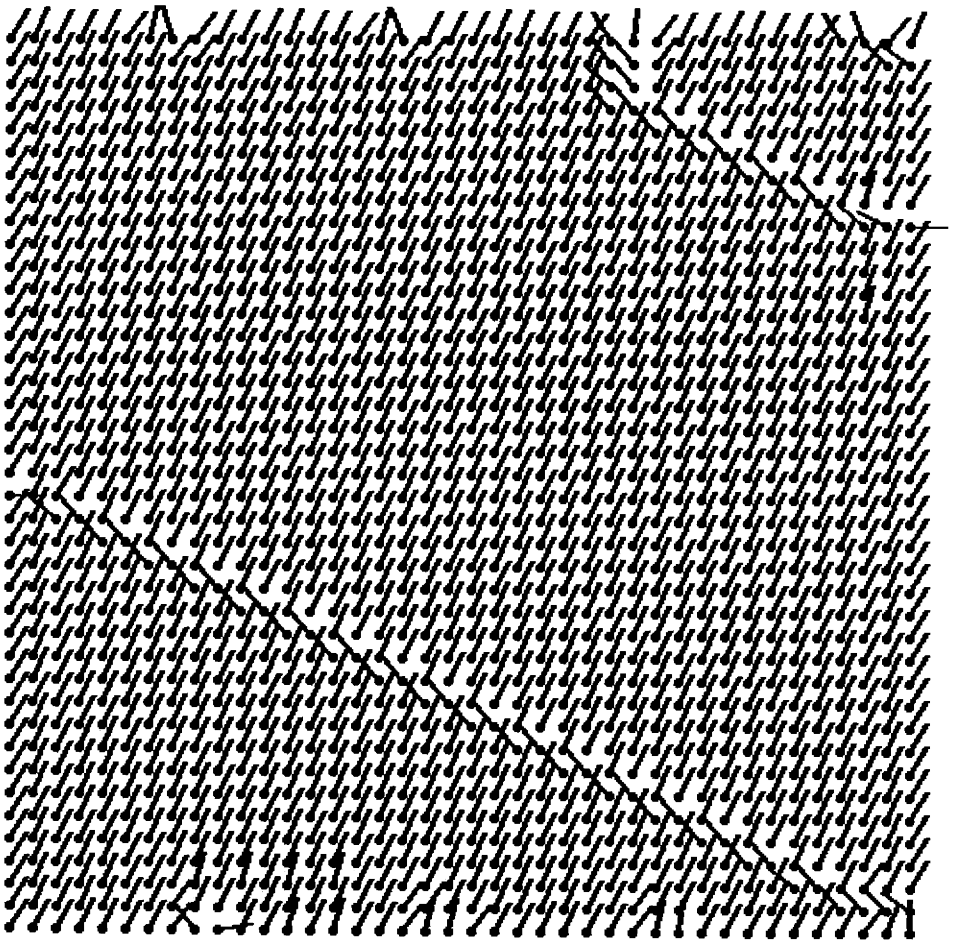

Figure 4.32: PROJ algorithm using the ρ-model

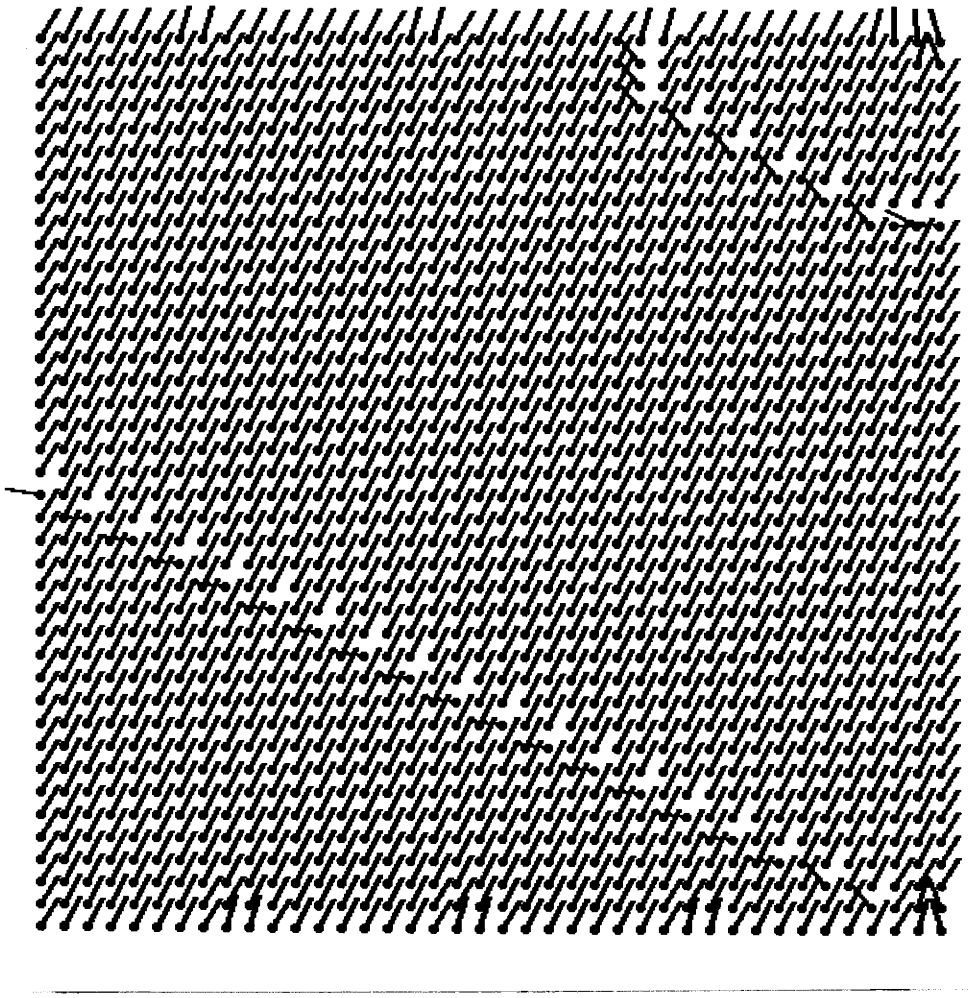

Figure 4.33: PROJ algorithm using the $\rho\gamma$-model

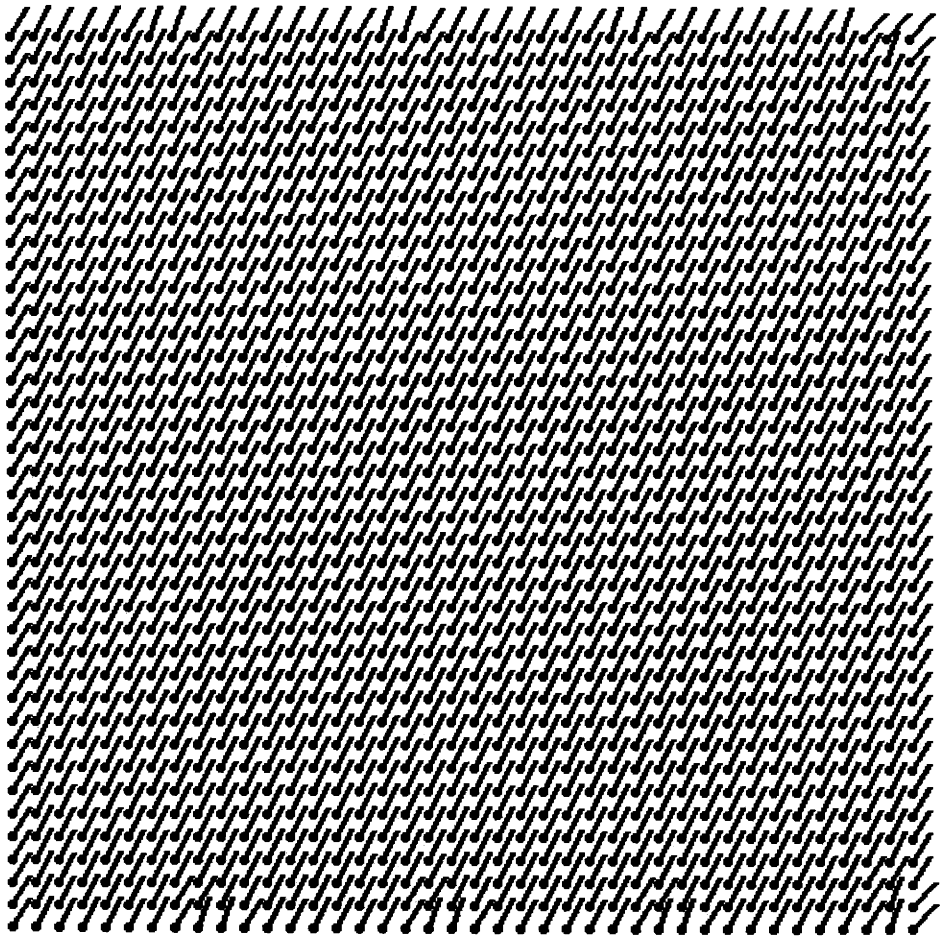

Figure 4.34: SPROJ algorithm using the ρ-model

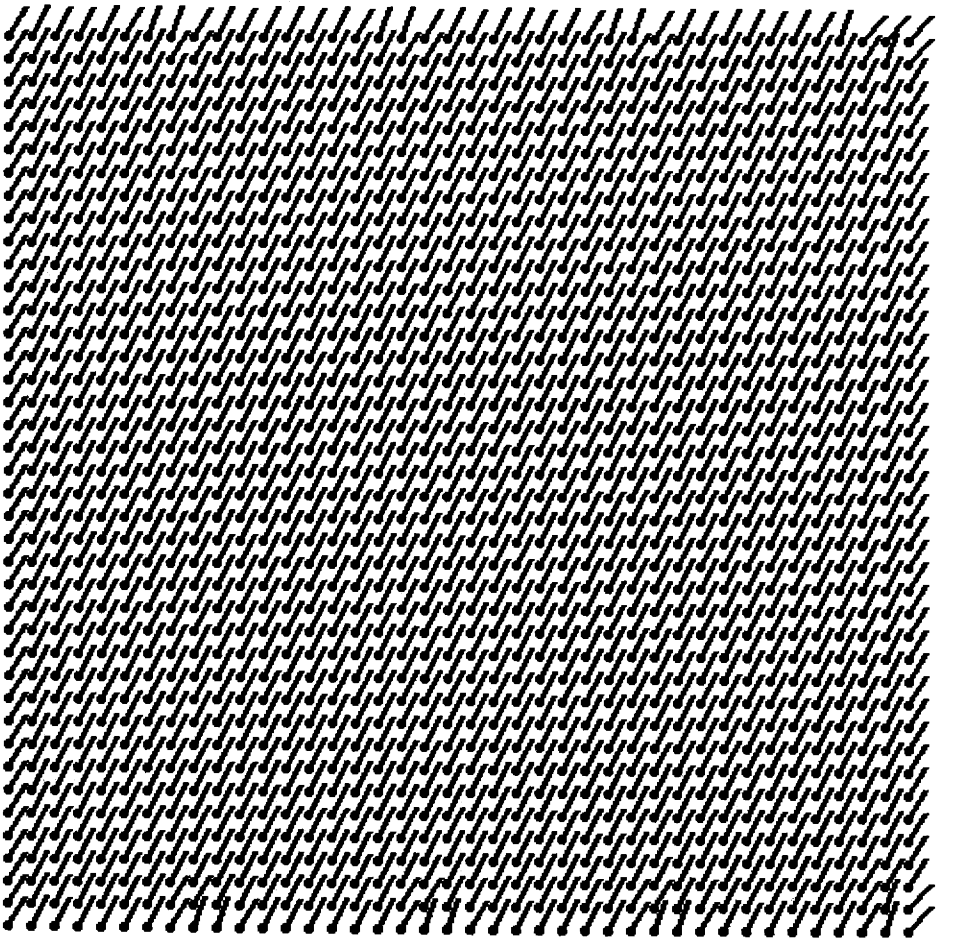

Figure 4.35: SPROJ algorithm using the $\rho\gamma$-model

on the data in Figure 4.19. The SA algorithm incorporates the SPROJ algorithm with a smoothing term. As the figures show, the results are good, however, they are not as good as the results from the SPROJ algorithm. The SA algorithm is a smoothing algorithm which finds an approximate MAP estimate based on the ρ and $\rho\gamma$-models. For data which is noisy one could expect the SA algorithm to perform very nicely and possibly much better than the SPROJ algorithm.

As a test the SA algorithm was re-run on the data where the estimates were initialized to the values obtained from the SPROJ algorithm. The expectation is that the SA algorithm should perform better due to this initialization. We found that indeed the performance was improved as seen by comparing the energy of the Gibb's distributions for the initialized case versus the uninitialized case. The difference in performance, however, is small due to the extremely regular nature of the data.

The final two examples compare the SPROJ and SA algorithms based on the ρ-model when applied to noisy data. Figure 4.38 shows a noisy version of the data in Figure 4.19. The data in Figure 4.19 takes values in $[-1, 1]$. The data in Figure 4.38 takes values in $[-.8, .8]$ and the additive noise takes values in $[-.2, .2]$. The signal to noise ratio could be considered to be 6db under these conditions. It should be noted that several of the layer boundaries that are readily identified in Figure 4.19 are obscured by the additive noise in Figure 4.38.

Figures 4.39 and 4.40 illustrate the dip estimates obtained from the data in Figure 4.38 using the SPROJ and SA algorithms, respectively. Both algorithms are based on the ρ-model. In Figure 4.39 there are several regions where the dip estimates are incoherent with respect to the main character of the data. This is due to the additive noise which obscures several of the bed boundaries in the data. Since the window structure of the SPROJ algorithm is a 7×7 array of elements the noise prevents the algorithm from correctly estimating the dip at the boundaries of low contrast layers.

The dip estimate in Figure 4.40 is obtained using the SA algorithm with the smoothing constant $D = 1.0$. As can be seen in the figure, the regions which were troublesome for the SPROJ algorithm are smoothed over by the SA algorithm. This

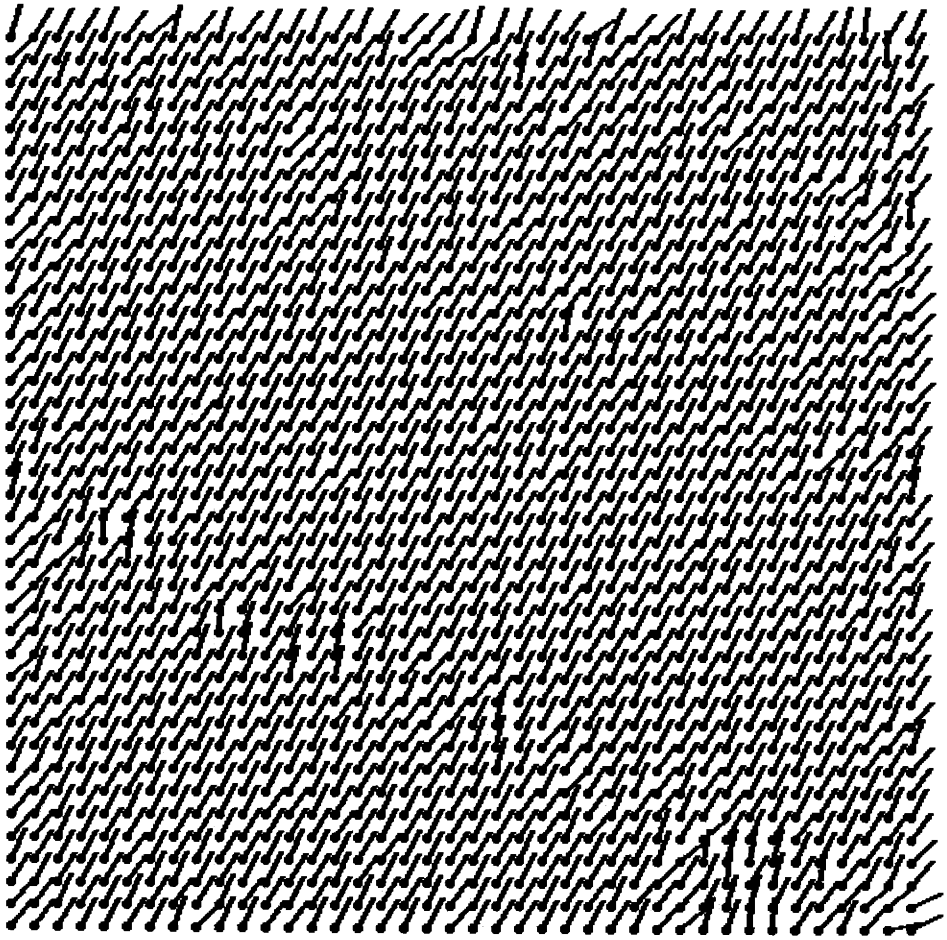

Figure 4.36: SA algorithm using the ρ-model

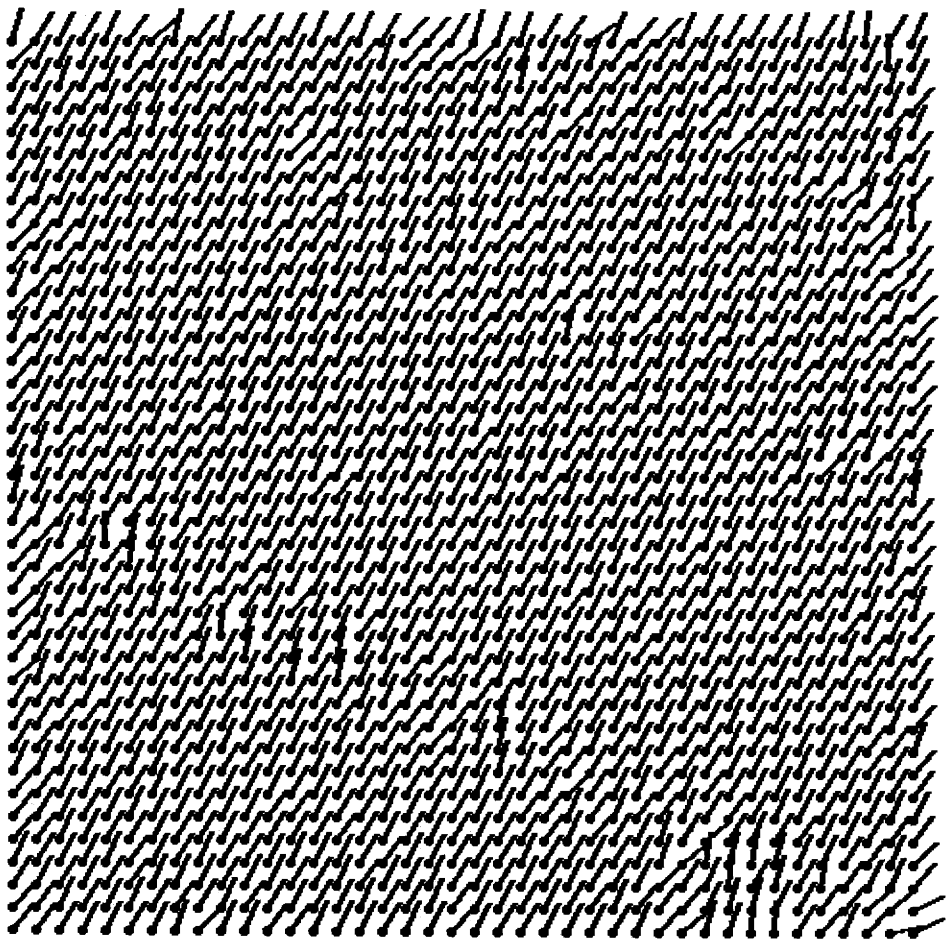

Figure 4.37: SA algorithm using the $\rho\gamma$-model

is a consequence of the smoothing term employed in the SA algorithm.

4.6 Conclusions

This chapter introduce several models for non-binary anisotropic random fields. These models are developed in substantial detail and fast algorithms are derived from these models for the purpose of processing observed data.

To summarize the chapter, Sections 4.1 and 4.2 develop a MRF model for non-binary data. Sections 4.3 and 4.4 are alternate models for the data which, through a sequence of steps, provides a rationale for the MRF models introduced previously. In the process of discussing the models we develop several fast algorithms in Section 4.4. Section 4.5 is an examples section which illustrates many of the features discussed within the chapter.

Chapters 3 and 4 developed models and methods of analyzing data. A possible criticism of the results obtained in these chapters is that analysis of the data results in estimates of the same cardinality as the data. That is, there is no compression of the data by the estimation algorithms.

The feature that our estimation algorithms have concentrated on is dip. Dip is conceptually an aggregate feature of the data. Consequently, it is reasonable to expect one to be able to compress the dimensionality of the dip estimates. Furthermore, it is highly desirable to do so since it simplifies the interpretation of the data.

The next chapter concentrates on estimation of bed boundaries. This is important since dip near bed boundaries can be more complicated when these boundaries are non-planar. Consequently, information about bed boundaries helps in calculating accurate dip estimates near these boundaries. Furthermore, characterizing the bed boundaries also helps in identifying higher level features in the data and, thus, is a step towards compressing the data.

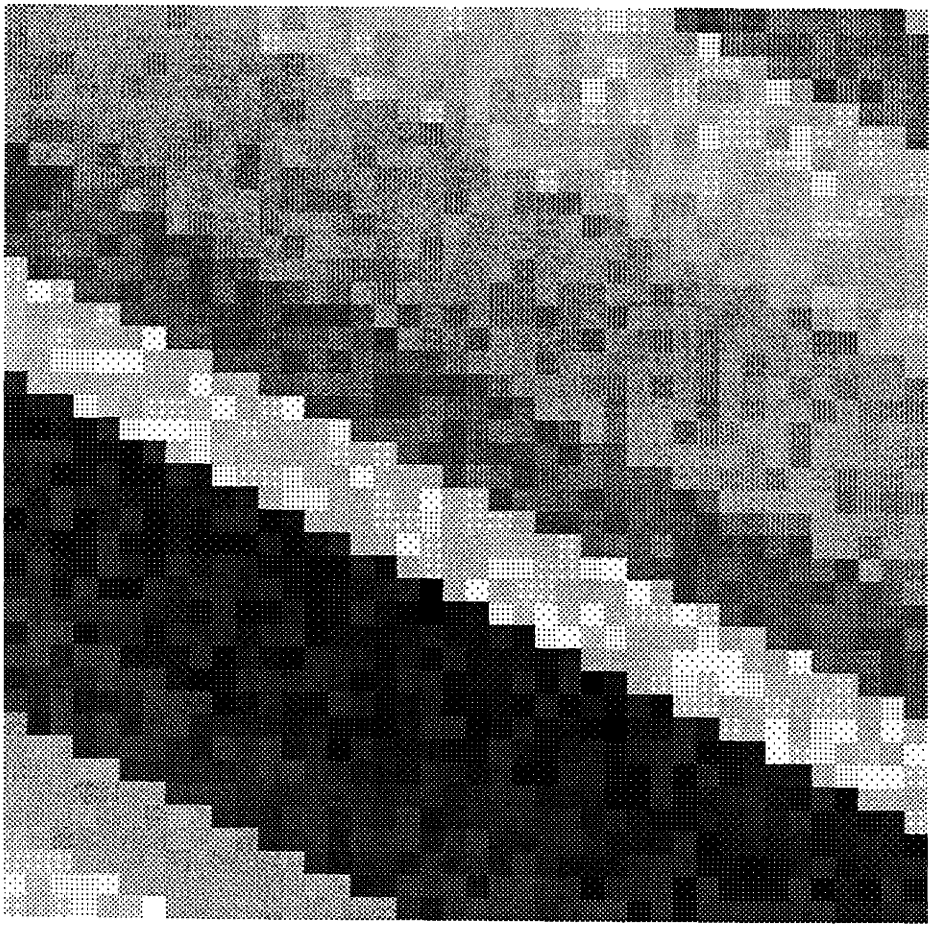

Figure 4.38: Synthetic, non-binary data in additive noise

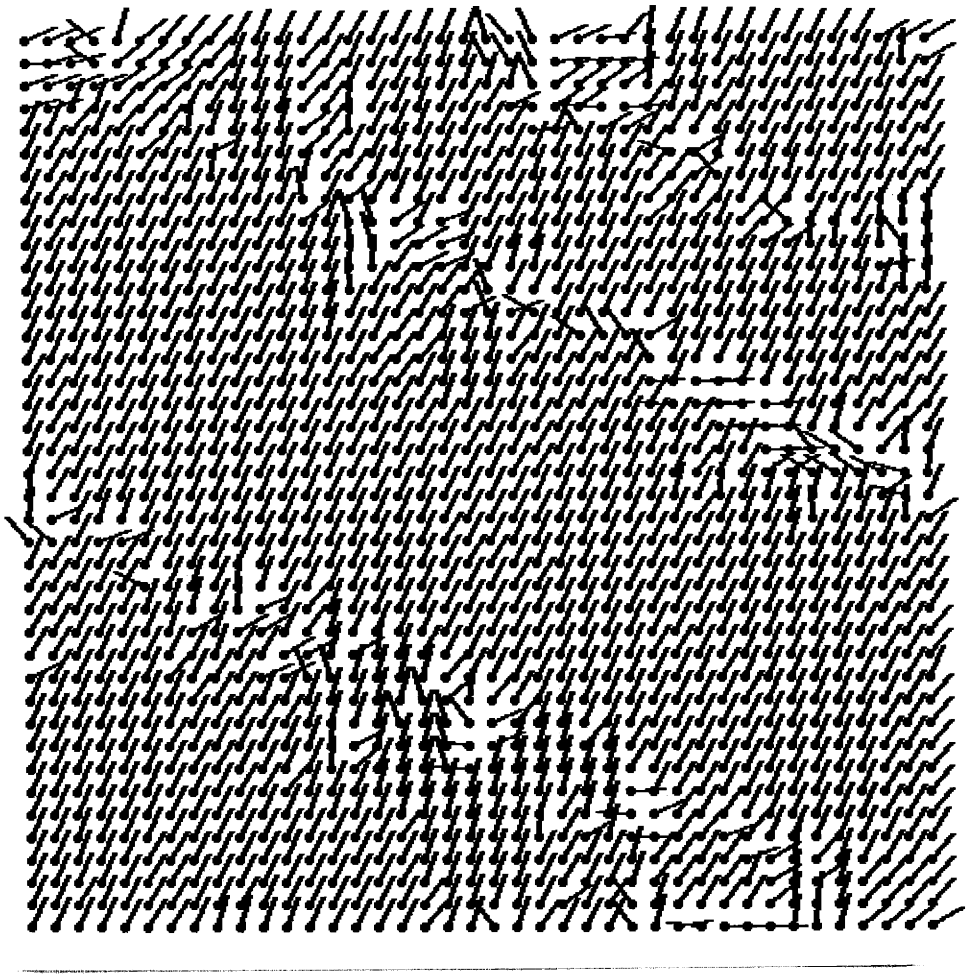

Figure 4.39: SPROJ algorithm using the ρ-model

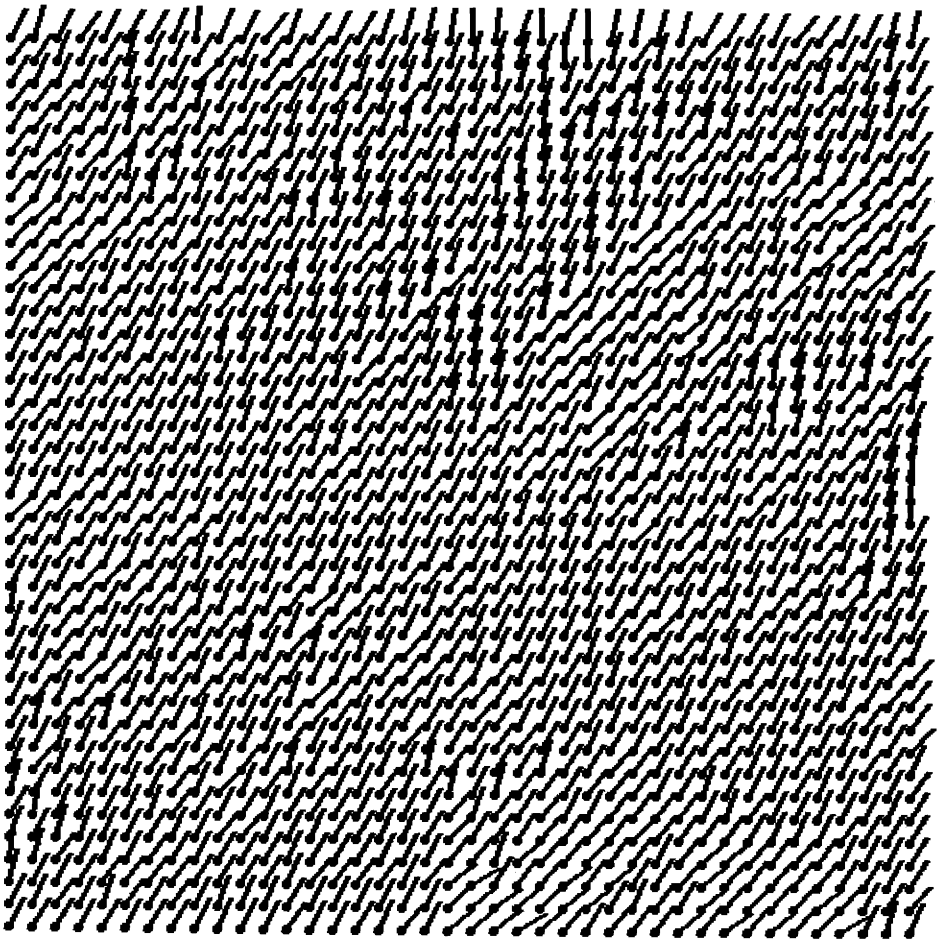

Figure 4.40: SA algorithm using the ρ-model with $D = 1$.

4.7 Appendix: Power Series Expansion of the Inverse Covariance Matrices

First it is shown that (4.42) satisfies (4.41) by making the ϵ^0 term equal to the identity and the remaining terms zero. Note that

$$S_k = \begin{cases} I_{n_1n_2\times n_1n_2} - \frac{1}{n_2}(I_{n_2\times n_2}\otimes U) & k=-1 \\ \frac{(-1)^k}{{n_2}^{k+2}}(R^{-(k+1)}\otimes U) & k=0,1,\cdots \end{cases} \tag{4.73}$$

and that

$$\Lambda_\rho^{-1} = \sum_{k=-1}^{\infty}\epsilon^k S_k \tag{4.74}$$

$$\begin{aligned}\Lambda_\rho &= (\mathcal{R}+\epsilon I_{n_1n_2\times n_1n_2}) \\ &= (R\otimes U)+\epsilon I_{n_1n_2\times n_1n_2}\end{aligned} \tag{4.75}$$

so that

$$\begin{aligned}\Lambda_\rho\Lambda_\rho^{-1} &= (R\otimes U)+\epsilon I_{n_1n_2\times n_1n_2})\sum_{k=-1}^{\infty}\epsilon^k S_k \\ &= \frac{1}{\epsilon}(R\otimes U)S_{-1}+(S_{-1}+(R\otimes U)S_0)+\sum_{k=0}^{\infty}\epsilon^{k+1}(S_k+(R\otimes U)S_{k+1})\end{aligned} \tag{4.76}$$

The $\frac{1}{\epsilon}$ term in (4.76) can be expanded by substituting for S_{-1}

$$\begin{aligned}(R\otimes U)S_{-1} &= (R\otimes U)(I_{n_1n_2\times n_1n_2}-\frac{1}{n_2}(I_{n_2\times n_2}\otimes U)) \\ &= (R\otimes U)-\frac{1}{n_2}(R\otimes U)(I_{n_2\times n_2}\otimes U)) \\ &= (R\otimes U)-\frac{1}{n_2}(R\otimes U^2)\end{aligned} \tag{4.77}$$

since $U^2=n_2U$ we have that (4.77) is zero as is desired.

The ϵ^0 term in (4.76) is now expanded

$$\begin{aligned}S_{-1}+(R\otimes U)S_0 &= (I_{n_1n_2\times n_1n_2}-\frac{1}{n_2}(I_{n_2\times n_2}\otimes U))+(R\otimes U)(\frac{1}{{n_2}^2}R^{-1}\otimes U) \\ &= (I_{n_1n_2\times n_1n_2}-\frac{1}{n_2}(I_{n_2\times n_2}\otimes U))+\frac{1}{{n_2}^2}(I_{n_2\times n_2}\otimes U^2)\end{aligned} \tag{4.78}$$

which by the same property of U makes (4.78) equal to $I_{n_1n_2\times n_1n_2}$. Finally, each of the remaining terms in (4.76) expand into

$$\begin{aligned} S_k + (R\otimes U)S_{k+1} &= \frac{-1}{{n_2}^{k+2}}(R^{-(k+1)}\otimes U) + (R\otimes U)\frac{(-1)^{k+1}}{{n_2}^{k+3}}(R^{-(k+2)}\otimes U) \\ &= \frac{(-1)^k}{{n_2}^{k+2}}[(R^{-(k+1)}\otimes U) - \frac{1}{n_2}(R\otimes U)(R^{-(k+2)}\otimes U)] \\ &= \frac{(-1)^k}{{n_2}^{k+2}}[(R^{-(k+1)}\otimes U) - \frac{1}{n_2}(R^{-(k+1)}\otimes U^2)] \\ &= \underline{0} \end{aligned} \tag{4.79}$$

which completes the argument.

It is now shown that (4.53) satisfies (4.52) by making the ϵ^0 term the identity and all other terms zero. Note that

$$S_k = \begin{cases} \mathcal{R}^{-1} & k=0 \\ (-1)^k\mathcal{R}^{-(k+1)} & k=1,2,\cdots \end{cases} \tag{4.80}$$

where $\mathcal{R} = R\otimes G + \epsilon I_{n_1n_2\times n_1n_2}$. Furthermore,

$$\Lambda_{\rho\gamma}^{-1} = \sum_{k=0}^{\infty}\epsilon^k(-1)^k\mathcal{R}^{-(k+1)} \tag{4.81}$$

$$\Lambda_{\rho\gamma} = \mathcal{R} + \epsilon I \tag{4.82}$$

so that,

$$\begin{aligned} \Lambda_{\rho\gamma}\Lambda_{\rho\gamma}^{-1} &= (\mathcal{R}+\epsilon I)(\sum_{k=0}^{\infty}\epsilon^k(-1)^k\mathcal{R}^{-(k+1)}) \\ &= \mathcal{R}\mathcal{R}^{-1} + \sum_{k=1}^{\infty}\epsilon^k[(-1)^k\mathcal{R}\mathcal{R}^{-(k+1)} + (-1)^{(k-1)}\mathcal{R}^{-k}] \\ &= I \end{aligned} \tag{4.83}$$

since, $\mathcal{R}\mathcal{R}^{-1} = I$ and all the terms in the sum are zero.

Chapter 5

Edge Models for Layered Data

Chapters 3 and 4 concentrate on modeling and estimation of low level features such as dip and bed frequency. The objective of this chapter is to present methods for the identification of bed boundary locations. Information about bed boundaries is useful for calculating dip near these boundaries. Furthermore, bed boundaries are important for characterizing higher level features of the data.

This chapter is organized into the following sections. Section 5.1 presents a local edge model which depends on the knowledge of dip in the region. Section 5.2 details an edge estimation algorithm using the model of Section 5.1. Section 5.3 illustrates some examples of the local edge estimation algorithm using the SPROJ dip estimates discussed in the last chapter. Section 5.4 presents a clustering algorithm for combining local edge estimates into global straight line estimates. Section 5.5 illustrates some examples of the global line estimation algorithm. Section 5.6 introduces another global line estimation procedure based on Kalman filtering. Section 5.7 illustrates examples of the Kalman filter algorithm. Section 5.8 illustrates examples combining the two global line estimation techniques and some other post-processing techniques. Section 5.9 summarizes the conclusions of this chapter.

5.1 A Local Edge Model

This sub-section describes a local edge model. This model is used in a manner similar to the way in which the local constant dip model of Section 4.3 is applied

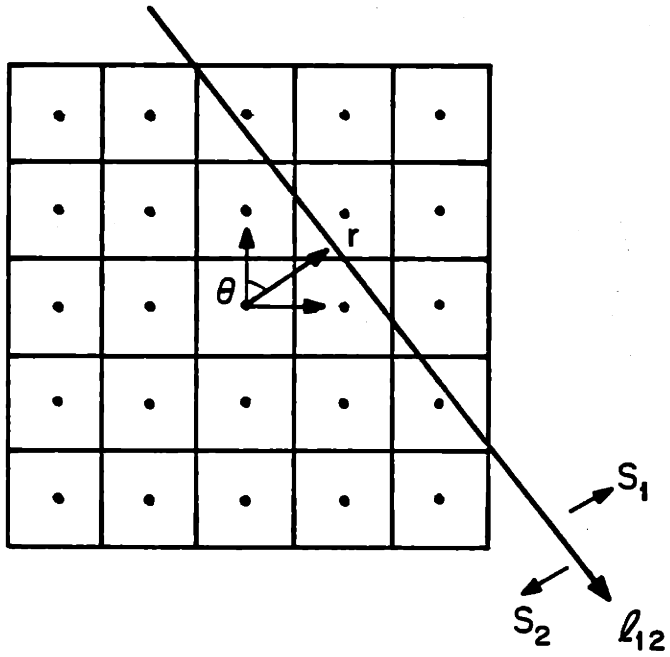

Figure 5.1: Data Separated By Line l_{12}

to large data sets. That is, the local edge model is proposed for an $n_1 \times n_2$ array of data. For data sets much larger than $n_1 \times n_2$ the local edge model is applied by moving an $n_1 \times n_2$ window over the data, calculating local edges as it is moved along. The collection of local edges is the output of our local edge estimation algorithm.

As stated above, it is assumed that the data associated with our local edge model consists of an $n_1 \times n_2$ array of elements, $\underline{s}$. Referring to Figure 5.1 it is assumed that the data array can be divided into two regions S_1 and S_2 separated by a straight line whose location is to be estimated. The line separating the two subsets is parameterized in terms of its distance, r, from the center of the array. The ray perpendicular to the line and through the array center makes an angle θ with the y-axis. This angle corresponds to the local dip within the data window.

The line in Figure 5.1 represents our edge. The edge enters into a mathematical model for the data through specification of the distribution for $\underline{s}$. The data, $\underline{s}$, is

assumed to be a zero-mean JGRV. The covariance matrix for the data is denoted by Π, and the elements of Π are specified by

$$[\Pi]_{ij} = \begin{cases} 1+\sigma^2\delta_{ij} & \{i,j\} \subset S_1 \text{ or } \{i,j\} \subset S_2 \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The covariance model in (5.1) indicates that if s_i and s_j are on opposite sides of line l_{12} then they are uncorrelated. Furthermore, if elements s_i and s_j are on the same side of line l_{12} then they can be viewed as being observations of a single unit-variance random variable in the presence of zero-mean, white Gaussian noise of variance σ^2. Consequently, the variance of s_i is $1+\sigma^2$.

If the elements of the data vector, $\underline{s}$, are ordered so that the elements belonging to set S_1 are in the upper portion of vector and the elements belonging to set S_2 are in the lower portion of the vector then

$$\Pi = \begin{bmatrix} 1_{|S_1|} & \underline{0} \\ \underline{0} & 1_{|S_2|} \end{bmatrix} + \sigma^2 I \tag{5.2}$$

where $|S_i|$ denotes the number of elements in the set S_i, $i = 1, 2$, $1_{|S_i|}$ is an $|S_i| \times |S_i|$ matrix whose elements are all unity and I is the $(n_1 n_2) \times (n_1 n_2)$ identity matrix.

The model just described implicitly depends upon the location of line l_{12}. The parameters r and θ define the location of this line. Consequently, the model can be used to estimate the line's position by estimation of the parameters r and θ. However, if some prior knowledge of either of the parameters is available this can be incorporated into the estimation procedure. Since we are concerned with layered data structures and dip estimation, a logical approach is to use the dip estimate provided by one of the algorithms in Chapter 4 to specify the value of θ. This is the approach taken in the next section and throughout this chapter.

5.2 ML Estimation of Edges

The previous section discussed a model for an edge in layered data. The purpose of this section is to develop the ML estimate of the edge location. It is assumed

that the edge orientation (the dip) is already known and the estimation procedure concentrates on the unknown value for r (see Figure 5.1).

The ML estimate of r is obtained by solving

$$\hat{r}_{ML} = \arg\max_{r}\{\log p[\underline{s}|r,\theta]\} \tag{5.3}$$

where θ is known. Substituting the distribution for $\underline{s}$ into (5.3) we obtain

$$\hat{r}_{ML} = \arg\max_{r}\{-\log|\Pi| - \underline{s}^T\Pi^{-1}\underline{s}\} \tag{5.4}$$

where the matrix Π is as in (5.2).

Solving (5.4) requires the determinant and inverse of Π. Fortunately, Π is a simple matrix for which the determinant and inverse are easily computed. The matrix Π is composed of two sub-matrices which have the form

$$X_{|S_i|} = 1_{|S_i|} + \sigma^2 I_{|S_i|}, \qquad \text{i=1,2} \tag{5.5}$$

where $1_{|S_i|}$ is the $|S_i| \times |S_i|$ unity matrix and $I_{|S_i|}$ is the $|S_i| \times |S_i|$ identity matrix. The determinant of $X_{|S_i|}$ is

$$\det(X_{|S_i|}) = (\sigma^2)^{|S_i|-1}[\sigma^2 + |S_i|], \qquad \text{i=1,2} \tag{5.6}$$

The inverse of $X_{|S_i|}$ is

$$X^{-1}_{|S_i|} = \tfrac{1}{\sigma^2(\sigma^2+|S_i|)}[-1_{|S_i|} + (\sigma^2 + |S_i|)I_{|S_i|}], \qquad \text{i=1,2} \tag{5.7}$$

The result in (5.7) is easily verified by multiplication with (5.5). The result in (5.6) is shown in the appendix to this chapter.

Using the results of (5.6) and (5.7) the determinant of Π is

$$\begin{aligned}\det(\Pi) &= \det(X_{|S_1|})\det(X_{|S_2|}) \\ &= (\sigma^2)^{|S_1|+|S_2|-2}[\sigma^2 + |S_1|][\sigma^2 + |S_2|] \\ &= (\sigma^2)^{n_1n_2-2}[(\sigma^2)^2 + n_1n_2\sigma^2 + |S_1||S_2|]\end{aligned} \tag{5.8}$$

where $(|S_1| + |S_2|)$ has been replaced by n_1n_2. The inverse of Π is

$$\Pi^{-1} = \begin{bmatrix} X^{-1}_{|S_1|} & \underline{0} \\ \underline{0} & X^{-1}_{|S_2|} \end{bmatrix} \tag{5.9}$$

The quadratic product of the data, $\underline{s}$, with the inverse of Π yields a simple result.

$$\begin{aligned}
\underline{s}^T\Pi^{-1}\underline{s} &= \underline{s}^T_{|S_1|}X^{-1}_{|S_1|}\underline{s}_{|S_1|} + \underline{s}^T_{|S_2|}X^{-1}_{|S_2|}\underline{s}_{|S_2|} \\
&= \sum_{i=1}^{2}\underline{s}^T_{|S_i|}\{\frac{1}{\sigma^2(\sigma^2+|S_i|)}[-1_{|S_i|}+(\sigma^2+|S_i|)I_{|S_i|}]\}\underline{s}_{|S_i|} \\
&= \sum_{i=1}^{2}[\frac{\sigma^2+|S_i|}{\sigma^2(\sigma^2+|S_i|)}\sum_{k\in S_i}s_k{}^2 - \frac{1}{\sigma^2(\sigma^2+|S_i|)}(\sum_{k\in S_i}s_k)^2] \\
&= \sum_{i=1}^{2}\{\frac{1}{\sigma^2}[\sum_{k\in S_i}s_k{}^2 - \frac{1}{(\sigma^2+|S_i|)}(\sum_{k\in S_i}s_k)^2]\} \qquad (5.10)
\end{aligned}$$

When $\sigma^2 \ll |S_i|$ the inner bracket of (5.10) is approximately $|S_i|\hat{\sigma}^2_{S_i}$ where $\hat{\sigma}^2_{S_i}$ is the sample covariance of the data in set S_i.

Substituting (5.8) and (5.10) into (5.4) yields

$$\begin{aligned}
\hat{r}_{ML} = \arg\max_r\{ &- \log[(\sigma^2)^{n_1n_2-2}[(\sigma^2)^2 + n_1n_2\sigma^2 + |S_1||S_2|] \\
&- \frac{1}{\sigma^2}\sum_{i=1}^{2}[\sum_{k\in S_i}s_k{}^2 - \frac{1}{\sigma^2+|S_i|}(\sum_{k\in S_i}s_k)^2]\} \qquad (5.11)
\end{aligned}$$

Discarding constants which are independent of r yields

$$\begin{aligned}
\hat{r}_{ML} = \arg\max_r\{ &- \log[\sigma^2(n_1n_2+\sigma^2) + |S_1||S_2|] \\
&- \frac{1}{\sigma^2}\sum_{i=1}^{2}[\sum_{k\in S_i}s_k{}^2 - \frac{1}{(\sigma^2+|S_i|)}(\sum_{k\in S_i}s_k)^2]\} \\
= \arg\max_r\{ &- \log(\sigma^2+|S_1|) - \log(\sigma^2+|S_2|) \\
&- \frac{1}{\sigma^2}\sum_{i=1}^{2}[\sum_{k\in S_i}s_k{}^2 - \frac{1}{(\sigma^2+|S_i|)}(\sum_{k\in S_i}s_k)^2]\} \qquad (5.12)
\end{aligned}$$

Assuming $\sigma^2 \ll |S_i|$ gives

$$\hat{r}_{ML} \approx \arg\max_r\{-\log[|S_1||S_2|] - \frac{1}{\sigma^2}[\sum_{i=1}^{2}|S_i|\hat{\sigma}^2_{S_i}]\} \qquad (5.13)$$

Equation (5.13) has an intuitively pleasing interpretation. Specifically, the ML estimate for the r is obtained by separating the data into the two sets so that we minimize the sum of the sample variances in each of the two regions plus an energy term.

The local edge detection algorithm proposed in this section is based on a model which assumes data on either side of the edge is constant and observed in additive white noise. The ML estimate of the edge location consists of finding the edge offset which minimizes the sum of the sample variances on either side of the edge. The model is similar in character to those used by other edge detection algorithms found in the literature [19], [21]. However, it seems that this edge estimation algorithm based on this model is unique. Although, for the purposes of this thesis the algorithm for estimating the edge location looks only for the edge offset, the proposed model could be used for estimating both offset and angular orientation. The prior knowledge obtained from dip estimation algorithms (which indicates the likely value of the edge's angular orientation) described in Chapter 4 is utilized here to simplify the edge estimation problem.

The next section illustrates the use of the ML estimation procedure described here.

5.3 Examples of Local Edge Estimation

This section demonstrates some of the characteristics of the ML edge estimation procedure described in Section 5.2. In particular, the examples discussed here illustrate the effect of the parameter σ^2 on the performance of the algorithm. Also, some approximations to the ML estimation procedure are examined.

As in Chapter 4 the edge estimation algorithm is applied to a large data set which has varying edge characteristics. The algorithm is applied to the data by moving a small window over the data assuming that the characteristics of the edge model hold within the window boundaries. The edge estimation algorithm described in Section 5.2 is repeatedly used on the data within the confines of the window as it is moved over the data. For each window location an edge is estimated and the location of the edge is associated to the center pixel of the window. In this manner, and edge estimate field is produced (similar to the dip estimate fields of Chapter 4).

The data set which receives the most attention in this chapter is illustrated in Figure 5.2. This data set is used to illustrate the edge estimation algorithm of

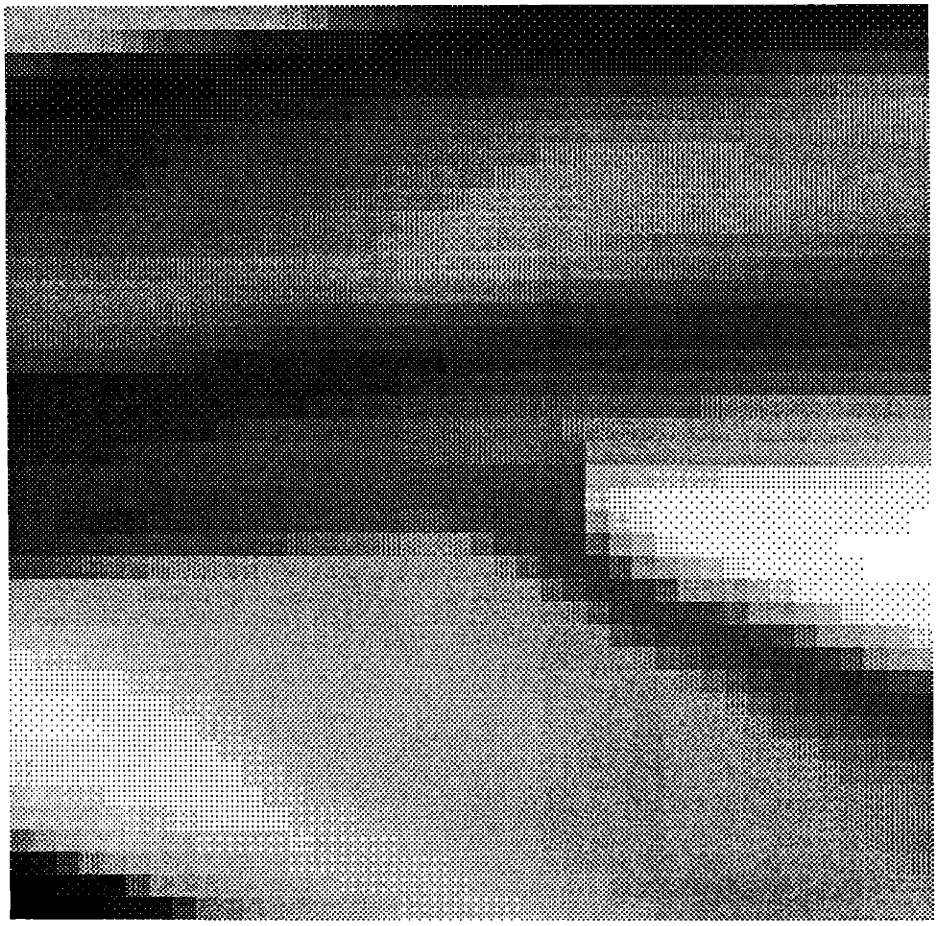

Figure 5.2: Raw Data

Section 5.2. The examples of this section assume that the data window consists of a 7×7 array of data elements. Furthermore, the prior information concerning the slope of edge estimates is given by the dip estimates obtained from the data in Figure 5.2 using the SPROJ algorithm described in Chapter 4. These dip estimates are illustrated in Figure 5.3.

Figure 5.4 illustrates an edge estimate based on the raw data and dip data of Figures 5.2 and 5.3. In the figure each of the large dots represents an estimated edge location. The small dots represent the locations of pixel centers. Each edge is associated with a pixel, and this is illustrated by a line connecting the pixel's small dot to the edge's large dot. For clarity, the ensuing examples display neither the small dots representing pixel locations nor the lines connecting pixels to their edges.

The first three examples in this section are based on (5.12) in Section 5.2. In the three examples the value of σ^2 changes. The values for σ^2 are .01, .1, and 1.0. The estimates based on these values are illustrated in Figures 5.5-5.7, respectively.

It can be seen that the sharpest edges in Figure 5.2 correspond to highly clustered edge locations in Figures 5.5-5.7. However, there are clear bed boundaries in Figure 5.2 which correspond to edge estimates which are not so well clustered. An example of this is highlighted in Figure 5.5. The bed boundary in the data corresponding to the region of edge estimates boxed in Figure 5.7 has an important characteristic. This bed boundary has a cross-section which consists of a broad ramp rather than a sharp jump. The model for our estimation algorithm assumes a sharp jump. Consequently, there is spread in the edge estimates as windows which are farther away from the bed boundary continue to find a portion of the ramp cross-section.

As the value of σ^2 increases the amount of scatter in the edge estimates increases in the three figures. This is a result of the algorithm being less sure of low contrast edges in the assumed presence of more noise. In particular, the boxed region of Figure 5.6 corresponds to a clear, but low contrast, bed boundary in Figure 5.2. As can be seen, for $\sigma^2 = .1$ this boundary is fairly clustered, however, less so than for $\sigma^2 = .01$. For $\sigma^2 = 1.0$ this bed boundary completely disappears.

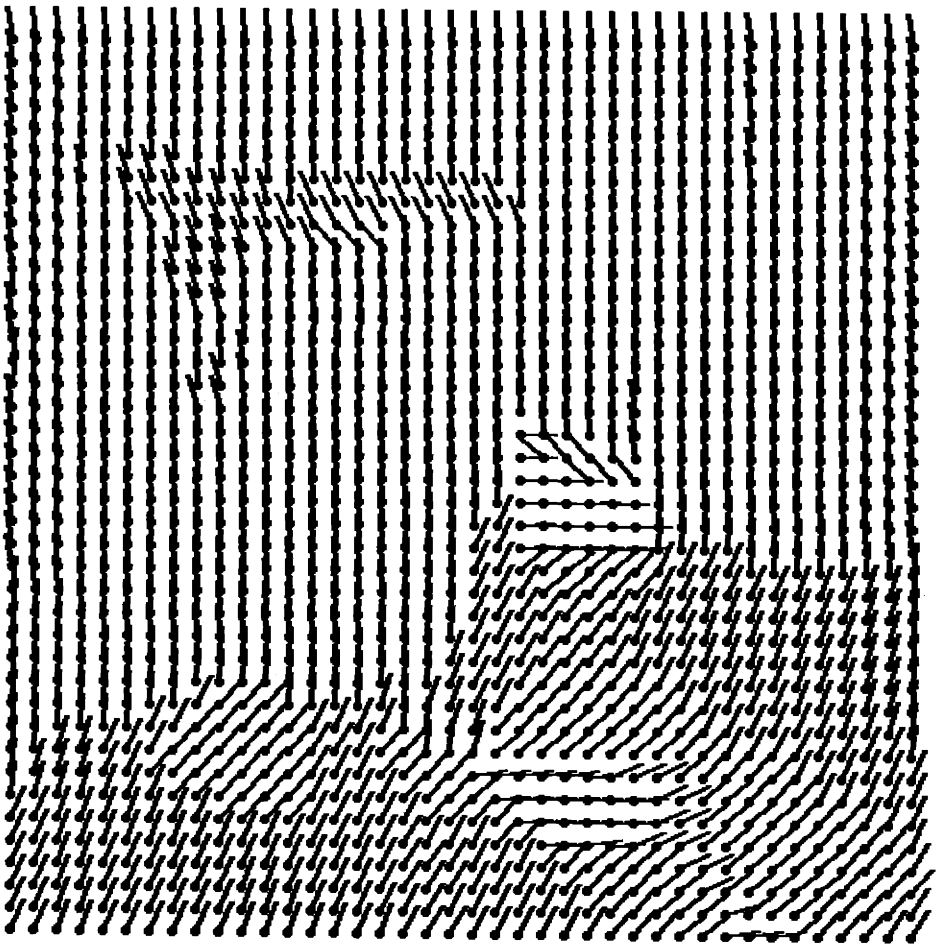

Figure 5.3: Dip Estimates Obtained Using the SPROJ Algorithm

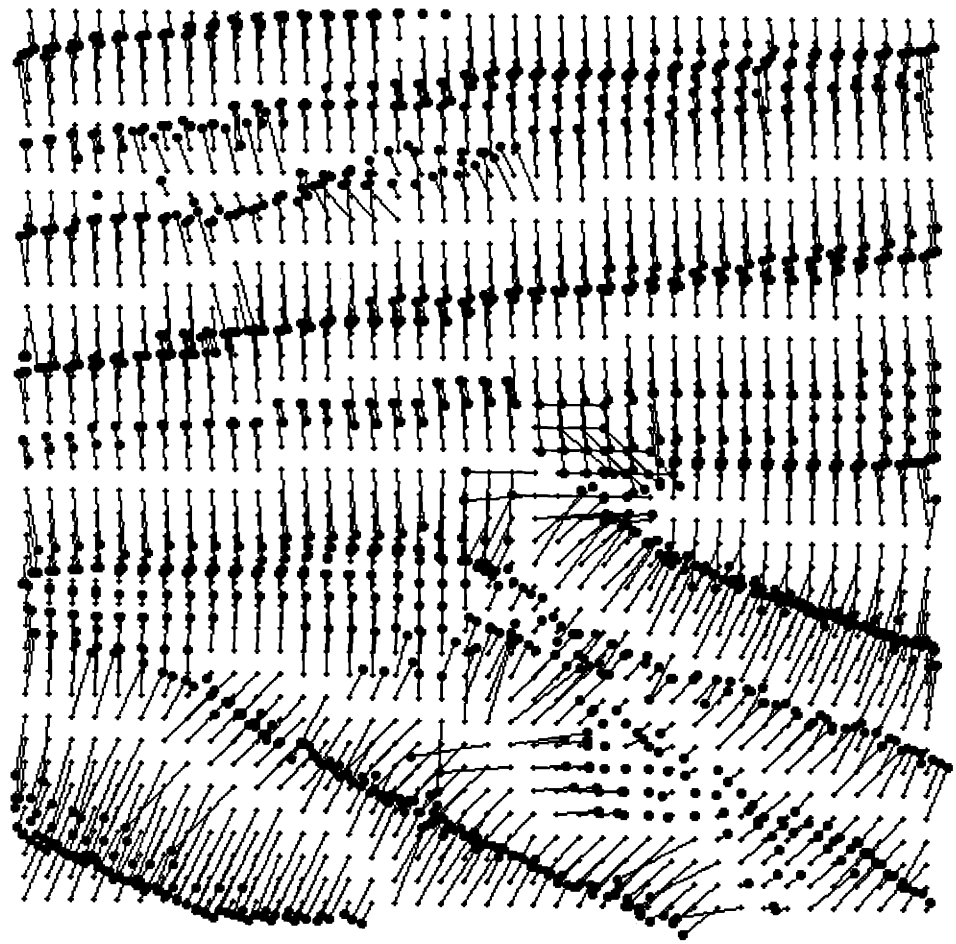

Figure 5.4: Edge Estimates and Their Respective Pixel Locations

Figure 5.5: Edge Estimates for $\sigma^2 = .01$.

Figure 5.6: Edge Estimates for $\sigma^2 = .10$.

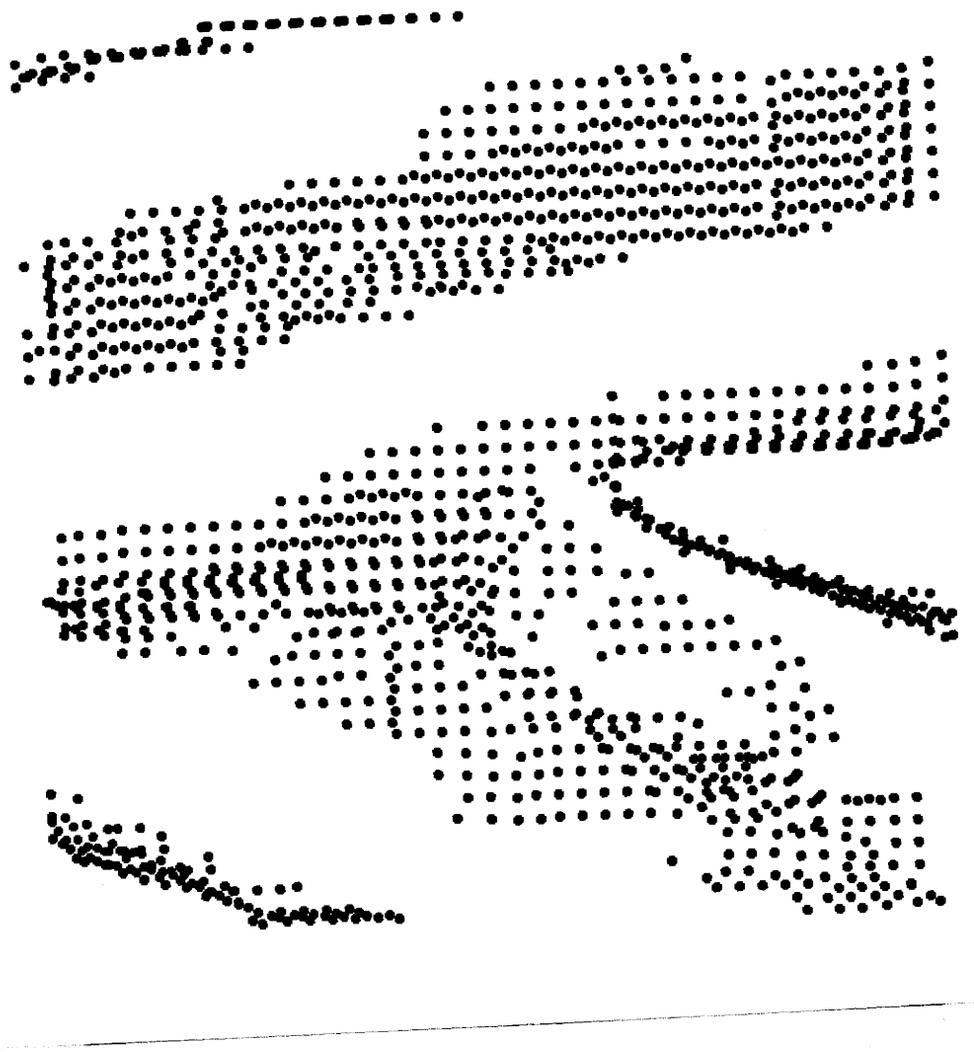

Figure 5.7: Edge Estimates for $\sigma^2 = 1.0$.

The following two figures, Figure 5.8 and Figure 5.9 illustrate edge estimates based on the algorithm suggested by (5.13). Here it has been assumed that the value of σ^2 is much less than the number of data elements in either group separated by the edge. In Figure 5.8 $\sigma^2 = .01$ and in Figure 5.9 it takes the value $\sigma^2 = 1.0$. It is not surprising that the results in Figure 5.8 look identical to those in Figure 5.5. The value for σ^2 used in generating these two pictures is .01. Since the number of data elements in a set can be no less than 1 the assumption that σ^2 is much less than the number of elements of either set is satisfied. However, for $\sigma^2 = 1.0$ we have a situation where sometimes the number of group elements approaches the value of σ^2. The approximation used to obtain (5.13) does not hold under this condition which is reflected in the additional scatter observed in the estimates of Figure 5.9. The final example of this section is illustrated in Figure 5.10. Here a markedly different approximation is used. First, the log term of (5.13) has been discarded. Second, the weighted sum of sample variances in (5.13) has been changed so that it is an unweighted sum of sample variances.

The rationale for the algorithm is as follows. The weighting of the sample variances by the number of elements in their respective groups has an effect on the estimation procedure. This effect tends to favor edge estimates which are closer to the center of the data window. Consequently, since our data windows are small, the edges tend to be biased away from strong bed boundaries when these boundaries intersect the window far from its center.

The illustration of the unweighted sum of sample variances algorithm is shown in Figure 5.10. The value of σ^2 is assumed to be very small since, as stated, the log term of (5.13) has been discarded. As can be seen, the estimates are much more clustered than in the previous figures. It is this algorithm which is used in the discussion of algorithms for global line estimation described in the remainder of this chapter.

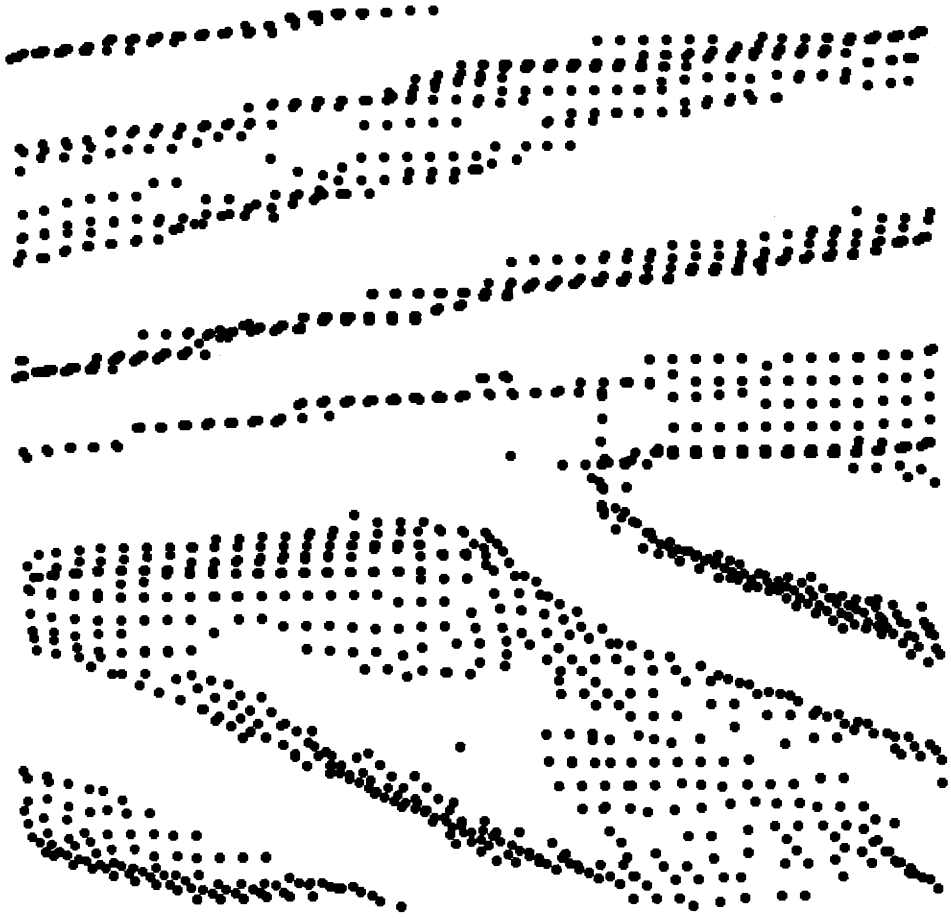

Figure 5.8: Edge Estimates for $\sigma^2 \ll |S_i|$, $i = 1, 2$, $\sigma^2 = .01$

Figure 5.9: Edge Estimates for $\sigma^2 \ll |S_i|$, $i = 1, 2$, $\sigma^2 = 1.0$

Figure 5.10: Sum of Variances Edge Estimation Method

5.4 Global Line Fitting by Clustering

In Sections 5.1 and 5.2 we discussed modeling and estimation of local edges. As was illustrated in Section 5.3 the edge estimation procedure produced estimates for large data sets by using a moving window. These edge estimates were highly clustered along bed boundaries. The objective of this section is to reduce the quantity of edge estimates. This is accomplished by fitting straight lines to appropriate portions of the edge estimates. The line fitting serves to compress the information in the data by identifying the edges (and consequently dips) between layer boundaries.

The type of data we have considered in this thesis consists of an unknown number of layers and, consequently, our methods must be independent of this knowledge. Our method of global line fitting of local edge estimates begins by clustering local edges together. Edges which seem likely to belong to the same global line are clustered together. Each group of local edge estimates is then fit by a line which is a least squares fit of the edges.

An examination of Figure 5.4 reveals the underlying goal of the clustering portion of the algorithm. As can be seen, the pixel locations seem to be groupable in terms of how their respective edges cluster around the major bed boundaries in the data. One intuitively reasonable grouping is illustrated in Figure 5.11. The objective, then, is to find some rules which allow us to partition the data elements into such groups.

The grouping part of the algorithm begins by choosing an element of the data array. This element has a dip and edge estimate associated with it. The algorithm then finds all the elements which adjoin the first element and which satisfy certain constraints. This is followed by checking adjoining elements of the elements adjoining the first element. In this way a region is grown from a single element which consists of elements which have characteristics similar to the first element.

The above-mentioned constraints are as follows. First, the value of dip cannot drift too far from the first element's dip. Second, the distance between local edges estimates of adjoining elements must be less than some specified value. Finally, the geometry of edge estimates belonging to adjoining elements must satisfy certain

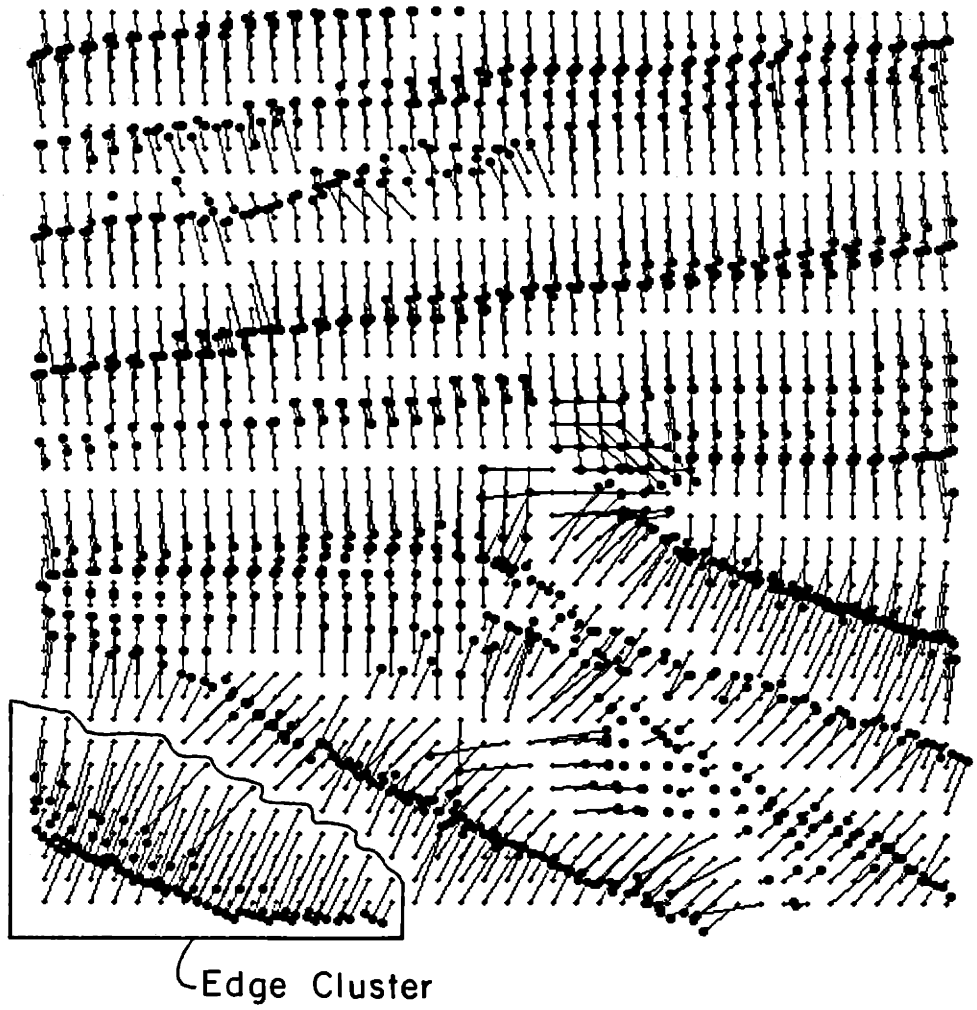

Figure 5.11: A Cluster Group

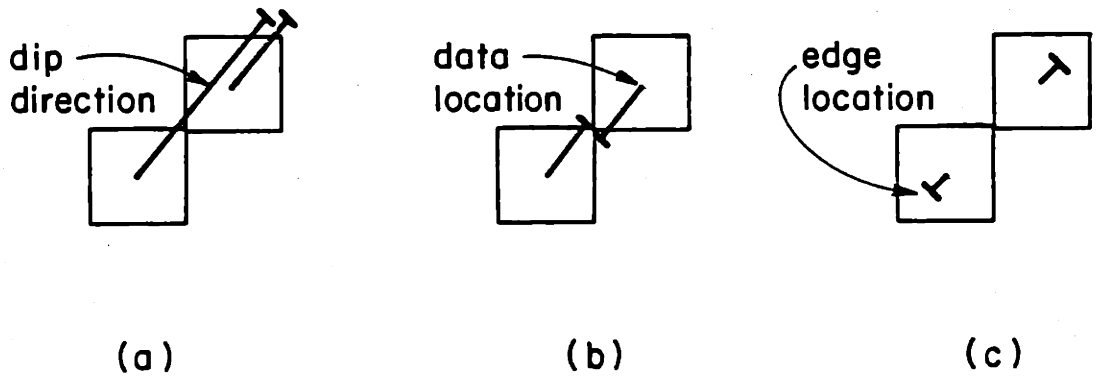

Figure 5.12: Types of Edge Geometries

conditions.

The geometric conditions which must be satisfied by edge estimates are illustrated in Figure 5.12. Figure 5.12 illustrates adjacent pairs of data elements (represented by squares). Also illustrated are the edge estimates associated with the data elements. The edges are connected to the data elements by lines starting in the data element center and terminating at the edge center. The connecting lines represent dip directions. The function of the geometric condition is to rule out groupings of elements which satisfy the dip and edge distance constraints but from geometric considerations should not be clustered together. The illustration in Figured 5.12c depicts a situation where the dip is equal and the edge locations are close. However, the connecting lines point in opposite directions. Under these conditions it seems evident that the data elements should not be clustered together. The geometries in Figures 5.12a-b are acceptable. The rule for deciding whether the geometry is acceptable or not is as follows. Project the edge locations onto the line connecting the data elements. If both edge projections fall between the data element locations or if both fall to the same side of one of the data element locations then the geometry is acceptable (Figure 5.12a and b). If one edge projection falls

on one side of both data elements and the other edge projection falls on the other side of both data elements then the geometry is unacceptable (Figure 5.12c).

Consequently, each element of a cluster must satisfy three conditions with at least one other neighboring element in the cluster. For these two elements we denote the pixel locations by p_i and p_j, their respective dip values as d_i and d_j, and the edge locations as e_i and e_j. The first two constraints are

$$d_i \cdot d_j \geq t_d \tag{5.14}$$

$$|e_i - e_j| \leq t_e \tag{5.15}$$

where t_d and t_e are positive constants. The final constraint is that the projections

$$\begin{aligned} f_i &= (e_i - p_i) \cdot (p_j - p_i) \\ f_j &= (e_j - p_i) \cdot (p_j - p_i) \\ p &= (p_j - p_i) \cdot (p_j - p_i) \end{aligned} \tag{5.16}$$

may not configure such that $f_i < 0$ and $f_j > p$ or $f_j < 0$ and $f_i > p$.

After all the elements associated with the first cluster have been identified, a new cluster is started. The new cluster is begun by choosing a data element which is not assigned to any cluster yet. Then a cluster is grown from this new data element. In this way the entire data set is partitioned into sets of elements which contribute to an individual global line.

The second stage of the algorithm performs a least squares fit of a line to the edge elements in each group. The least squares fit of a line is a simple computation to make. For example, assume that one of the edge groups consists of a set of edge locations $\{(x_i, y_i) | i = 1, 2, ..., N\}$. We desire to find the parameters (a, b) such that

$$E = \sum_{i=1}^{N} {e_i}^2 = \sum_{i=1}^{N} (y_i - ax_i - b)^2 \tag{5.17}$$

is minimized. The values of (a, b) which minimize E are (see [7])

$$a = \hat{\lambda}_{xy}/\hat{\lambda}_{xx} \tag{5.18}$$

$$b = \hat{\mu}_y - a\hat{\mu}_x \tag{5.19}$$

$$\hat{\mu}_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5.20}$$

$$\hat{\mu}_y = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{5.21}$$

$$\hat{\lambda}_{xy} = \frac{1}{N}(\sum_{i=1}^{N} x_i y_i) - \hat{\mu}_x\hat{\mu}_y \tag{5.22}$$

$$\hat{\lambda}_{xx} = \frac{1}{N}(\sum_{i=1}^{N} {x_i}^2) - (\hat{\mu}_x)^2 \tag{5.23}$$

The length of the line is also determined by the edge locations $\{(x_i, y_i)|i = 1, 2, ..., N\}$. The line is terminated at the two edge locations which have the most extreme projections onto the least squares line. Figure 5.13 illustrates the procedure.

One might suggest a modification to the above least squares estimation of the slope a. Since we have the estimated dip values for each of the elements in a cluster group we can use this information to obtain an estimate of the slope associated to the cluster. For example, if $\hat{d}$ is the average dip of the cluster then a reasonable estimate of the slope is $a = -\hat{d}_x/\hat{d}_y$ where $\hat{d}_x$ and $\hat{d}_y$ are the x and y components of $\hat{d}$, respectively. However, such a procedure is subject to bias errors in the estimated local dip values. If for some reason many of the local dip estimates are not perpendicular to the bed boundary (see Figure 5.14) then the global slope estimate obtained by averaging these dips will also be biased. Consequently, it is more sensible to rely on the global information contained in the clustered elements to obtain the slope estimate than it is to rely on the local dip estimates. This conclusion has been born out by our experimental observations. Another possibility, of course, is to use some convex combination of the two methods for estimating the slope. The weighting on the two estimates would then reflect our prior confidence on the reliability of the two methods.

The next section illustrates the use of the global edge estimation procedure.

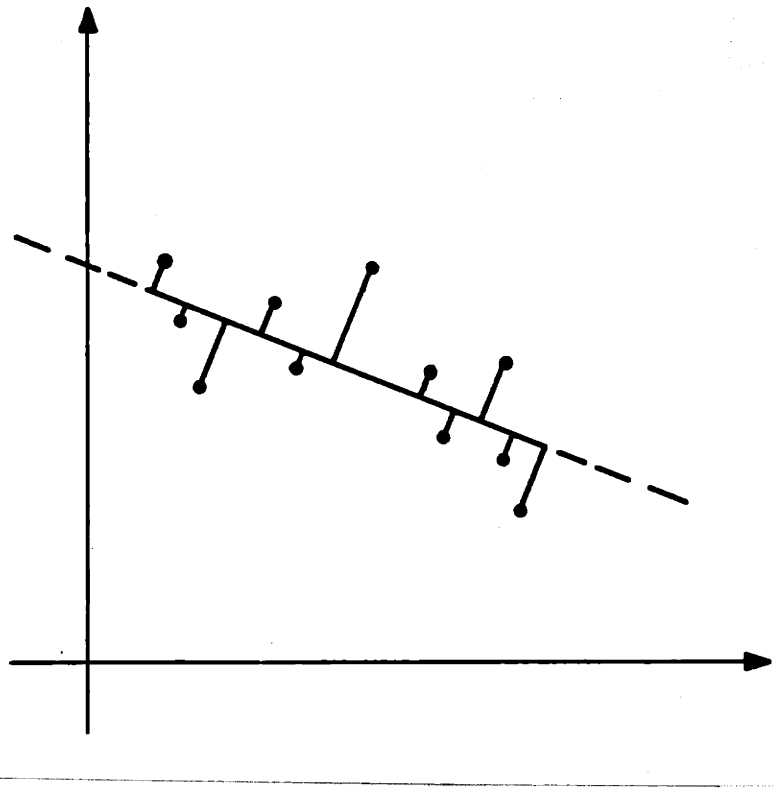

Figure 5.13: Least Squares Fit of Line to Data Where Line Boundaries are Determined by Extreme Projections of Data

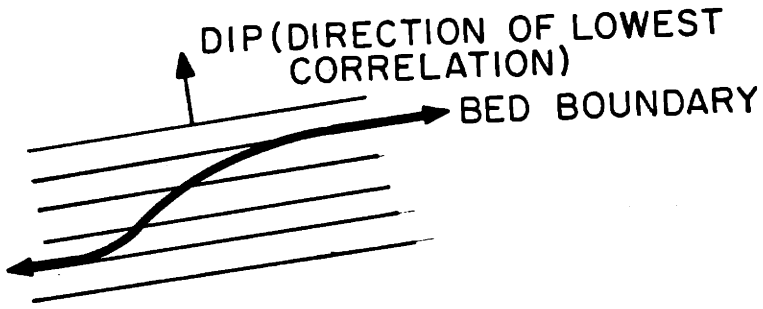

Figure 5.14: Edge Which is Not Perpendicular to Dip

5.5 Examples of Global Line Estimation by Clustering

In this section we illustrate the algorithm described in Section 5.4. The input data to the algorithm is the data set illustrated in Figure 5.10. Three examples are discussed in this section. These examples illustrate the behavior of the clustering threshold parameters for edges and dip.

In each of the ensuing examples we choose an edge and dip threshold. The edge threshold is the maximum distance between any edge in a cluster and the seed edge for that cluster. The dip threshold is the value for which all the dips in the cluster must have inner product less than or equal to with that of the seed dip. These two threshold values are referred to as t_e and t_d (see (5.14) and (5.15)), respectively.

The first example is illustrated in Figure 5.15. In this example the values of the cluster thresholds are $t_e = .9999$ and $t_d = .9999$. These two threshold values are significant for the following reasons. The pixels of the original data are of unit distance from each other. Consequently, edges which fall right at the center of their windows will not be clustered together with $t_e = .9999$. The value of t_d is chosen so that only data elements with identical dip are clustered together. The dip values of the data in Figure 5.3 are discretized to values of $\pi/51$. Consequently, since $\cos(\pi/51) = .9981$, when $t_d = .9999$ the dips within a cluster must all have the same value.

As can be seen in Figure 5.15 the global line estimates consist of lines which range in length from small lines to lines which almost span the image. The number of clusters obtained for this figure was 380 and the clusters consisting of a single element are not displayed. Figure 5.16 is an overlay of the estimates in Figure 5.15 on the raw data. As can be seen, the estimates nicely superimpose on the bed boundaries of the data.

The example of Figure 5.17 illustrates the global line estimates for $t_e = .75$ and $t_d = .9999$. Clearly, most of the lines become shorter due to the more stringent constraints imposed on cluster membership. The number of cluster groups for this example was 724.

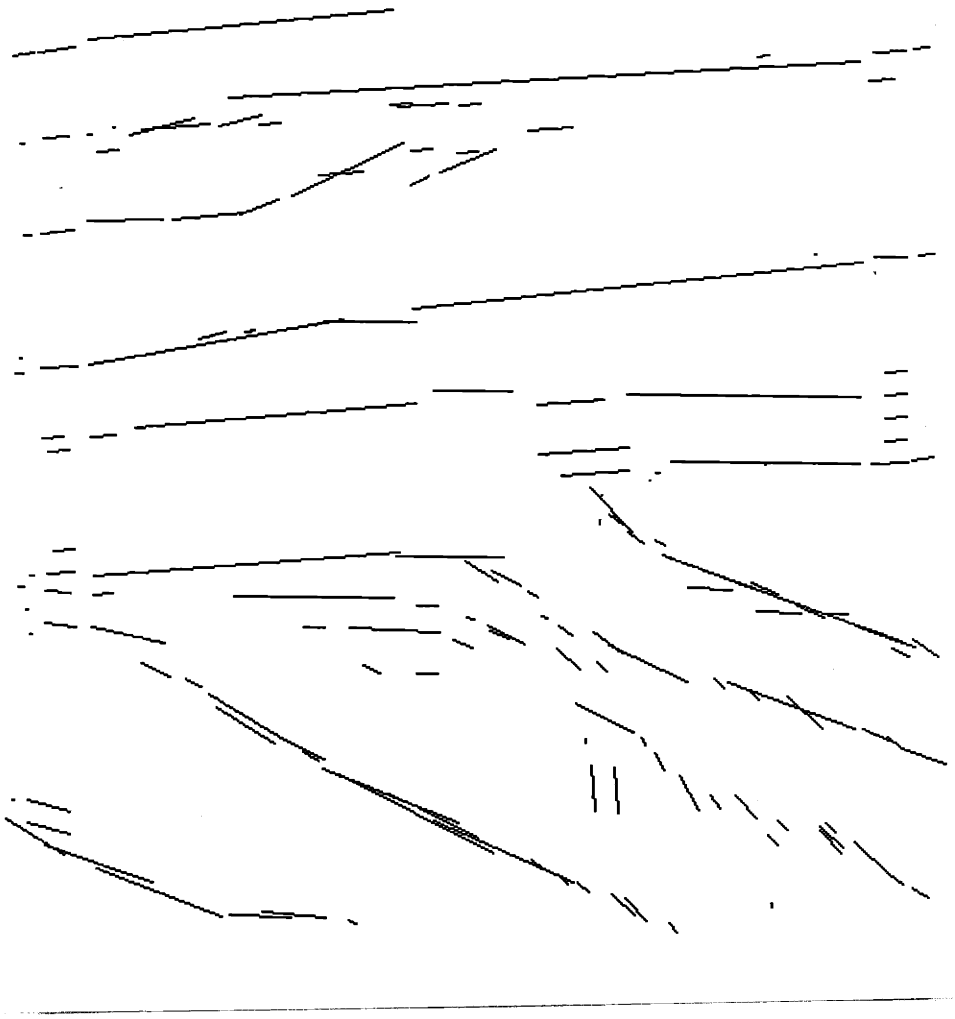

Figure 5.15: Global Line Estimate for $t_d = .9999$ and $t_e = .9999$

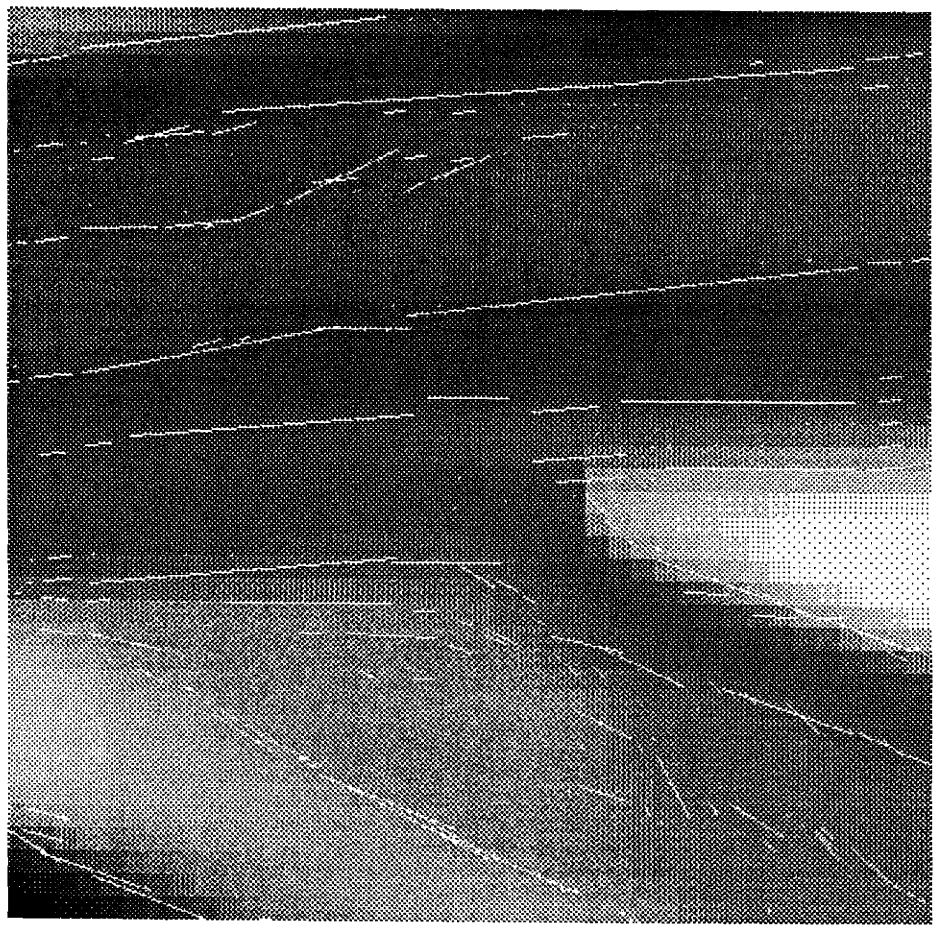

Figure 5.16: Overlay of Global Line Estimates on Raw Data

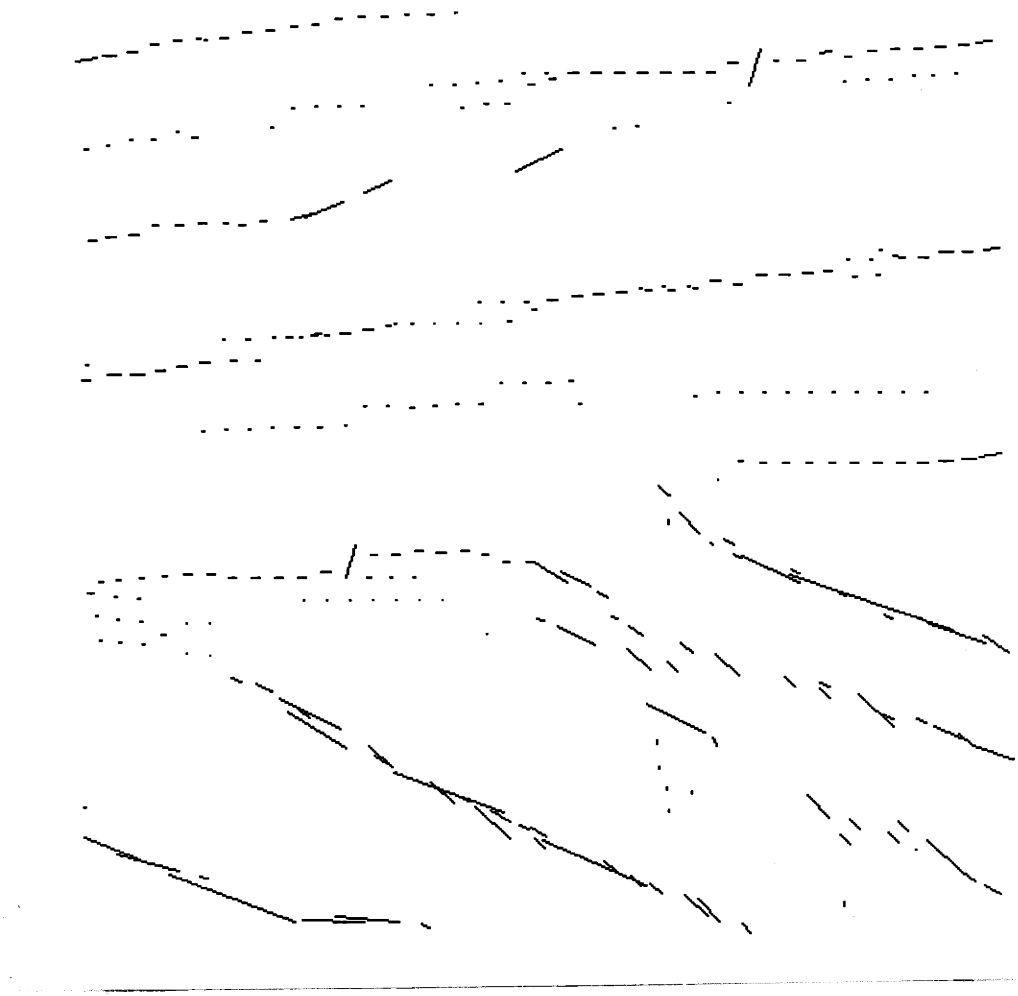

Figure 5.17: Global Line Estimates Using $t_e = .75$ and $t_d = .9999$

The final example is illustrated in Figure 5.18 where the cluster thresholds are $t_e = 1.5$ and $t_d = .86$. In this example both thresholds have been changed to dramatically increase membership of cluster groups. The total number of cluster groups in this example was 34. Since the thresholds are greatly relaxed the cluster groups contain many more elements and consequently produce global line estimates which are highly biased. Too many elements are contained in many of the cluster groups to obtain satisfactory results from the least squares line fitting performed in the algorithm.

The global line estimation algorithm described in Section 5.4 does produce some reasonable compression of the dip and edge estimates for certain values of t_e and t_d. More compression can be had, however, only at the price of substantial bias in the line estimates. Furthermore, the cohesiveness of the line estimates obtained in the global line estimation algorithm is low. It could be desirable to tie together lines which are associated to the same bed boundary so that only long lines exist. This would be an improvement in the cohesiveness of the global line estimates. The next section addresses this goal by introducing an algorithm which employs a bank of Kalman filters on the raw edge and dip data. Later, the output of the global line estimation algorithm is used to drive the Kalman filter bank algorithm.

5.6 Edge Tracking Using the Kalman Filter

The results in Sections 5.4 and 5.5 are encouraging. The objective of these sections is to show how to compress the quantity of local edge estimates obtained from algorithms proposed in Section 5.2. The edges are somewhat compressed by the algorithms of Section 5.4 however the degree of compression must be limited. Specifically, if the algorithms of Section 5.4 are used to obtained too high a degree of compression then a high degree of bias is present in the fitted lines. This is because the algorithm attempts to fit straight lines to clusters of data. If the clusters are large due to loose thresholds then there is some chance that the cluster is associated with a curved line. The straight line fit to this will then necessarily be biased. When the thresholds are tight relatively straight lines are broken up into

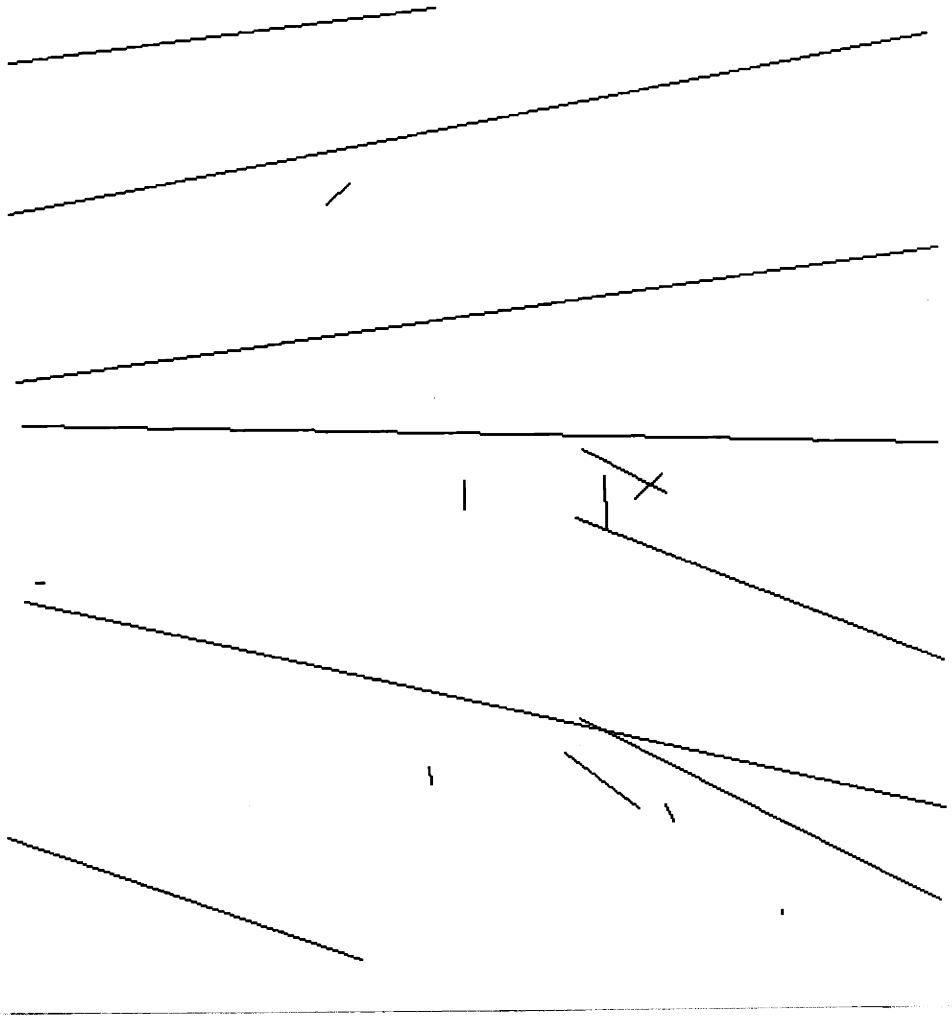

Figure 5.18: Global Line Estimates Using $t_e = 1.5$ and $t_d = .86$

many pieces. These pieces lead to global line estimates which lack cohesiveness due to the fact that the lines are not connected.

The objective of this section is to describe an algorithm which produces a more cohesive grouping of the edges into global line estimates. This is accomplished using a modified version of an edge tracking algorithm described in [1]. The method employs a bank of Kalman filters which track edges from one side of the data set to the other.

The output of the algorithm in Section 5.3 yields locations of local edge estimates. These estimates consist of the edge locations. The edge orientations (slopes) are already known from dip estimation algorithms described in Chapter 4. A bank of Kalman filters is used in this section to track these local edge estimates. Each of the Kalman filters is governed by the same dynamics and observations model. This model is described first and is followed by a discussion on how the model is implemented to accommodate the local edge estimates of Section 5.4.

5.6.1 Edge Tracking Model

The edge tracking model is simple. It consists of discrete dynamics and observation equations. They are as follows

$$\begin{aligned} \underline{x}_{k+1} &= F\underline{x}_k + \underline{w}_k \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \underline{x}_k + \underline{w}_k \qquad (5.24) \\ \underline{y}_k &= \underline{x}_k + \underline{v}_k \qquad (5.25) \end{aligned}$$

where $\underline{x}_k$ is a two-vector for which the first element is the edge location and the second element is the edge slope. The dynamics (5.24) state that the edge location at $k+1$ is equal to the edge location at k plus the slope at k (which is appropriate if the horizontal distance from location k to $k+1$ is unity). Furthermore, the slope at location $k+1$ is the same as that at location k subject to the addition of dynamics noise $\underline{w}_k$. This allows our filters to track lines that vary somewhat from straight lines. The observation equation (5.25) is simply the state vector in additive noise,

$\underline{v}_k$. Both $\underline{w}_k$ and $\underline{v}_k$ are assumed to be zero-mean, white, Gaussian processes which are mutually independent of each other.

The covariance matrices of $\underline{w}_k$ and $\underline{v}_k$ need to be specified. These covariance matrices are denoted Q_k and R_k, respectively, and in this model take the forms

$$Q_k = \begin{bmatrix} q1_k & 0 \\ 0 & q2_k \end{bmatrix} \tag{5.26}$$

$$R_k = \begin{bmatrix} r1_k & 0 \\ 0 & r2_k \end{bmatrix} \tag{5.27}$$

5.6.2 The Kalman Filter Equations

Since the model in Section 5.6.1 is for a two-dimensional state vector the Kalman filter equations are easily obtained. The Kalman filter equations are composed of three parts. There are the initialization equations

$$\hat{\underline{x}}(1|1) = \underline{x}_1 \tag{5.28}$$

$$p(1|1) = p_1 \tag{5.29}$$

where $\underline{x}_1$ is the initial state estimate and p_1 is the 2×2 error covariance matrix associated with $\underline{x}_1$.

The next part of the Kalman filter equations consist of the prediction updates

$$\begin{aligned} \hat{\underline{x}}(k+1|k) &= F\hat{\underline{x}}(k|k) \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \hat{\underline{x}}(k|k) \end{aligned} \tag{5.30}$$

$$\begin{aligned} p(k+1|k) &= Fp(k|k)F^T + Q_k \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} p(k|k) \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} q1_k & 0 \\ 0 & q2_k \end{bmatrix} \end{aligned} \tag{5.31}$$

Finally, there are the observation update equations

$$\hat{\underline{x}}(k|k) = \hat{\underline{x}}(k|k-1) + K_k[\underline{y}_k - \hat{\underline{x}}(k|k-1)] \tag{5.32}$$

$$p(k|k) = [I - K_k]p(k|k-1) \tag{5.33}$$

$$\begin{aligned} K_k &= p(k|k-1)[p(k|k-1) + R_k]^{-1} \\ &= p(k|k-1)[p(k|k-1) + \begin{bmatrix} r1_k & 0 \\ 0 & r2_k \end{bmatrix}]^{-1} \end{aligned} \tag{5.34}$$

where R_k is as in (5.27).

The next sub-section describes how the Kalman filter equations of this section are applied to the data (i.e. the edge locations and slopes). The issues addressed are concerned with specifying the observations in (5.25) and covariance values in (5.26) and (5.27).

5.6.3 Edge Tracking with the Kalman Filter Bank

The Kalman filter equations of Section 5.6.2 specify filtering for a single state process. The edge data generated by the algorithms of Section 5.2 consists of many state processes. We are interested in identifying the individual processes as well as tracking each process. This is accomplished by running a bank of Kalman filters, one filter for each process. However, there are two issues which must be resolved to accomplish this goal. The first issue concerns representation of the data in a form suitable for the Kalman filtering. The second issue deals with how observations of the data are assigned to the individual Kalman filters. This section discusses the first issue and the next section discusses the second.

The Kalman filter equations in Section 5.6.2 are for discrete observations of the data. Consequently, the data must be located on a discrete grid. The edge estimates produced by the algorithms in Section 5.2 are converted to absolute locations in the image and then are assigned to discrete locations on an $N \times N$ grid. The edge locations are represented on the $N \times N$ grid via an $N \times N$ matrix. If an edge exists at grid point (k,l) then a one is entered into the $(k,l)^{th}$ matrix location. The grid locations with no edges associated are represented by zeros in the corresponding matrix locations. Another $N \times N$ matrix contains the slopes of these edges where the slopes are obtained by computing the appropriate ratio of the dip components.

Four other $N \times N$ matrices are required. These matrices contain the associated entries of the noise variances $q1_k$, $q2_k$, $r1_k$, and $r2_k$. The parameters $q1_k$ and $q2_k$ are noise variances for the dynamics equation (5.24). These parameters must be chosen. The performance of the edge tracking algorithm will depend on the values chosen and more about this is discussed in a later section. For the moment it is sufficient to say that, for our purposes, $q1_k$ and $q2_k$ are constants independent of location.

The variances $r1_k$ and $r2_k$, however, reflect our confidence in the estimates of the k^{th} edge location and its slope, respectively. Since the estimates of the edge location and its slope (derived from the dip) are obtained by maximizing likelihood functions we can say something about the error variances associated with these estimates.

Our method of determining a bound on the error variance of estimates is by computing the Cramer-Rao lower bound. The Cramer-Rao bound is a lower bound on the error variance for an estimate. In the case of a linear estimation problem the Cramer-Rao lower bound is "tight". That is, the Cramer-Rao bound is equal to the error variance for linear estimation problems. Furthermore, in the linear estimation problem the expected value of the log-likelihood function is a down-turned quadratic of the form

$$y = -a(x-b)^2 + c \tag{5.35}$$

where c is the height of the maximum, b is the location of the maximum (that is the ML estimate of the parameter), and $1/(2|a|)$ is the Cramer-Rao bound.

Our procedure for obtaining $r1_k$ and $r2_k$ is to fit a quadratic of the form (5.35) to the sample likelihood functions computed in determining the ML estimates of edge locations and dip. The value of $1/2|a|$ is then taken to be our estimate for the error variance. This is not equivalent to computing the Cramer-Rao bound, however, it is similar in character. The procedure computes a as a least squares fit to the sample likelihood function in the vicinity of the maximum. That is

$$\hat{a} = \min_a E = \min_a \{\sum_{i=1}^{M} [y_i - c - a(x_i - b)^2]^2\} \tag{5.36}$$

where b and c are assumed to be known from computation of the ML estimate. The

value of $\hat{a}$ is obtained by differentiating E with respect to a and setting the result to zero.

$$\frac{\partial E}{\partial a} = -2\sum_{i=1}^{M}[y_i - c - a(x_i - b)^2](x_i - b)^2 = 0 \tag{5.37}$$

Solving (5.37) yields

$$\hat{a} = \frac{\sum_{i=1}^{M}(y_i - c)(x_i - b)^2}{\sum_{i=1}^{M}(x_i - b)^4} \tag{5.38}$$

Summarizing this section, we have converted the edge estimate locations to a discrete $N \times N$ grid. Associated with each edge location are five other parameters: the slope value of the edge (obtained from dip estimates), the dynamics noise variances $q1_k$ and $q2_k$ (chosen values), and the observation noise variances $r1_k$ and $r2_k$ (computed from sample log-likelihood functions). The next section discusses how observations of edges are assigned to Kalman filters.

5.6.4 Assigning Edge Estimates to Kalman Filters

As described in the previous section the edge locations are discretized to an $N \times N$ array. The Kalman filter bank tracks edges from one side of the array to the other. The question discussed in this section concerns how edges are assigned to the Kalman filters.

The bank of Kalman filters is initialized using the edge estimates on one side of the data array. Since the array is $N \times N$ there can be no more than N tracks. Consequently, the Kalman filter bank consists of N filters. In our implementation the left hand column of the data array is used to implement the Kalman filter bank. If an edge exists at the k^{th} position of the column then a Kalman filter is initialized there. The initial state values are simply the estimates (the edge location k and its corresponding slope) themselves. The initial error variance matrix is the diagonal matrix consisting of the estimated values for $r1_k$ and $r2_k$ (discussed in Section 5.6.3). If no edge exists at the k^{th} position then no filter is initialized there.

The next step of the algorithm consists of using the prediction update equations in (5.30) and (5.31) on the Kalman filters which have been initialized. The Kalman prediction update equations produce the best estimates of the edge and slope values

in the next column given the data in the present column. Furthermore, the error variance associated with these estimates is obtained.

The next stage of the algorithm is the critical step. The second column from the left contains edge locations and their slopes as well as error covariance values. The question is how to assign these values to the initialized Kalman filters. Several possibilities exist. For each Kalman filter it must be decided whether or not an edge can be assigned to it. For each edge it must be decided which Kalman filters it may possibly be assigned to.

The assignment of edges to Kalman filters can be formulated as an optimal hypothesis testing problem. However, the algorithm presented here does not perform hypothesis testing. Rather the algorithm presents a simple heuristic procedure for making the edge assignments.

The first step of the procedure does the following. For each of the Kalman filters the prediction updates and their error covariances are used to determine regions of the next column of the array where edges are predicted to be. That is, the l^{th} Kalman filter looks in a window of the next column which is centered at the prediction update for the edge location of that filter (see Figure 5.19). The window width is proportional to the square root of the error variance of the prediction update. Thus, the l^{th} Kalman filter looks in the range $[\hat{\underline{x}}_{k+1|k}(1) - \alpha {p_{k+1|k}}^{1/2}(1,1), \hat{\underline{x}}_{k+1|k}(1) + \alpha {p_{k+1|k}}^{1/2}(1,1)]$ for the existence of edges. If an edge is found then it is similarly checked to see whether the slope is in the window $[\hat{\underline{x}}_{k+1|k}(2) - \beta {p_{k+1|k}}^{1/2}(2,2), \hat{\underline{x}}_{k+1|k}(2) + \beta {p_{k+1|k}}^{1/2}(2,2)]$. An edge in column $k+1$ satisfying these conditions is a candidate observation for this Kalman filter.

The procedure of manufacturing windows for the Kalman filters based on the prediction update equations yields the observations for the filters. However, there are conflicts which must be resolved. The possible situations include no observations for a filter, exactly one observation for a filter, many observations for a filter, and one observation shared by more than one filter. We have used the following procedure for handling these situations.

First, we insist on one edge per Kalman filter. If no edge exists for the l^{th} Kalman filter then it is updated by prediction only for up to h steps. If no edge is found in

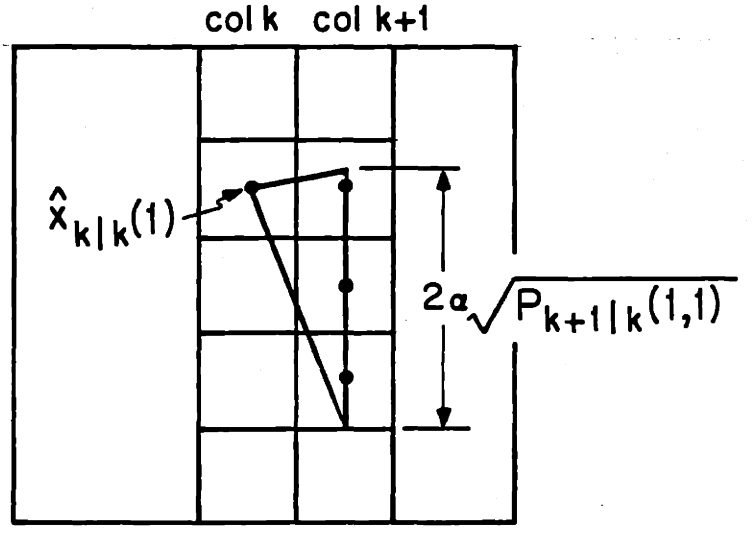

Figure 5.19: Kalman Prediction Window

these h steps the filter is terminated at the first prediction only update. If edges exist which are unassigned to any Kalman filter they are used to initialize new filters. An edge which belongs to only one window is assigned to the associated Kalman filter. Finally, the edges which could belong to several filters due to overlapping windows are assigned based on optimizing a weighted innovation process. That is, for each possible edge belonging to each possible window the following function is computed

$$|K_k[\underline{y}_k - \underline{\hat{x}}(k|k-1)]|^{\frac{1}{2}} \tag{5.39}$$

Over the entire set of edges which could belong to more than one filter, the edge and filter pair which minimize (5.39) are assigned to each other. Once this assignment is made that edge can be assigned to no other filter and that filter can have no other edge. The remaining filters are assigned edges in an identical fashion, always forming the next pair which minimizes (5.39).

The final stage of the algorithm performs the observation updates of the Kalman filters using the observation assignments of the previous stage. This in turn is followed by the prediction update stage again, etc, until the entire $N \times N$ array is traversed. The next section illustrates this procedure with some examples.

5.7 Examples of Edge Tracking Using a Kalman Filter Bank

This section illustrates the features of the Kalman filter bank algorithm described in Section 5.6. There are many parameters involved in the use of the Kalman filter bank algorithm. The objective of this section is to present some of the major characteristics of the algorithm.

The Kalman filter model presented in Section 5.6.1 is controllable and observable. Consequently, the filter characteristics are well behaved. The major characteristics of the Kalman filter bank algorithm are functions of the relative amounts of noise getting into the dynamics and observations of the model. The ratio of dynamics to observations noise controls the balance between the prior assumptions

concerning the model validity and the confidence in the observations of the data. The other parameter of interest is the one specifying the number of updates by prediction only that are allowed before termination of a line.

Addressing the trade-off between observation and dynamics noise we refer to Figures 5.20-5.23. Each of the figures illustrates the use of the Kalman filter bank algorithm where the input data is the dip data of Figure 5.3 and the edge data of Figure 5.10. Figures 5.20-5.22 illustrate the algorithm for specified values of observation and dynamics noise. Figure 5.23 uses observations noise estimated from the raw data as described in Section 5.6.3.

In Figure 5.20 the observation noise and dynamics noise have covariances which are equal and the values of these covariances are set to $q1_k = r1_k = 1.$ and $q2_k = r2_k = .5$. The value of the window scale parameter , α, is set to unity in this example and in all of the remaining examples. The maximum number of prediction only updates is $h = 6$.

The performance of the Kalman filter bank estimates in Figure 5.20 is good. For the most part edges have been tracked for extended distances. In many regions several edges are tracked which are very close to one another due to the spread in the original local edge estimates. There are a handful of points in Figure 5.20 which are the locations of local edges which were not successfully tracked to any other local edge. This is due to the failure to find appropriate edges located in the prediction windows of the Kalman filters associated to these edges.

Figures 5.21 and 5.22 are estimates from the Kalman filter bank algorithm for increased dynamics noise and increased observation noise, respectively. In Figure 5.21 the dynamics noise covariance is $q1_k = 1000.$ and $q2_k = 500.$ whereas the observation noise covariance is $r1_k = 1.$ and $r2_k = .5$. As might be expected, the algorithm gives more weight to the observation than the model dynamics under these conditions. Consequently, the tracks illustrated in Figure 5.21 look noisier than those in figure 5.20.

In Figure 5.22 the observation noise covariance is $r1_k = 1000.$ and $r2_k = 500$ whereas the dynamics noise covariance is $q1_k = 1.$ and $q2_k = .5$. Here the estimates are much smoother than those in Figures 5.20 or 5.21. The reason for this is that the

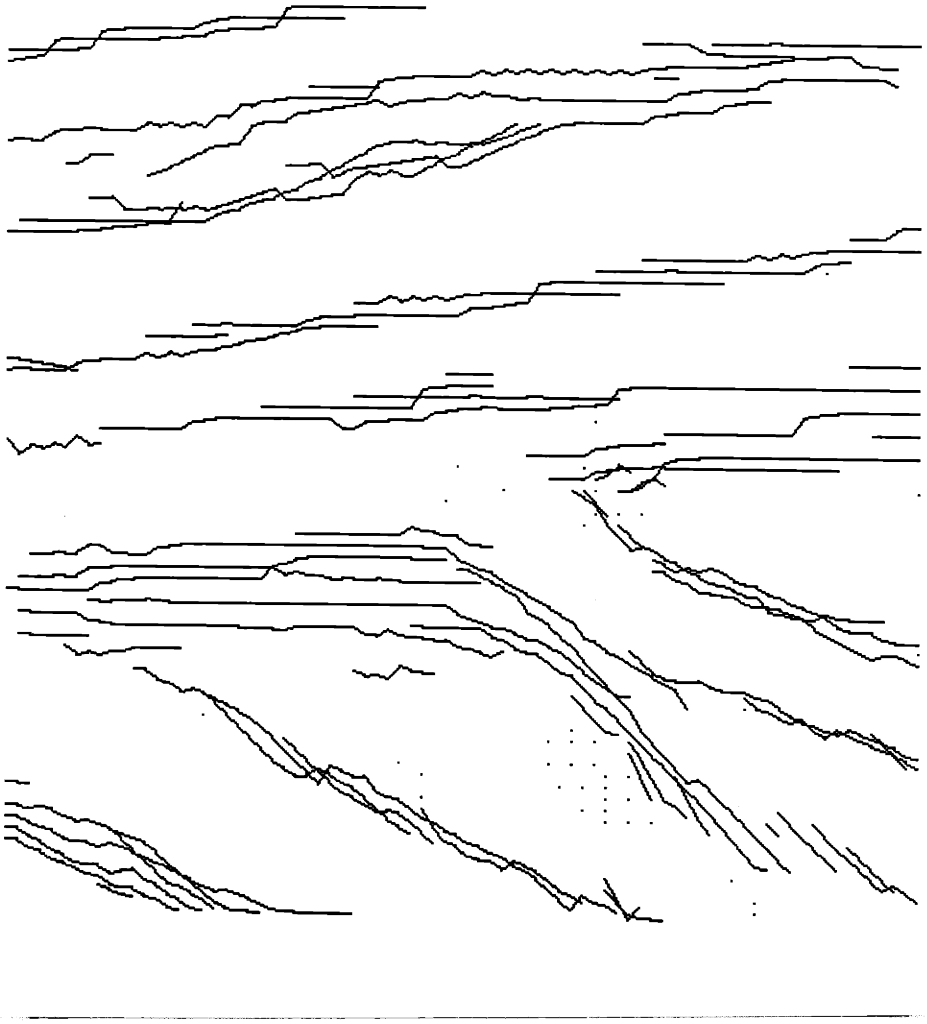

Figure 5.20: Kalman Filter Bank Estimates for $q1_k = r1_k = 1.$, $q2_k = r2_k = .5$, and $h = 6$

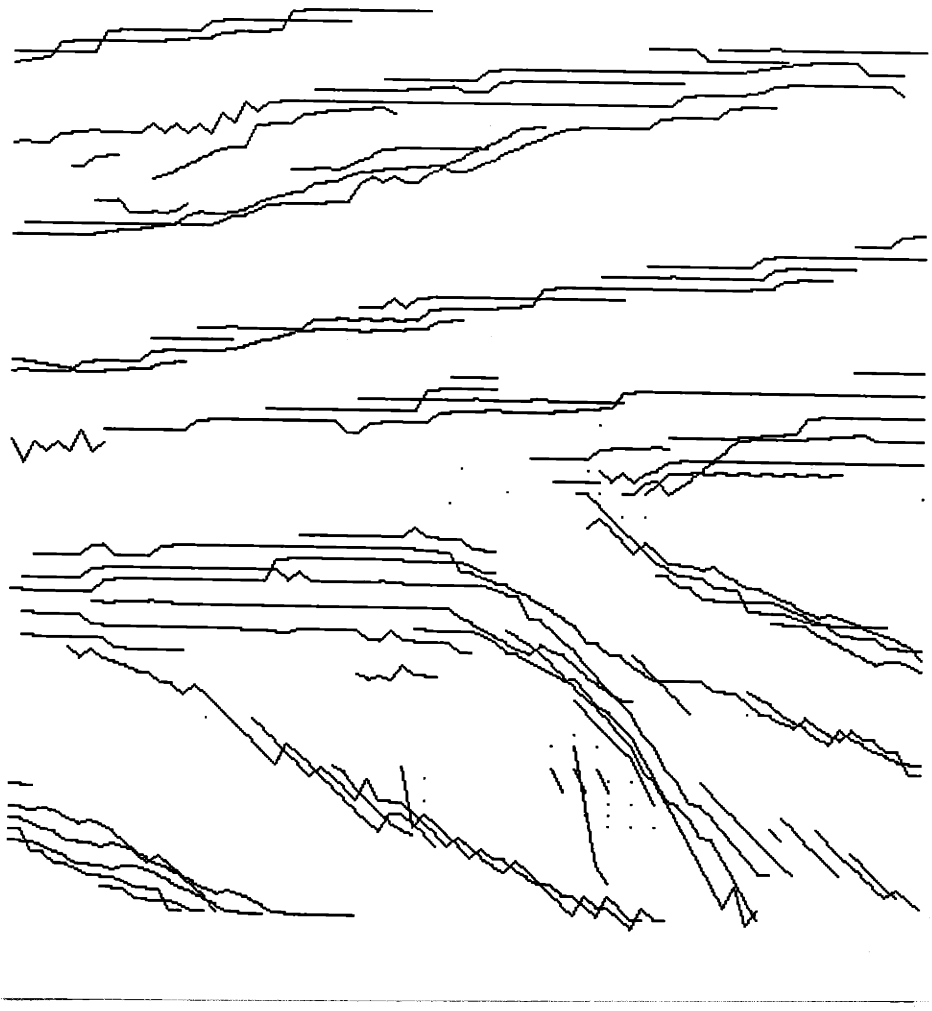

Figure 5.21: Kalman Filter Bank Estimates for $q1_k = 1000$, $q2_k = 500$, $r1_k = 1.$, $r2_k = .5$, and $h = 6$

filters are more confident of the model dynamics than they are of the observations. Consequently, the estimates are more likely to conform to long straight lines.

Figure 5.23 is similar to the previous figures, however, here the values of $r1_k$ and $r2_k$ are estimated and as such are not constant valued. The estimated values for $r1_k$ and $r2_k$ enter into the Kalman filter equations in the computation of the Kalman gain. In this way the confidence of the local edge estimates is directly accounted for in the way they are weighted in the observation update equation.

Comparison of the results in Figure 5.20, 5.22, and 5.23 is interesting. In Figure 5.20 the values for the covariance of the observation noise is small and so the estimated lines are noisy. In Figure 5.22 the values for the variance of the observation noise is high and consequently the estimated lines are smooth. However, the estimated lines are overly smooth and depend too heavily on the initial edge estimates at the left side of the figure. The estimates in Figure 5.23 find a nice middle ground. This is due to the use of the estimated values for $r1_k$ and $r2_k$ which gives rise to an appropriate scaling and balance for the observation noise in the filtering algorithm.

The final example of this section illustrates the effect of the prediction only threshold parameter, h. In the previous examples of this section $h = 6$. In Figure 5.24, $h = 18$. As can be seen in the figure the performance of the algorithm is most notable for the extended tracking of edges. In particular, there are several tracked edges which extended across regions for which it appears there should be no edges. However, the value of the prediction only threshold parameter is so large in this case that connections are made between local edges which are very far apart.

In conclusion, this section illustrates how a cohesive global line algorithm can be achieved using a Kalman filter bank. However, in comparison with the results of the global line estimation algorithm of Section 5.4, very little compression is achieved. The next section shows how both compression and cohesiveness can be obtained by employing both methods in conjunction with each other.

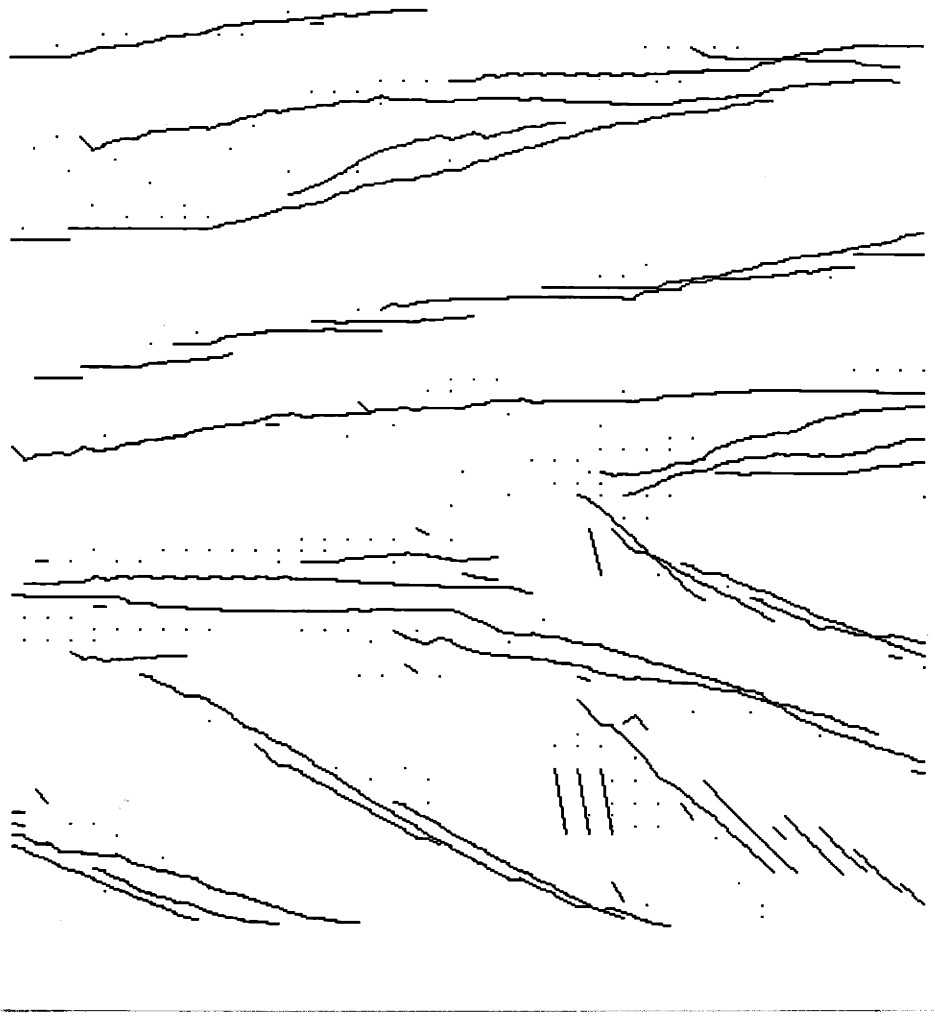

Figure 5.22: Kalman Filter Bank Estimates for $r1_k = 1000$, $r2_k = 500$, $q1_k = 1.$, $q2_k = .5$, and $h = 6$

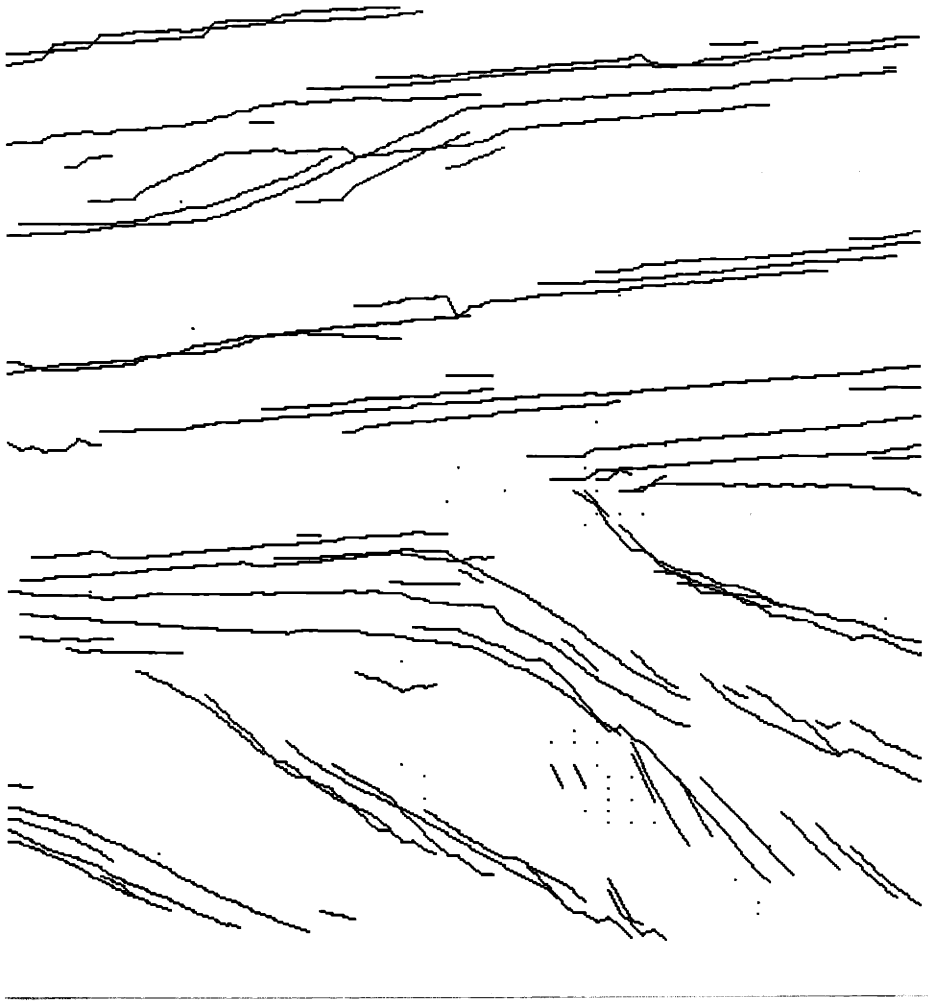

Figure 5.23: Kalman Filter Bank Estimates for $r1_k$ and $r2_k$ estimated, $q1_k = 1.$, $q2_k = .5$, and $h = 6$

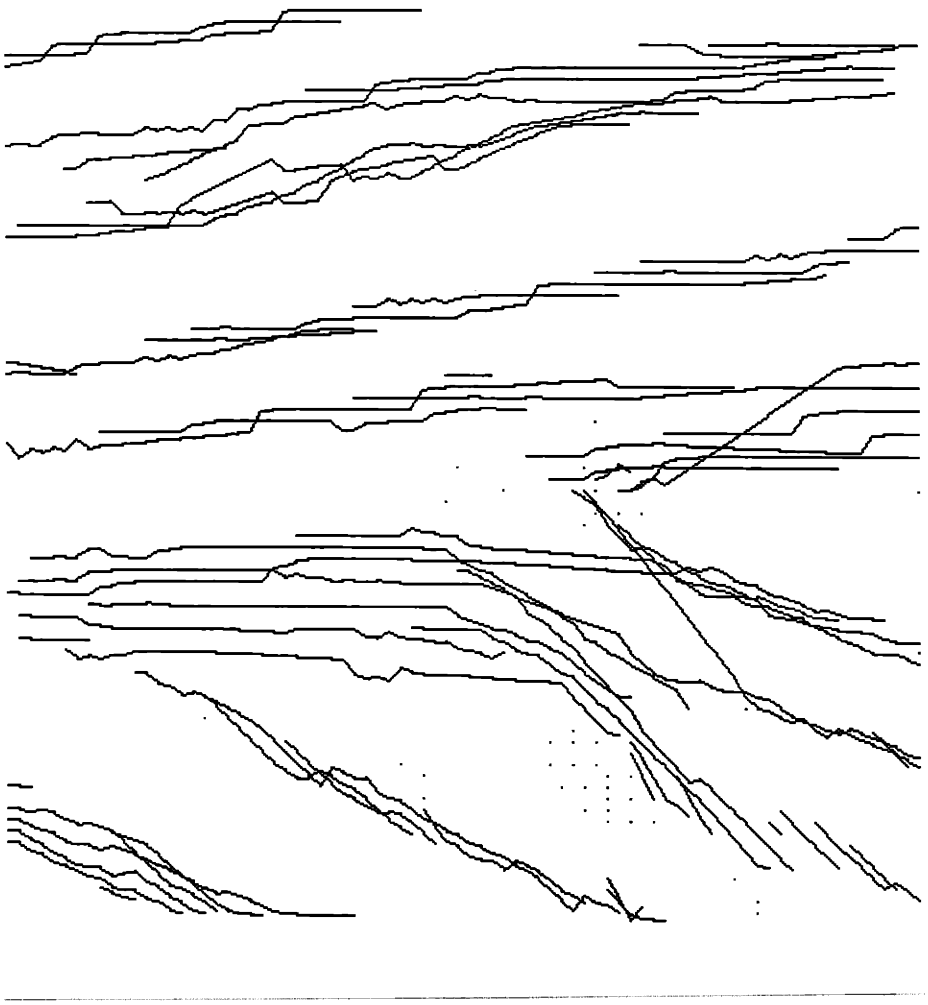

Figure 5.24: Kalman Filter Bank Estimates for $q1_k = r1_k = 1.$, $q2_k = r2_k = .5$, and $h = 18$

5.8 Combining Groupings with Kalman Filtering

As was seen from the examples in Section 5.7, the Kalman filtering approach for obtaining global line estimates has some undesirable features. These have to do with the fact that there is very little compression of the local edge estimates by using the Kalman filter approach[1]. The desirable features of the Kalman filtering approach are the cohesiveness of the global line estimates.

The Kalman filter global line estimates are in contrast with the grouping estimates illustrated in Section 5.5. The grouping estimates realize quite a bit of compression from the local edge estimates. However, the cohesion of lines is poor.

This section shows how combining the two techniques, Kalman filtering and the grouping estimates, can yield a high degree of compression and cohesion in the global line estimates. The method is quite simple. It proceeds by performing the grouping estimate first. The output of the grouping estimate is then converted to a form acceptable as input to the Kalman filter algorithm. The output of the Kalman filters then consists of a compressed (due to groupings) and cohesive (due to Kalman filtering) estimate of the global lines.

The conversion from the output of the groupings estimate to input acceptable to the Kalman filters is simple. The groupings' output consists of pairs of points representing the endpoints of straight lines. These endpoints are discretized to the $N \times N$ grid used in the Kalman filtering algorithm. Furthermore, for each column between the pair of converted endpoints there is a grid row location which is associated to an edge on this line.

In this way, all the straight lines from the groupings output are converted to edge locations in the $N \times N$ array for the Kalman filtering algorithm. The associated slope values come from the slope of the lines. The values for the error variances are chosen in the same manner as described in Section 5.7. The following example illustrates how this combination method performs.

Figure 5.25 shows the output of the Clustering-Kalman filtering algorithm. The inputs to the clustering algorithm are the edge and dip estimates of Figures 5.10 and

[1]There is some compression due to the discretization of local edges to an $N \times N$ grid.

5.3, respectively. We used the clustering parameters $t_e = .9999$, $t_d = .9999$ which produced output identical to that in Figure 5.15. The clustering algorithm output is utilized by the Kalman filter bank algorithm. The parameters of the Kalman algorithm are $q1_k = r1_k = 1.$ and $q2_k = r2_k = .5$ and $h = 6$.

As can be seen by comparing Figure 5.25 to Figure 5.15, the cohesiveness of the output has been greatly improved. Comparison of Figure 5.25 to Figure 5.20 shows that the compression of the edge estimates has also been greatly improved. An additional modification can be made to the Clustering-Kalman filtering as we have described it. The output of the Clustering algorithm has some groups which consist of a single element. The objective of the Clustering algorithm is to compress the edge estimates by clustering them into groups which can then be fitted by straight lines. Fitting lines to cluster groups containing few elements is more prone to errors than fitting lines to cluster groups containing many elements. Consequently, singleton groups can be interpreted as consisting of spurious edge estimates which one might not want to use as input to the Kalman filter stage of the algorithm. The results of Figure 5.26 illustrate the output of the Kalman filter bank when the singletons in the Clustering algorithm output have been discarded. As can be seen the results in Figure 5.26 are a bit smoother and less cluttered than those of Figure 5.25. If one accepts the interpretation of singleton groups as spurious edge estimates then the reduction in clutter is no surprise. The Kalman filter bank algorithm avoids tracking edges which belong to no global edge. On the other hand, it is possible to have singleton groups which correspond to legitimate edges. An example of this would be a local edge located at the knee of a sharply curving global line. In this situation, the local edge is an important part of a global line which one would not want to discard since it would help to maintain the cohesiveness of the global line in the Kalman filtering algorithm. The loss of an important singleton may be overcome by increasing the value of h, the prediction only parameter. As will be seen in Chapter 6 the discarding of singletons is a useful modification.

There is an additional post-processing technique that is discussed in this section. The results in Figure 5.25 can be improved by performing some additional combining of the estimated lines. The procedure looks through the set of lines which are the

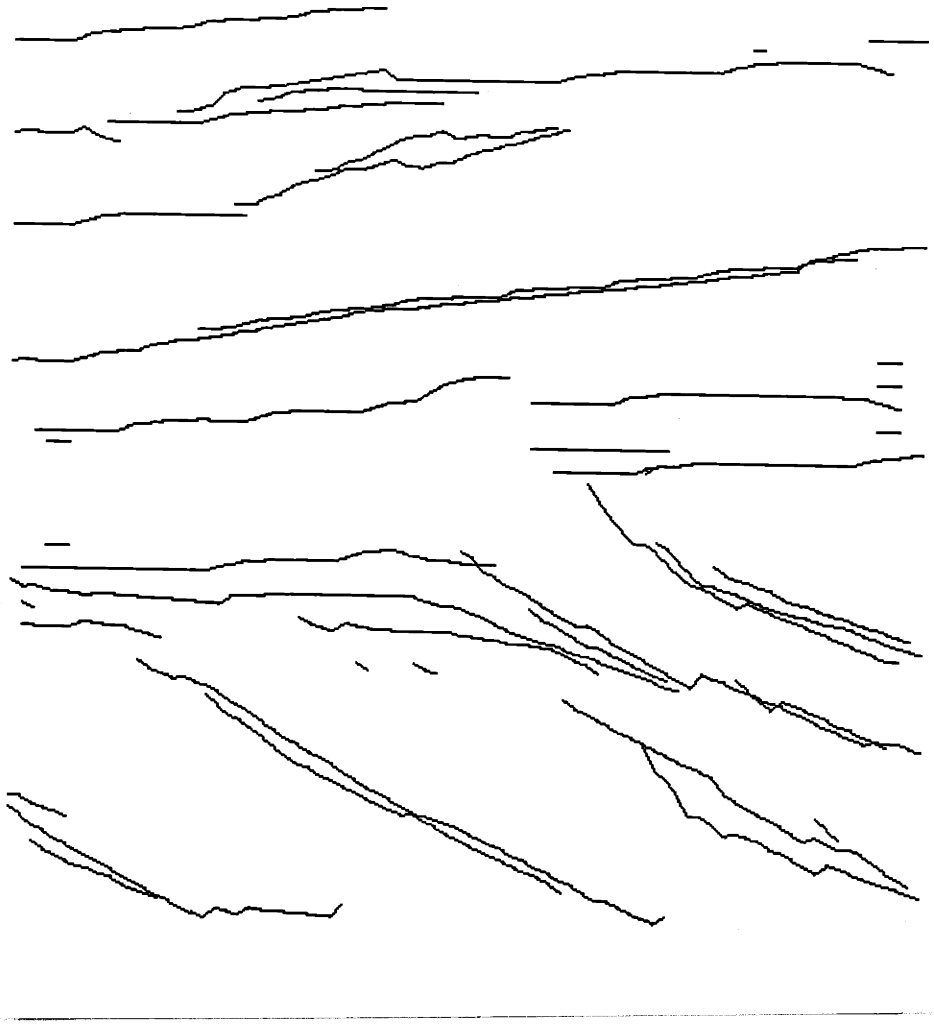

Figure 5.25: Combined Clustering-Kalman Filtering Algorithm $t_e = t_d = .9999$, $q1_k = r1_k = 1.$, $q2_k = r2_k = .5$, and $h = 6$

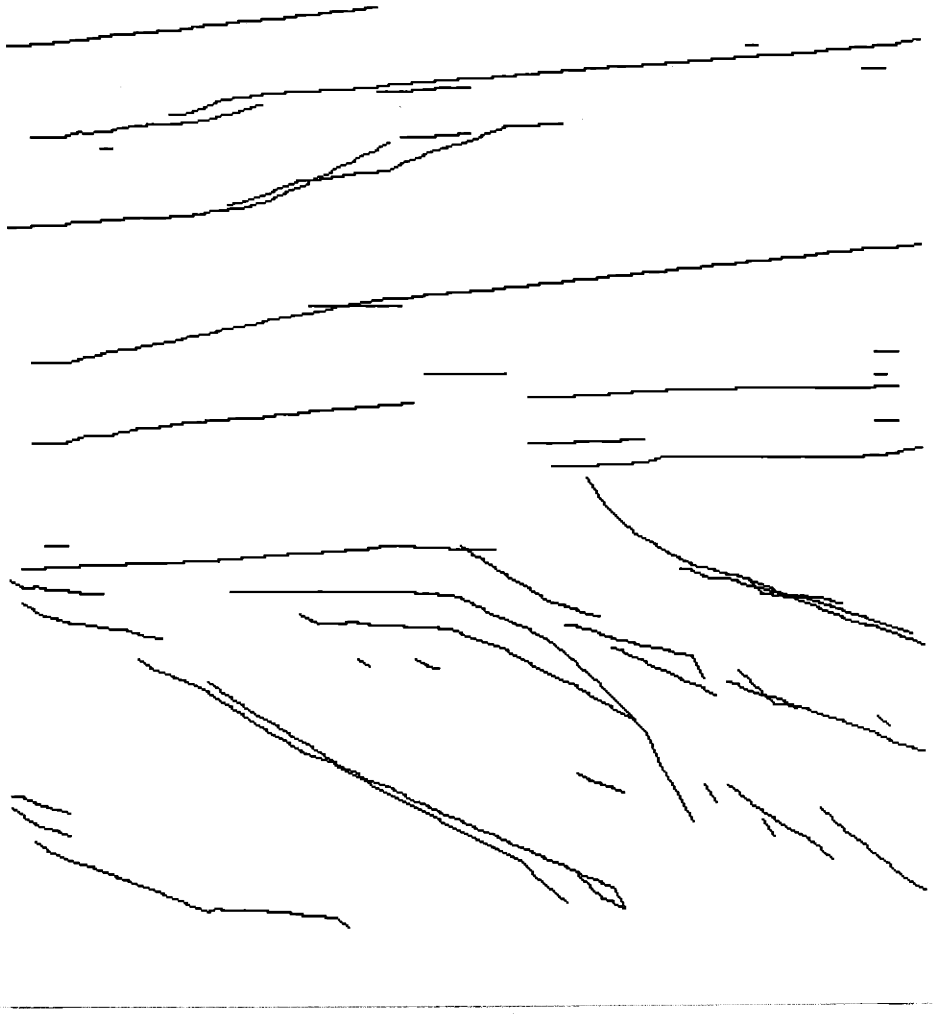

Figure 5.26: Combined Clustering-Kalman Filtering Algorithm $t_e = t_d = .9999$, $q1_k = r1_k = 1.$, $q2_k = r2_k = .5$, and $h = 6$ with Singletons Discarded

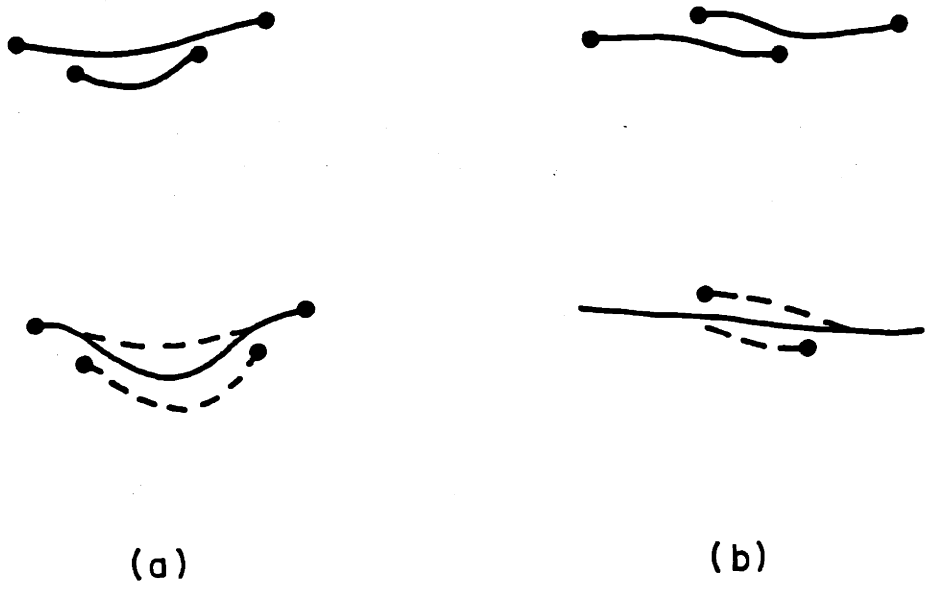

Figure 5.27: Averaging of Overlapping Lines

output of the combined Clustering-Kalman filtering algorithm. For each line, the endpoints are examined to determine whether either endpoint is close to another line. Under two conditions which are illustrated in Figure 5.27 we perform some extra processing. In Figure 5.27a the top portion of the figure illustrates two lines for which the lower line has both of its endpoints near the upper line. In this case the two lines are combined by averaging the two lines between the endpoints of the lower line and attaching this to the remaining segments of the upper line. This produces the new line (a combination of the two old lines) which is illustrated in the lower portion of Figure 5.27b.

In Figure 5.27b a similar processing technique is illustrated. The difference here is that one endpoint of the lower line is close to the upper line whereas the other endpoint of the lower line is not. However, one of the endpoints of the upper line is close to the lower line. Consequently, the procedure retains the left portion of

the lower line and the right portion of the upper line. The middle portion is the average of the upper and lower portions.

Performing this averaging of lines requires an evaluation of the closeness of endpoints to lines. Therefore, a closeness threshold, l_c, must be chosen. Furthermore, since the raw data consists of layered beds (which should have long bed boundaries), short lines are discarded upon obtaining the output of the averaging procedure. The discarding of short lines relies on a line length threshold, l_d.

Figure 5.28 illustrates the averaging procedure applied to the data of Figure 5.25. The thresholds take values $l_c = 1.5$ and $l_d = 4$ where the units are in pixels. As can be seen, a substantial simplification in the global line estimate has been realized.

To close this section one further example is presented. A less complicated data set is used for this purpose. The raw data is illustrated in Figure 5.29. This is real data which consists mainly of beds inclined at essentially the same angle. One of the beds, however, has a non-planar boundary. The remaining beds all have fairly planar boundaries. The beds are of varying thicknesses.

Figures 5.30 and 5.31 illustrate the dip estimate (using the SPROJ algorithm of Chapter 4) and the local edge estimates obtained using the unweighted sum of covariances method. Figure 5.32 is the output of the Clustering-Kalman filtering algorithm described in this section. Finally, Figure 5.33 illustrates the line-averaging technique applied to the data of Figure 5.32.

Notice that throughout the procedure the results are cleaner than those associated with the data set used to generate Figure 5.25. This is due mainly to the clean, well defined nature of the beds in the raw data. The local edge estimates are highly clustered about the bed boundaries making the clustering and Kalman filtering algorithms highly effective.

To conclude this section we present some of the edge estimation techniques discussed in this chapter on a piece of noiseless synthetic data. This data is illustrated in Figure 5.34. The local dip and edge estimates are illustrated in Figures 5.35 and 5.36, respectively. As expected these estimates represent the bedding information in the data very well. Figures 5.37 and 5.38 illustrate the clustering and subsequent Kalman filtering of the local edge data. Finally, Figure 5.39 contains the line-

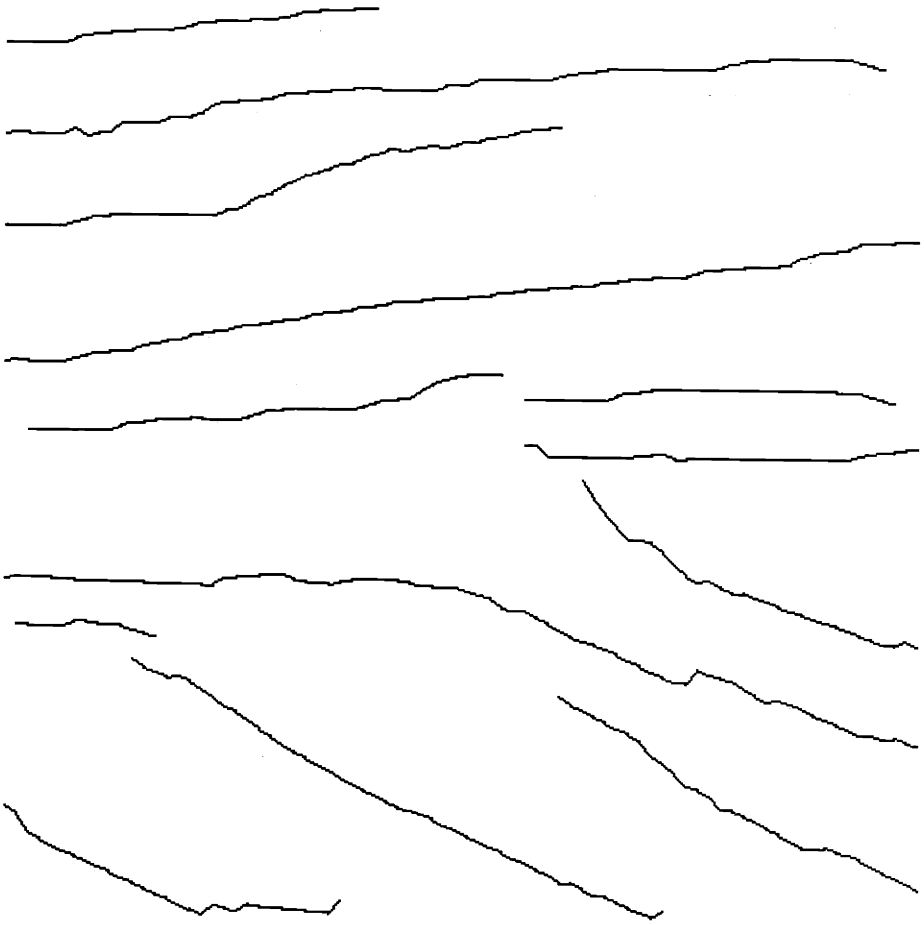

Figure 5.28: Line Averaging $l_c = 1.5$ and $l_d = 4$

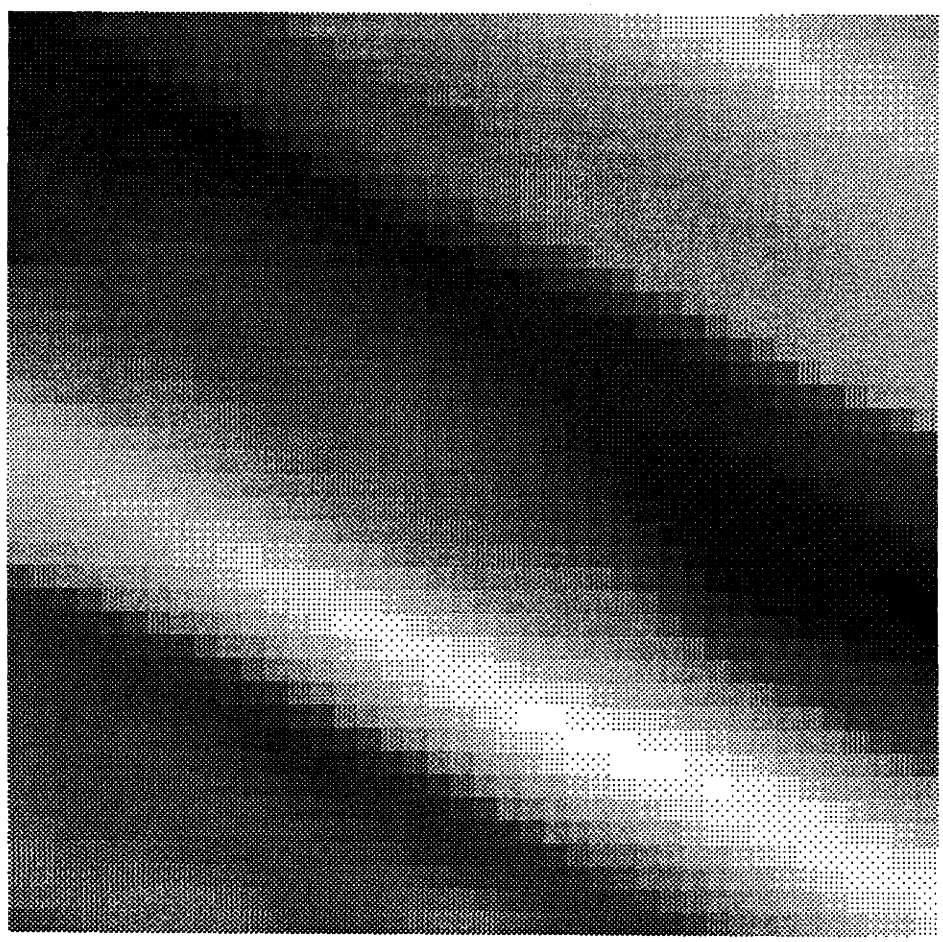

Figure 5.29: Raw Data

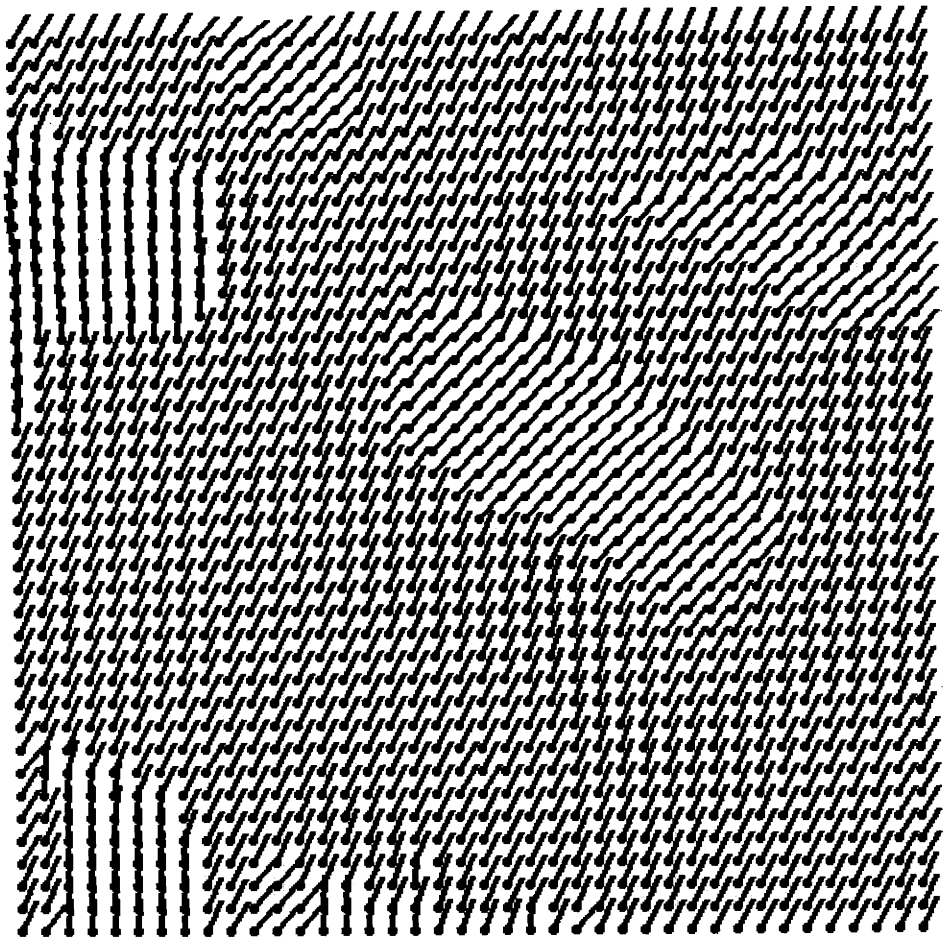

Figure 5.30: Dip Estimates

Figure 5.31: Local Edge Estimates

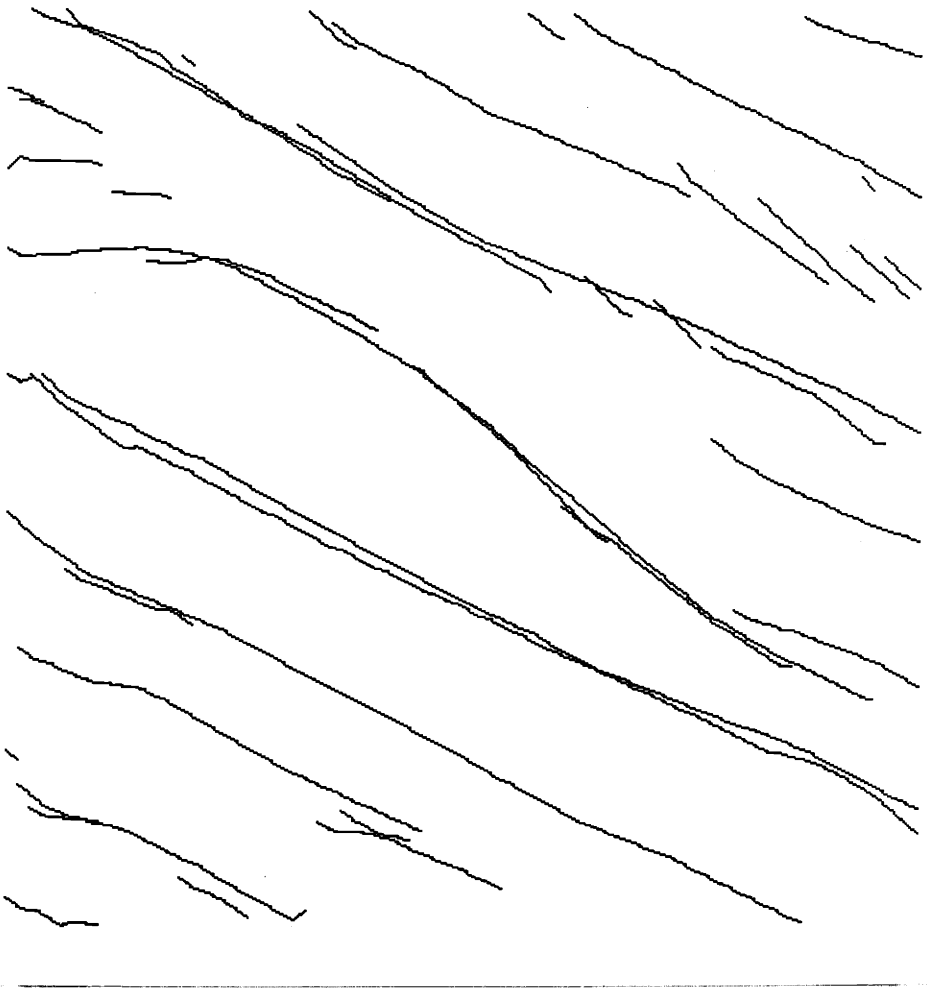

Figure 5.32: Output of Clustering-Kalman Filtering Algorithm

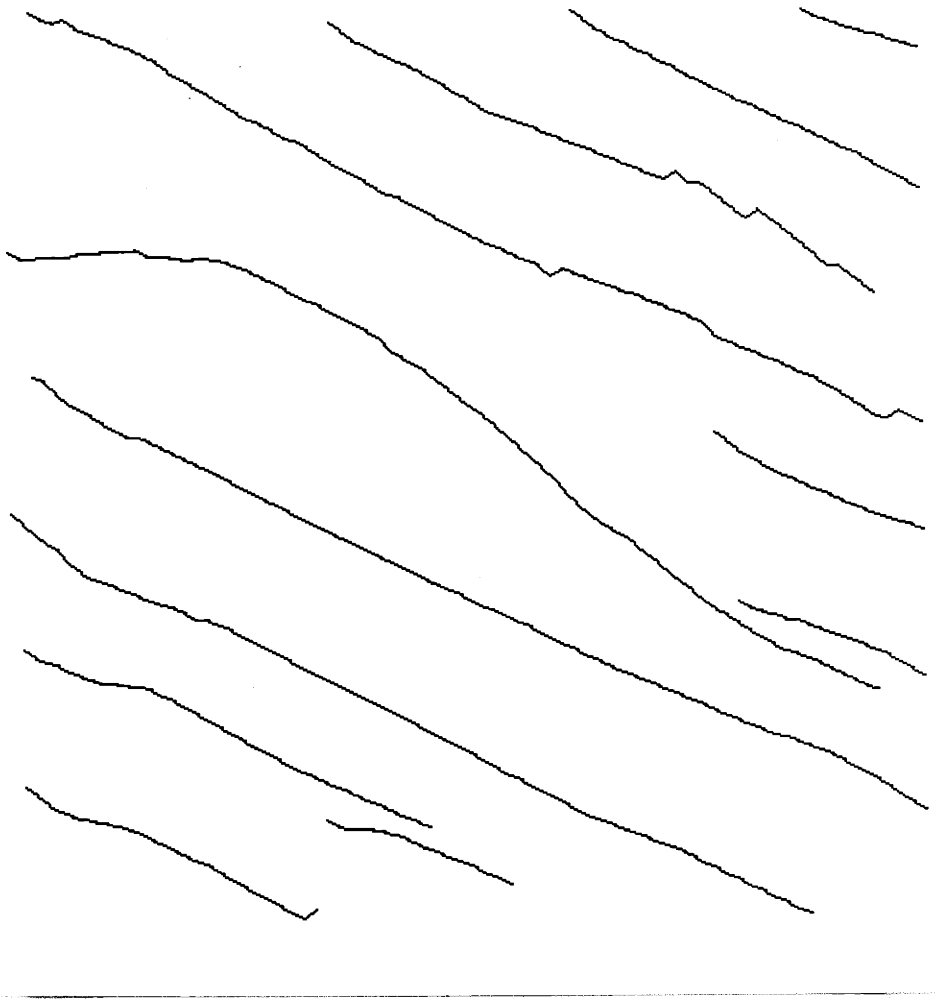

Figure 5.33: Averaged Lines

averaged version of the data in Figure 5.38. As can be seen, when the data is closely related to the model from which our signal processing algorithms are based, the results are very good.

5.9 Conclusions

The objective of this chapter has been to estimate bed boundaries. This has been accomplished by proposing a two-stage edge estimation algorithm. The first stage depends upon a local edge estimation algorithm which is based on an edge model related to the dip models of Chapter 4. The second stage produces global line estimates using the classic methods of clustering and Kalman filtering.

As has been shown in the examples a substantial amount of compression is obtained from the local edge estimates. This compression has been achieved while simultaneously producing cohesive bed boundaries from the raw data. Consequently, the results of this chapter form a stepping stone towards another goal, namely the extraction of higher level features such as patterns in bedding, etc. from the raw data.

5.10 Appendix: Determinant of $X = 1 + \epsilon I$

We show that the $N \times N$ matrix X has determinant $\det(X) = \epsilon^N + N\epsilon^{N-1}$ when

$$X_N = 1 + \epsilon I \tag{5.40}$$

The proof is by induction and requires an auxiliary matrix, Y_N. The matrix Y_N is identical to X_N with the exception of its $(1,1)$ element. This element is unity in the Y_N matrix whereas it is $1 + \epsilon$ in the X_N matrix.

Theorem 3 *The matrix X_N has determinant $\epsilon^N + N\epsilon^{N-1}$ and the matrix Y_N has determinant ϵ^{N-1}.*

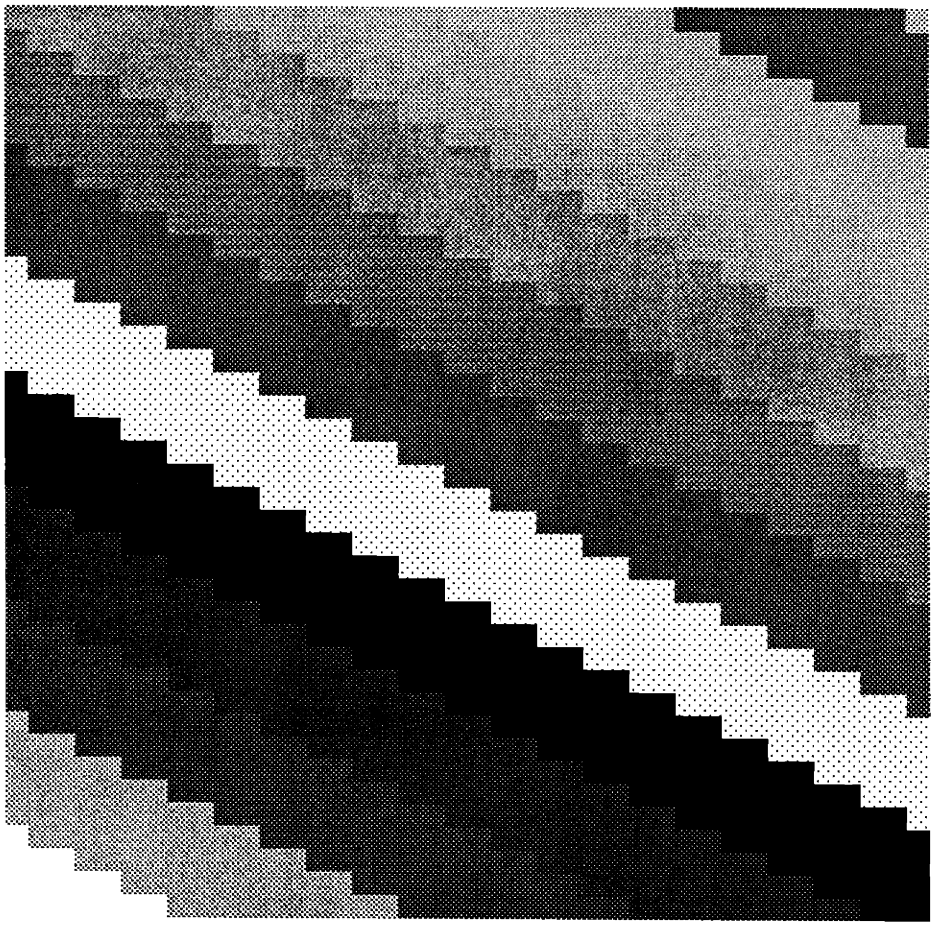

Figure 5.34: Noiseless Synthetic Data

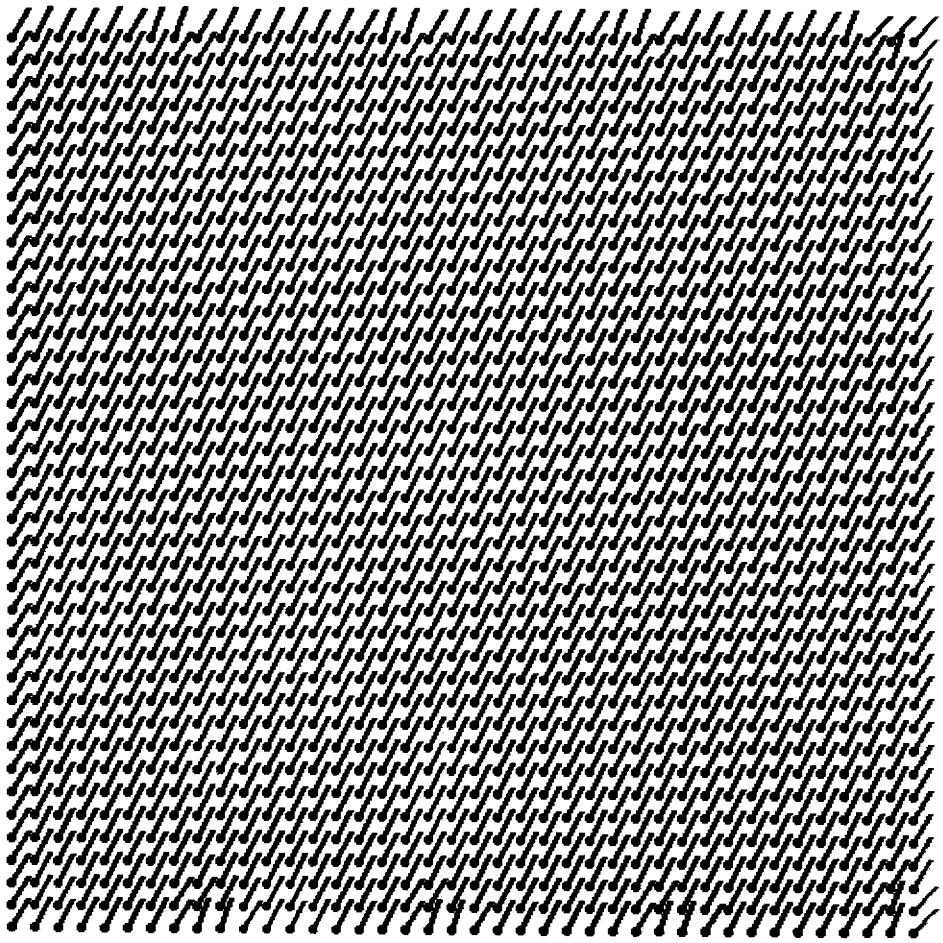

Figure 5.35: Local Dip Estimates

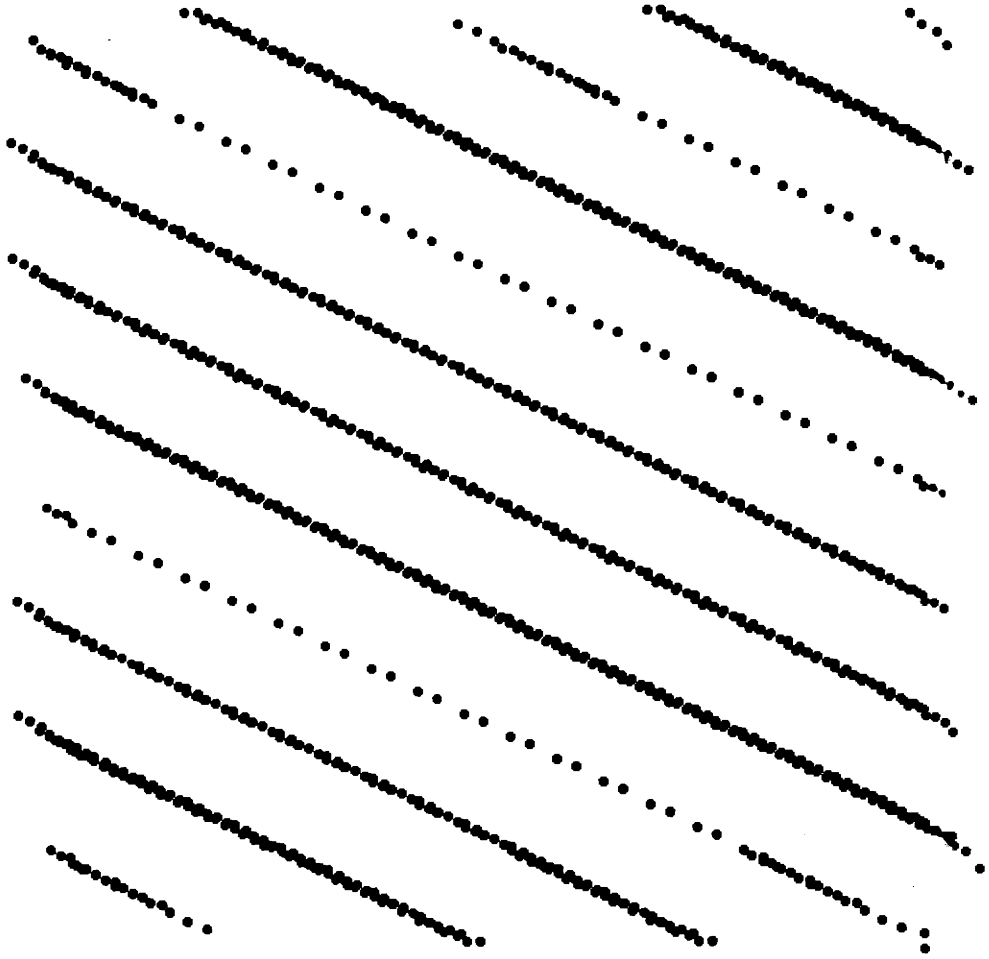

Figure 5.36: Local Edge Estimates

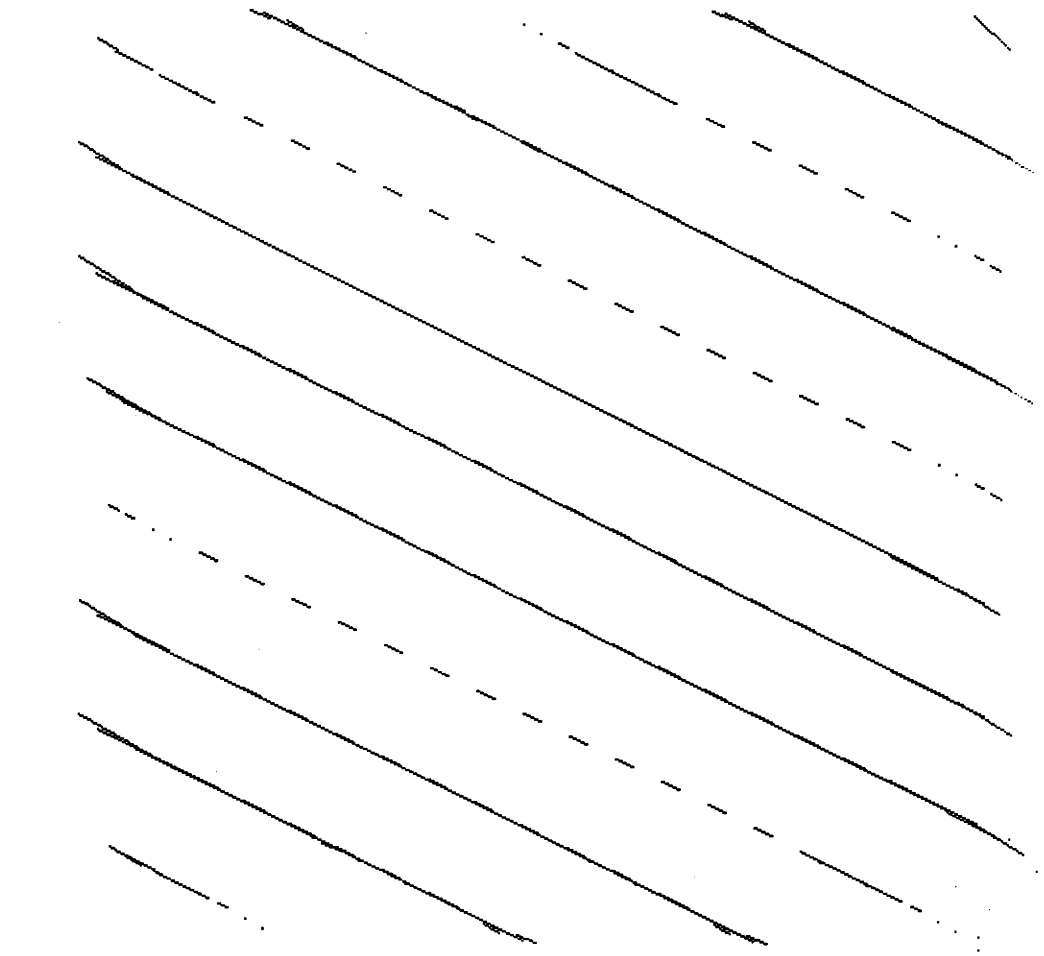

Figure 5.37: Clustering of Local Edge Estimates

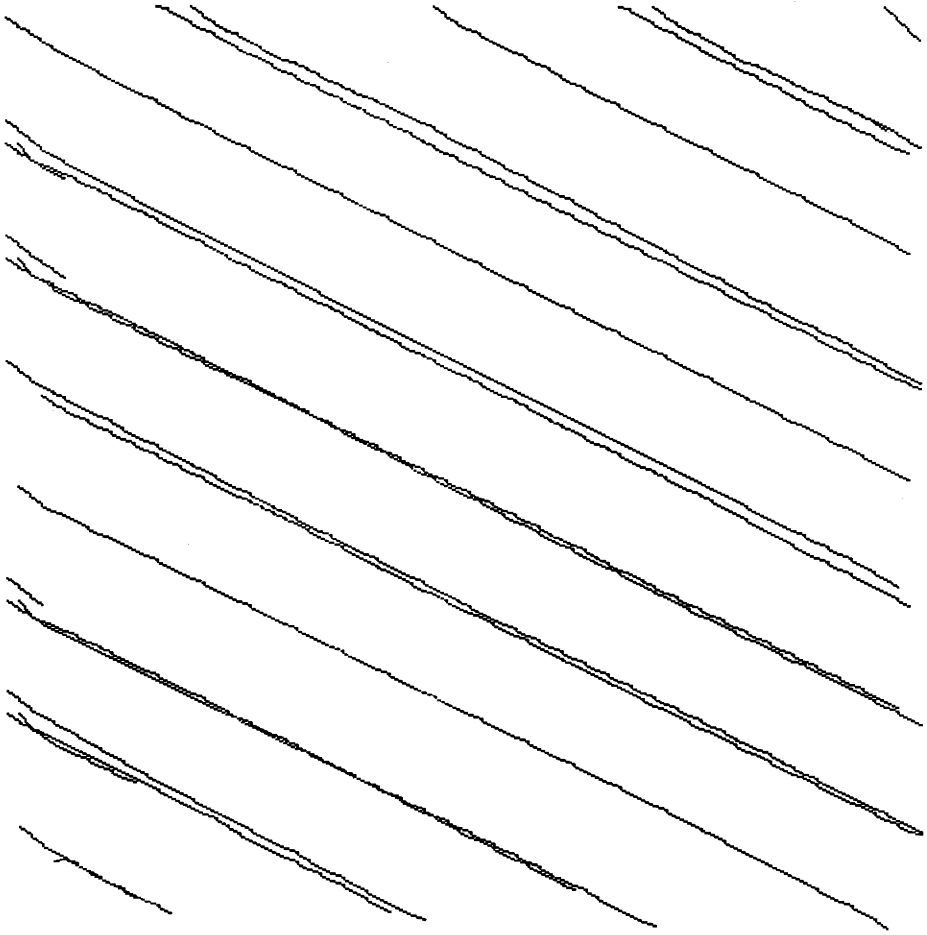

Figure 5.38: Kalman Filtering of Cluster Groups

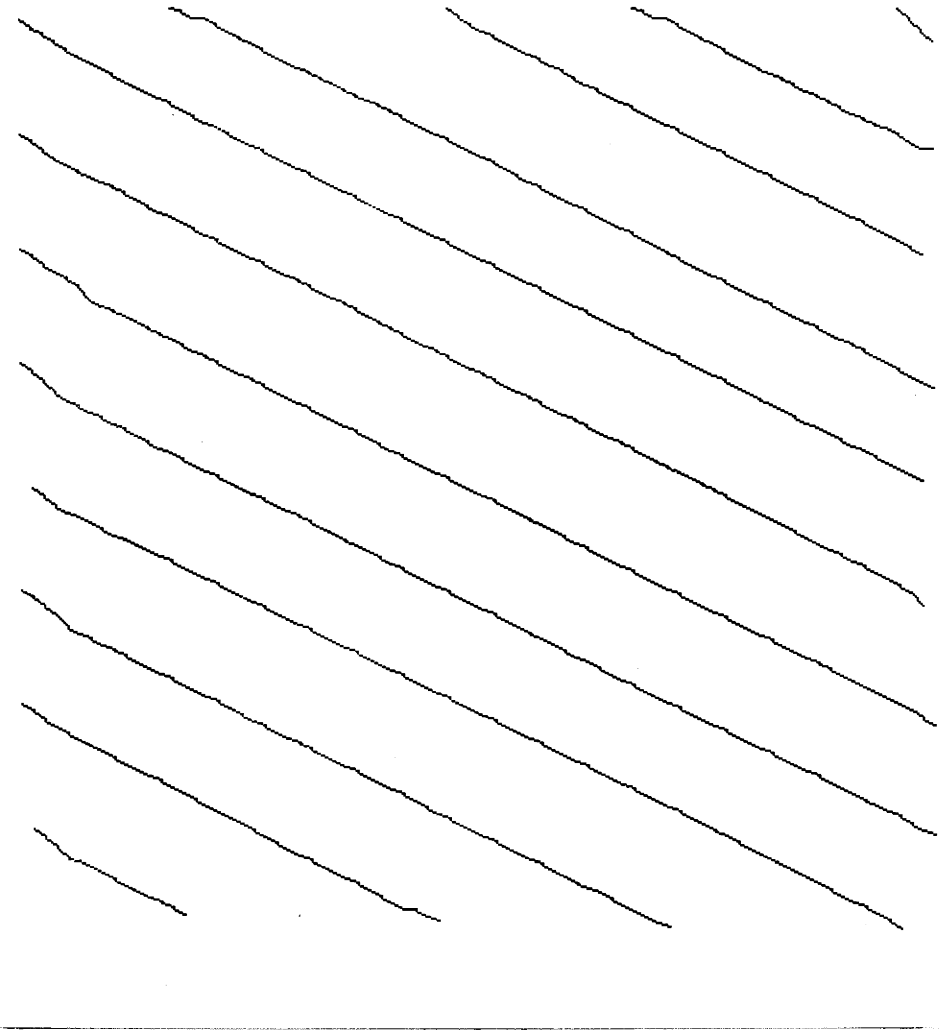

Figure 5.39: Line-Averageing of Kalman Filter Output

Proof 3 *The determinant of the* 2×2 *matrices* X_2 *and* Y_2 *is as follows:*

$$\det \begin{bmatrix} 1+\epsilon & 1 \\ 1 & 1+\epsilon \end{bmatrix} = \epsilon^2 + 2\epsilon$$

$$\det \begin{bmatrix} 1 & 1 \\ 1 & 1+\epsilon \end{bmatrix} = \epsilon \tag{5.41}$$

Assuming that $\det(X_n) = \epsilon^n + n\epsilon^{n-1}$ *and* $\det(Y_n) = \epsilon^{n-1}$ *we examine the determinants of* X_{n+1} *and* Y_{n+1}. *The determinant of* X_{n+1} *is*

$$\det(X_{n+1}) = \det \begin{bmatrix} 1+\epsilon & 1 & \cdots & 1 \\ 1 & 1+\epsilon & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1+\epsilon \end{bmatrix} \tag{5.42}$$

Expanding the determinant of X_{n+1} *by cofactors along the first column we obtain that*

$$\det(X_{n+1}) = (1+\epsilon)\det(X_n) - n\det(Y_n) \tag{5.43}$$

The determinant of Y_{n+1} *is*

$$\det(Y_{n+1}) = \det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1+\epsilon & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1+\epsilon \end{bmatrix} \tag{5.44}$$

Again, expanding by cofactors yields

$$\det(Y_{n+1}) = \det(X_n) - n\det(Y_n) \tag{5.45}$$

Substitution of the assumed values for X_n *and* Y_n *into (5.43) and (5.45) yields*

$$\begin{aligned} \det(X_{n+1}) &= (1+\epsilon)(\epsilon^n + n\epsilon^{n-1}) - n\epsilon^{n-1} \\ &= \epsilon^{n+1} + (n+1)\epsilon^n \end{aligned} \tag{5.46}$$

and

$$\begin{aligned} \det(Y_{n+1}) &= (\epsilon^n + n\epsilon^{n-1}) - n\epsilon^{n-1} \\ &= \epsilon^n \end{aligned} \tag{5.47}$$

Consequently, the theorem is proved.

Chapter 6

Analysis of Data Features With More Than One Scale

In the preceding chapters much attention has been given to techniques which have relied on specifying a neighborhood set or window size. These specifications are intimately related to the spatial scale at which our analysis can proceed. For features which occur at large spatial scales small neighborhoods or windows produce erratic results. However, increasing the size of these windows dramatically increases the computational burden of our methods.

This chapter is a case study centered around a piece of synthetic data. The synthetic data has features at two scales and is observed in additive white noise. The objective of the study is to show how to separate the two scales of data and effectively analyze the data at each scale utilizing some of the techniques presented in Chapter 4 and 5

The chapter is composed of the following sections. Section 6.1 presents the synthetic data set used in the study. Section 6.2 discusses block averaging as a means of analyzing the large scale features of the data. Section 6.3 discusses a method of identifying regions in the data of smaller scale. Section 6.4 analyzes a small scale segment of data. Section 6.5 uses the techniques discussed in this chapter on some real data and Section 6.6 concludes this chapter.

6.1 The Synthetic Data

The synthetic data set used in this chapter has spatial features at two scales. The data consists of a 160×160 array of pixels. This is in comparison to data sets of 40×40 arrays of pixels analyzed previously in this thesis. The large scale features in the data are composed of two bedded regions as is illustrated in Figure 6.1.

The upper portion of the large scale features have beds which take values $\pm\frac{1}{2}$. The slope of the bed boundaries is $\frac{1}{2}$. The lower portion of the large scale features take values ± 1 and have bed boundaries with slope -2. The bed thickness for both the upper and lower portions of the large scale features is approximately 36 pixels.

The small scale features are contained in a bed in the upper portion of the data (see Figure 6.2). These features consist of beds with alternating relative contrast values of $\pm.1$ (i.e. these values are added to the large scale feature values) and with boundaries having a $-\frac{1}{2}$ slope. The thickness of the small scale beds is approximately 5.4 pixels.

The observations of the synthetic data are made in the presence of additive, uniformly distributed, white noise. The signal to noise ratio between the small scale features and the noise is 20db and the ratio in power between the large and small scale features is also 20db. The synthetic data, with additive noise included, is illustrated in Figure 6.3.

6.2 Block Averaging for Extracting Large Scale Features

Our intention is to perform a hierarchical analysis of the data in Figure 6.3. That is, we first analyze the large scale features of the data followed by an analysis of the small scale features. To analyze the large scale features of the data using the methods already discussed in this thesis requires a method for re-scaling the data. This is accomplished in this section by block averaging the data.

The examples presented in Chapters 3, 4, and 5 of this thesis have all been executed on 40×40 arrays of data. Furthermore, the features in the data have been

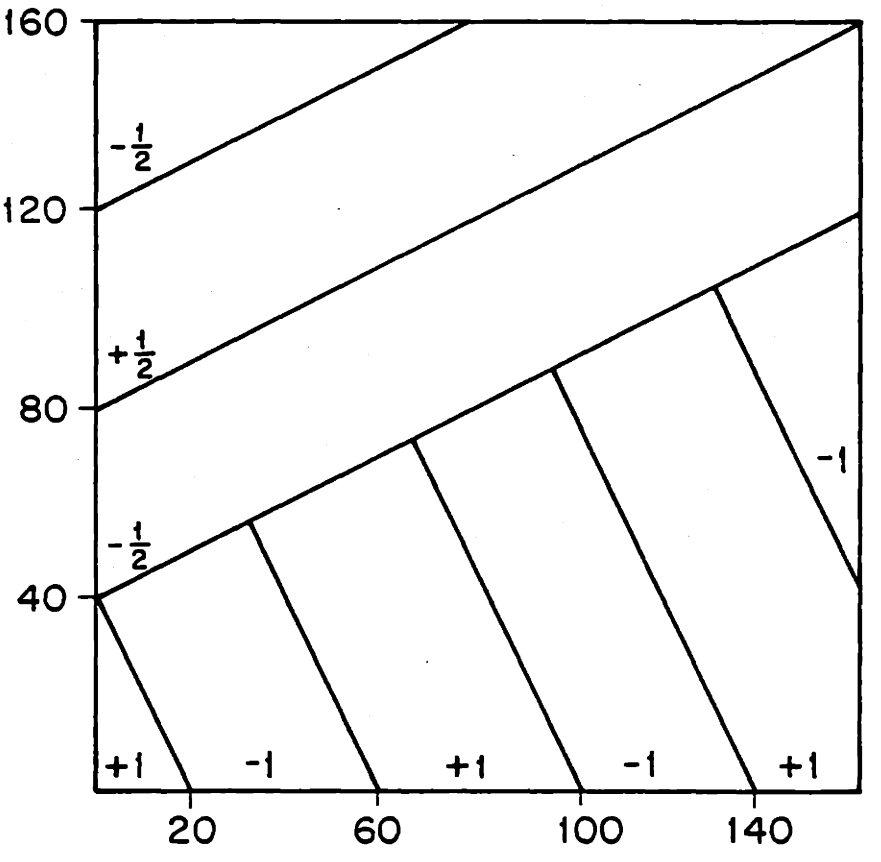

Figure 6.1: Large Scale Features

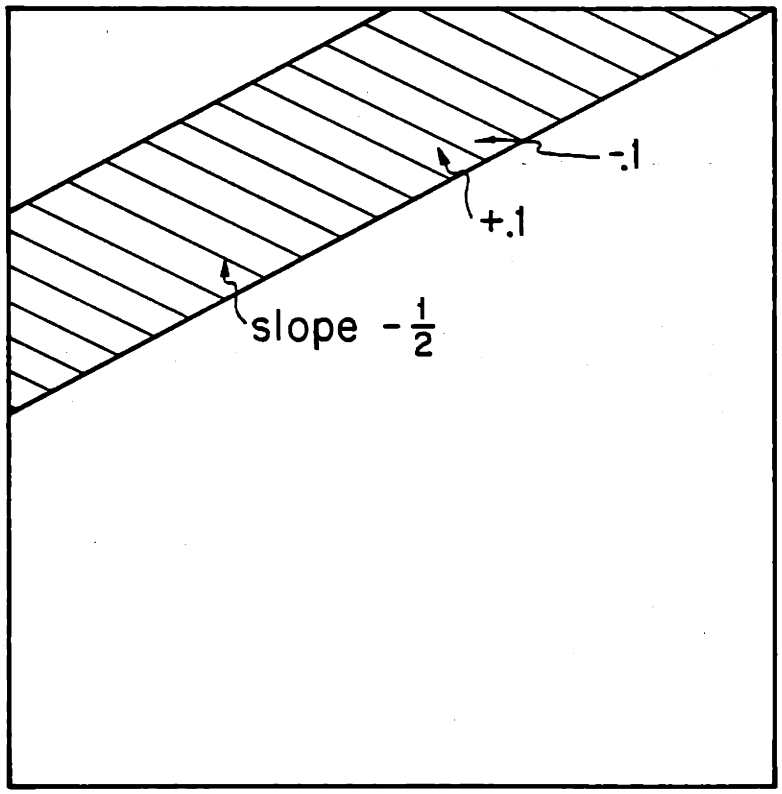

Figure 6.2: Small Scale Features

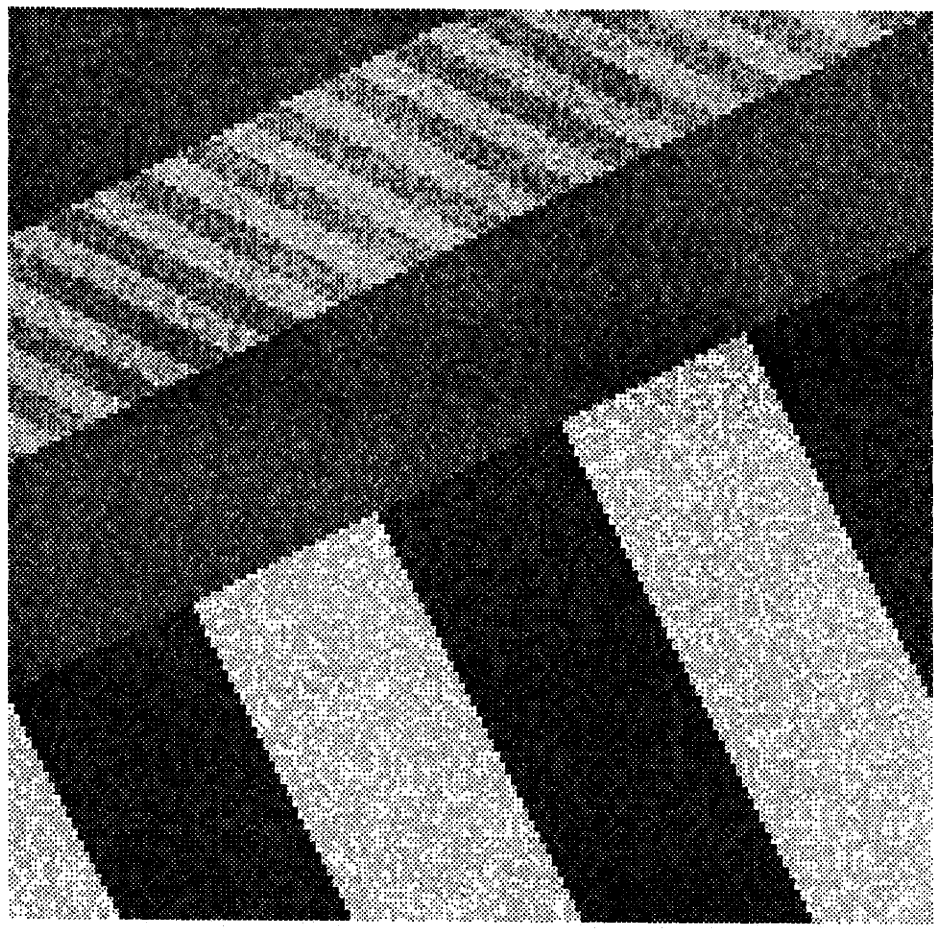

Figure 6.3: Synthetic Data With Features at Two Scales

such that our techniques have been effective when using 7×7 windows of data for our neighborhoods. Since the large scale beds in Figure 6.3 have bed thicknesses of over 35 pixels it should be clear that a 7×7 window will not be effective in analyzing this data set. Increasing the window size is undesirable since this increases the computational burden of our algorithms.

Block averaging of the data, however, serves our goals well. First, block averaging compresses the large scale data to a scale which can be handled by our 7×7 window. Second, the small scale features are attenuated by the block averaging so that they should essentially be ignored by this stage of the signal processing.

The block averaging procedure is performed as follows. The data is partitioned into smaller blocks of data. The data within a block is averaged and then a new data set is created where each block of the data is mapped to a single pixel of the new data set. The pixel value in the new data set is the average of the block pixels values in the old data set.

Figure 6.4 illustrates the block averaging procedure performed on the data set of Figure 6.3. In this example the blocks are 4×4 arrays of pixels. Consequently, since the data in Figure 6.3 is a 160×160 array, the data of Figure 6.4 is a 40×40 array of pixels. The bed thickness of the large scale features is now approximately 9 pixels and the small scale features have beds which are less than 1.5 pixels thick. Consequently, the small scale beds have been attenuated, however, some of the correlation structure can still be seen in the block averaged image.

The first part of our analysis of the block averaged data in Figure 6.4 is to estimate the local dip. Using the SPROJ algorithm of Chapter 4 with a 7×7 moving window, the result of estimating the local dip is illustrated in Figure 6.5.

The dips in Figure 6.5 mainly follow our expectations. However, there are regions of anomalous dip estimates caused by one of three types of situations. The first situation is illustrated in Figure 6.6a. In this figure a 7×7 window is illustrated in the lower portion of the block averaged data. As the illustration shows, the window is totally contained in a single bed of the data. Consequently, the dip estimate associated with the pixel at the center of this window is likely to be erratic since there is no bed boundary to indicate the direction of maximum correlation.

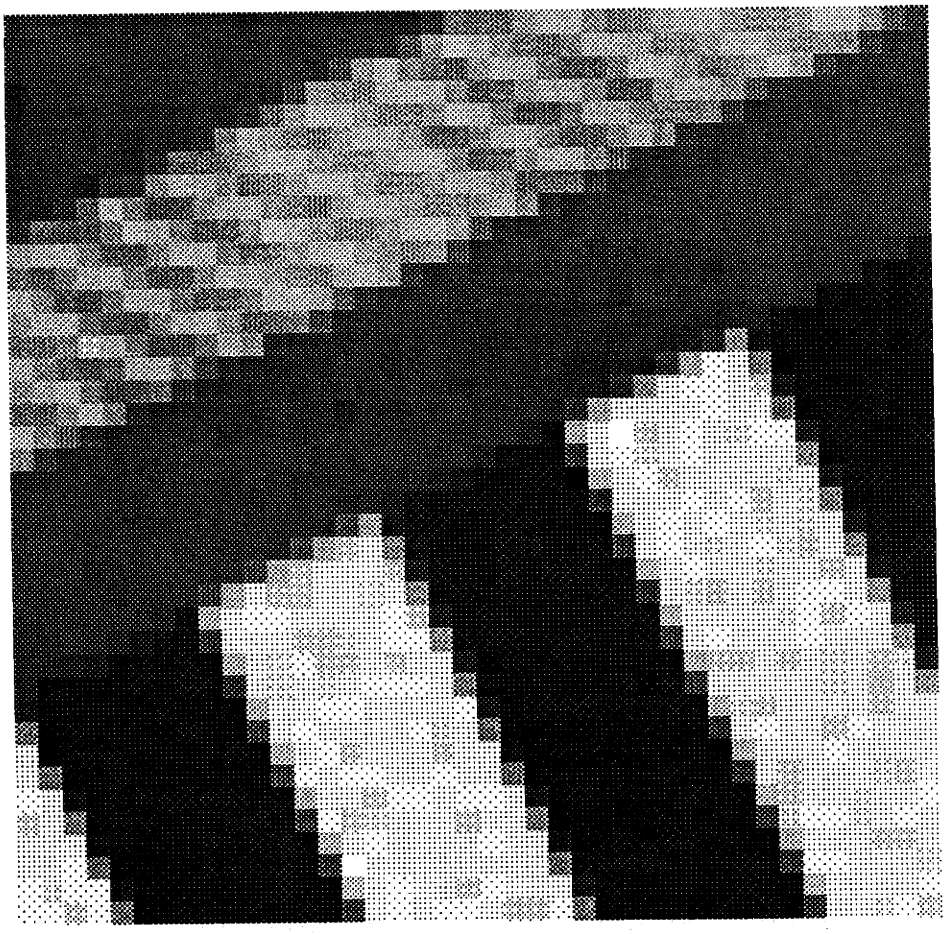

Figure 6.4: Block Averaged Date with 4×4 Blocks

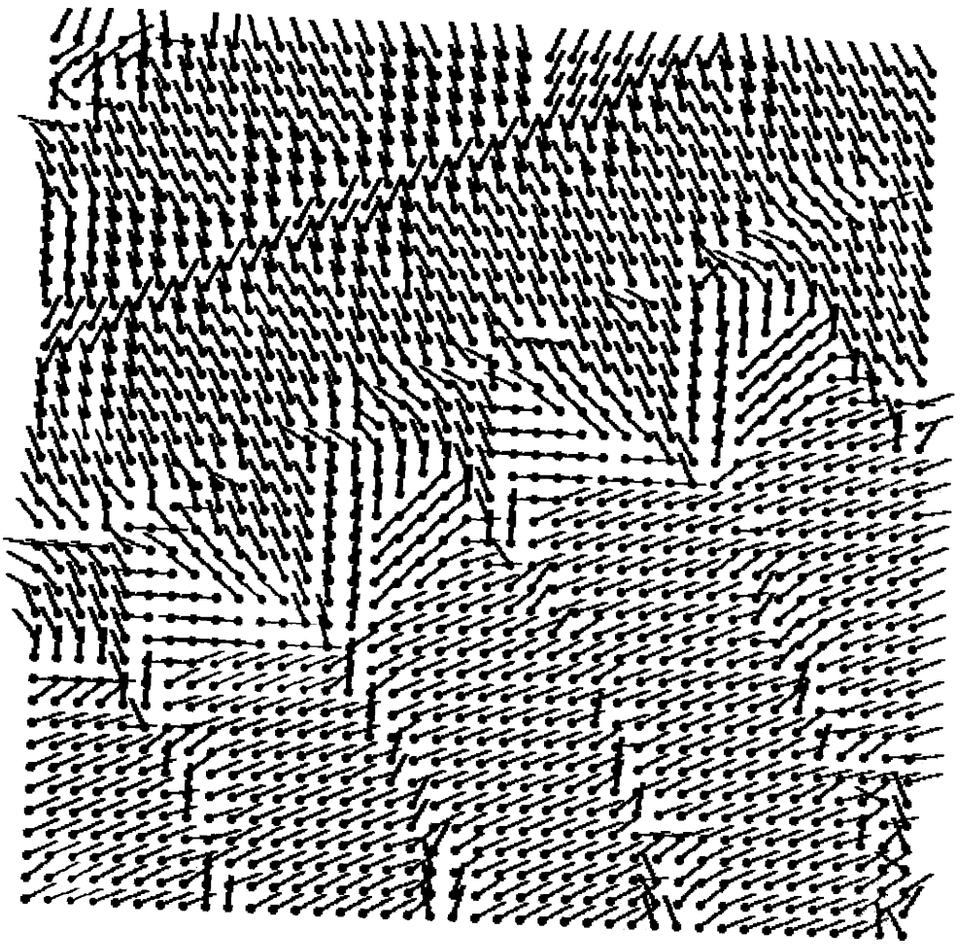

Figure 6.5: SPROJ Dip Estimate

This behavior is seen along the center of beds in Figure 6.5.

Figure 6.6b is similar to Figure 6.6a in that the 7×7 window is totally contained by a large scale bed in the upper portion of the block averaged synthetic data. However, a careful examination of Figure 6.4 reveals that the bed with the attenuated small scale features still contains the information concerning the dip of these small scale beds. This is in accordance with our previous observation concerning the block averaging of the data in this bed. Consequently, the dip estimates associated with windows which are contained in this large scale bed all reflect the correct dip value for the small scale beds. The block averaging procedure is intended to reduce the scale of the large scale features and to remove the small scale features. In this case, the amount of block averaging is not quite enough to fully accomplish these goals but under the circumstances is adequate, as we shall see.

The final region type is illustrated in Figure 6.6c. Here the 7×7 window is located at a junction of the upper and lower portions of the block averaged synthetic data. It should be clear that the estimated dip associated to this window will be biased since the data within the window does not follow any of the models proposed within this thesis. This conclusion is born out in the dip estimate of Figure 6.5. This characteristic can also be seen in the many dip estimates made on the data of Figure 4.4 which displays similar characteristics.

Figure 6.7 is an illustration of the associated local edge estimates for the block averaged data in Figure 6.4. These local edge estimates cluster strongly about the bed boundaries of the large scale features in the data. So, although the dip estimates reflect some of the small scale features of the block averaged data, the edge estimates do not. This is because the edge estimate window almost always contains a portion of a large scale bed boundary. Since the contrast of the large scale beds is much greater than that of the small scale beds, the local edge estimates are dominated by these boundaries. The most serious scatter occurs at the corners of beds formed where the upper and lower portions of the data meet.

Figure 6.8 shows the clustering algorithm applied to the local edge estimates in Figure 6.7 with parameters $t_e = t_d = .9999$. As expected, the major clusters produce line segments following the bed boundaries of the large scale features.

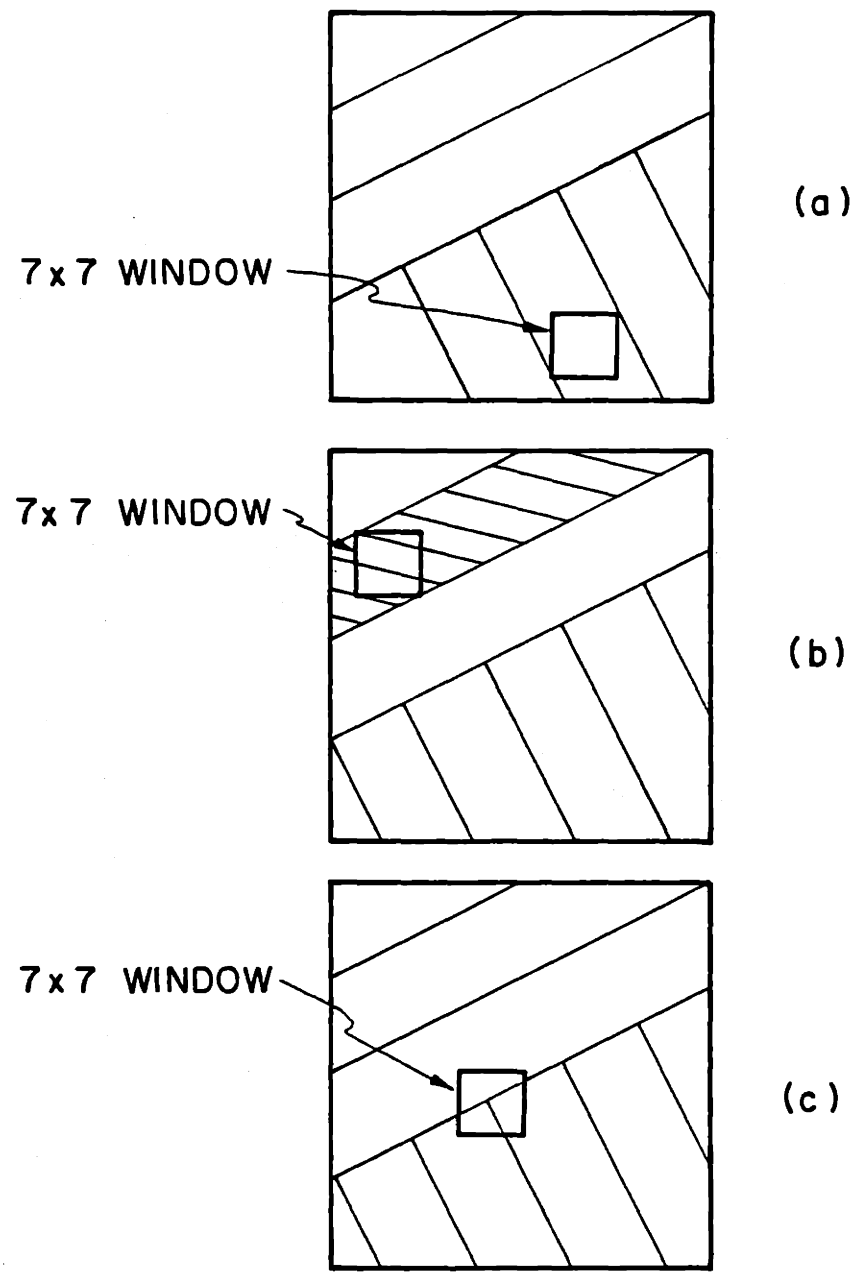

Figure 6.6: Anomolies in Dip Estimates

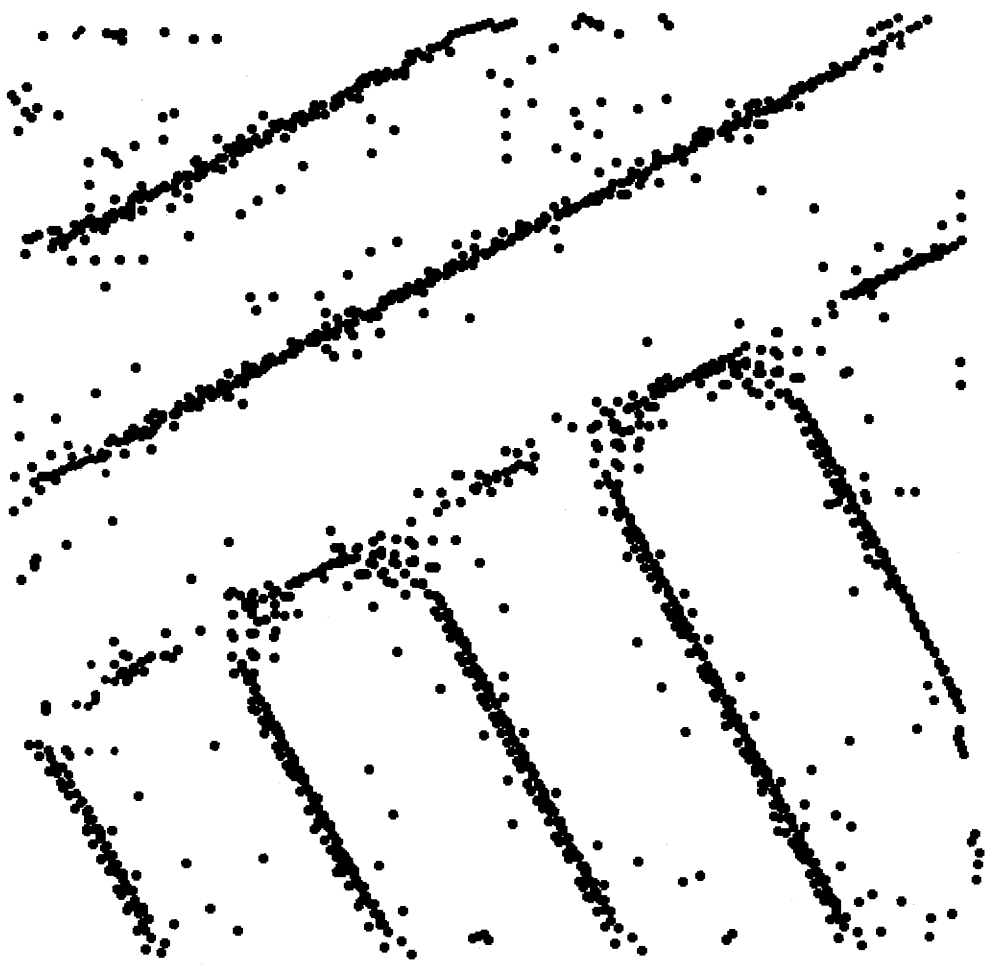

Figure 6.7: Local Edge Estimates

There are, however, several small clusters in the region of the small scale features which follow the small scale bed boundaries. This is significant when the Kalman filtering algorithm is applied.

Figure 6.9 is the result of applying the Kalman filter bank algorithm to the clustered data in Figure 6.8 The singletons in Figure 6.8 (the clusters consisting of a single point) adversely affect the performance of the Kalman filter algorithm. As can be seen in Figure 6.9, the filters attempt to track edges away from the main clusters. This divergence is due to the singletons. Figure 6.9 should be contrasted with Figure 6.10 which illustrates the use of the Kalman filtering algorithm on the output of the Clustering algorithm with the singletons discarded. As can be seen, the Kalman filter output is much cleaner in that the edge tracking is closer to the actual bed boundaries of the data. This illustration verifies the comments we made in Section 5.8 concerning singletons and their effects on the Kalman filtering algorithm.

Finally, Figure 6.11 is a processed version of Figure 6.10. Here we have combined Kalman filtered lines and we have discarded short lines. The threshold parameters were $l_c = 8$ and $l_d = 3$. As can be seen the edges follow bed boundaries well. The most trouble is found along the boundary between the upper and lower portions of the large scale features. Also, an edge associated with the bed boundaries in the small scale features is apparent in the upper portion of the figure.

6.3 Identifying Regions With Small Scale Features

As stated at the beginning of this chapter, the objective here is to investigate a hierarchical approach to analyzing the data. This approach is one in which we first analyze the large scale features and then analyze the small scale features. To accomplish the second task we require a method for identifying the data regions with small scale features. In this section a technique for identifying such regions is discussed.

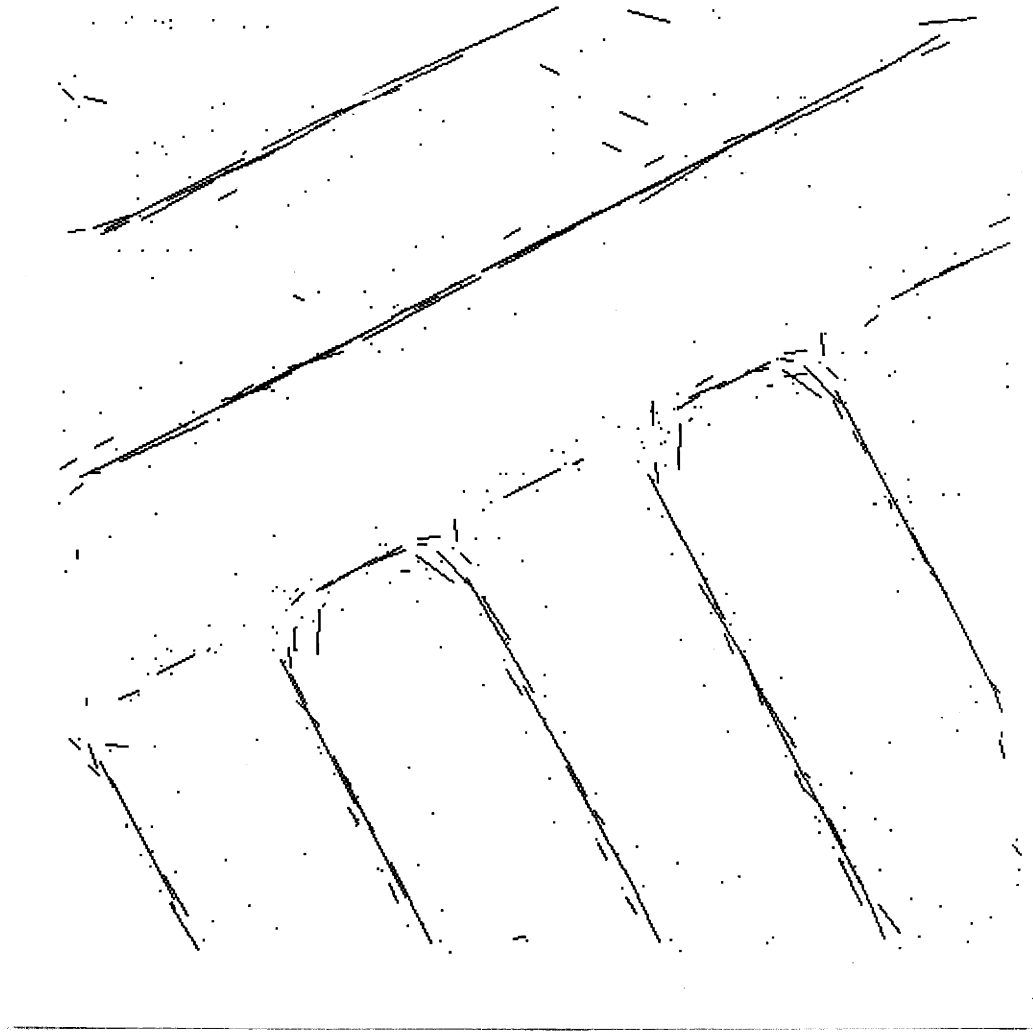

Figure 6.8: Clustering Algorithm $t_e = t_d = .9999$

Figure 6.9: Kalman Filtering of Clustered Data

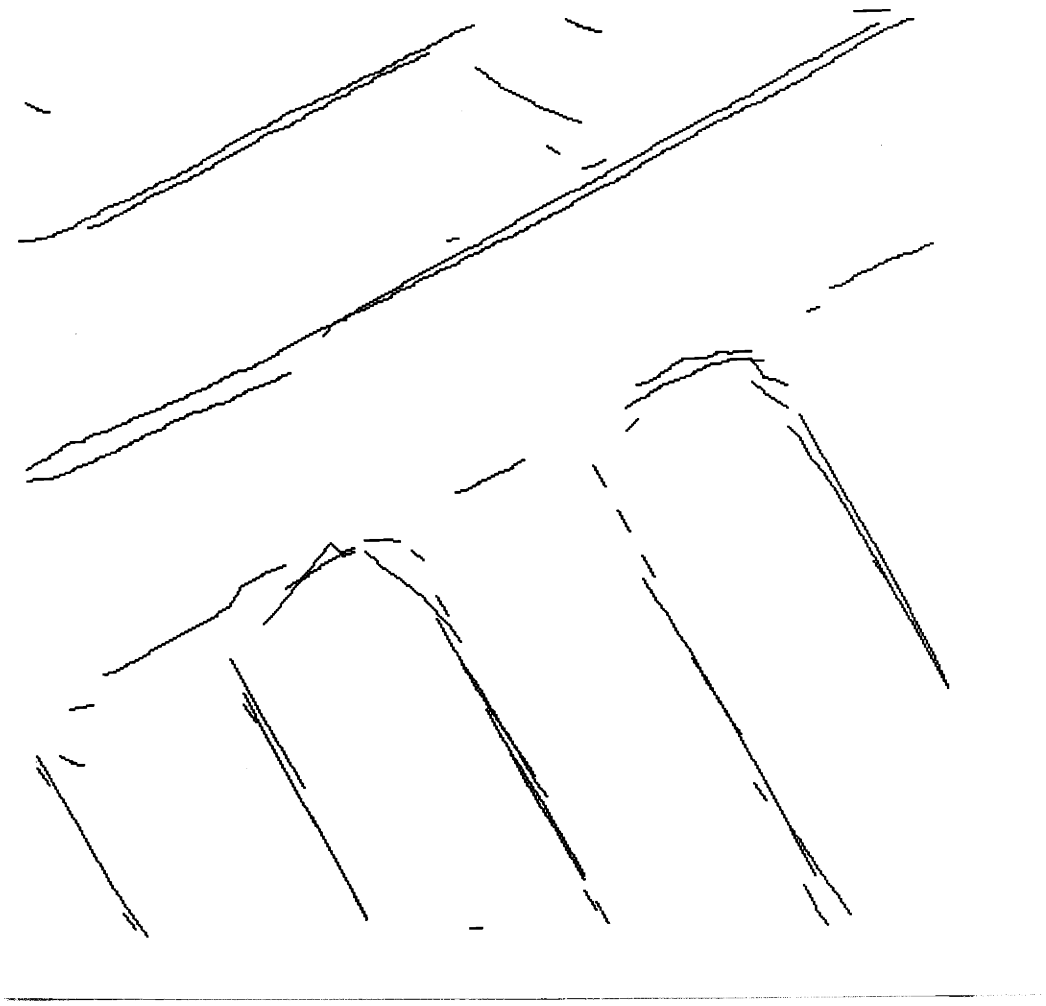

Figure 6.10: Kalman Filtering of Clustered Data With Discarded Singletons

The technique discussed in this section is a derivative of the block averaging procedure discussed in Section 6.2. The objective of the block averaging is to reduce the scale of the large scale features and to "wash-out" the small scale features. However, contained in each block of data is information about the small scale bedding features.

One method for extracting the small scale bedding information is by calculating the block variance of the data as well as the block average. In large scale bedded data which contains no small scale features the block variance is small. However, where small scale bedding features exist the block variance is comparatively much larger. This principle is illustrated in Figure 6.12. The upper 4×4 block of data is in a region which contains small scale features. The middle 4×4 block of data is in a region which contains no small scale features. Clearly the variance of the data in the upper 4×4 block is greater than that in the middle 4×4 block.

There is a third type of 4×4 block of data in Figure 6.12. The lowest 4×4 block indicated in the figure straddles two large scale beds. Consequently, the variance of this bed is greater than the variance of a block totally contained within a large scale bed. It is our intention to locate regions of the data with small scale features by finding the high variance blocks of data. Consequently, the blocks which straddle large scale beds will have misleading variances.

We can circumvent the above problem by utilizing our knowledge of the large scale bed boundaries obtained as described in Section 6.2. For each block variance computation the procedure determines whether one of the estimated bed boundaries intersects this block. If the block is intersected by an estimated line then not all the data within the block is used to calculated the block variance. As illustrated in Figure 6.13, only data points that are on the same side of the estimated boundary are included in the block variance computation.

Figure 6.14 illustrates this block variance computation when performed on the data of Figure 6.3. The dark pixels in Figure 6.14 represent low variances and the light pixels represent high variances. As can be seen the region of the small scale bed features has a high variance compared to the regions without small scale structure. Some of the bed boundaries of the large scale structure can also be seen.

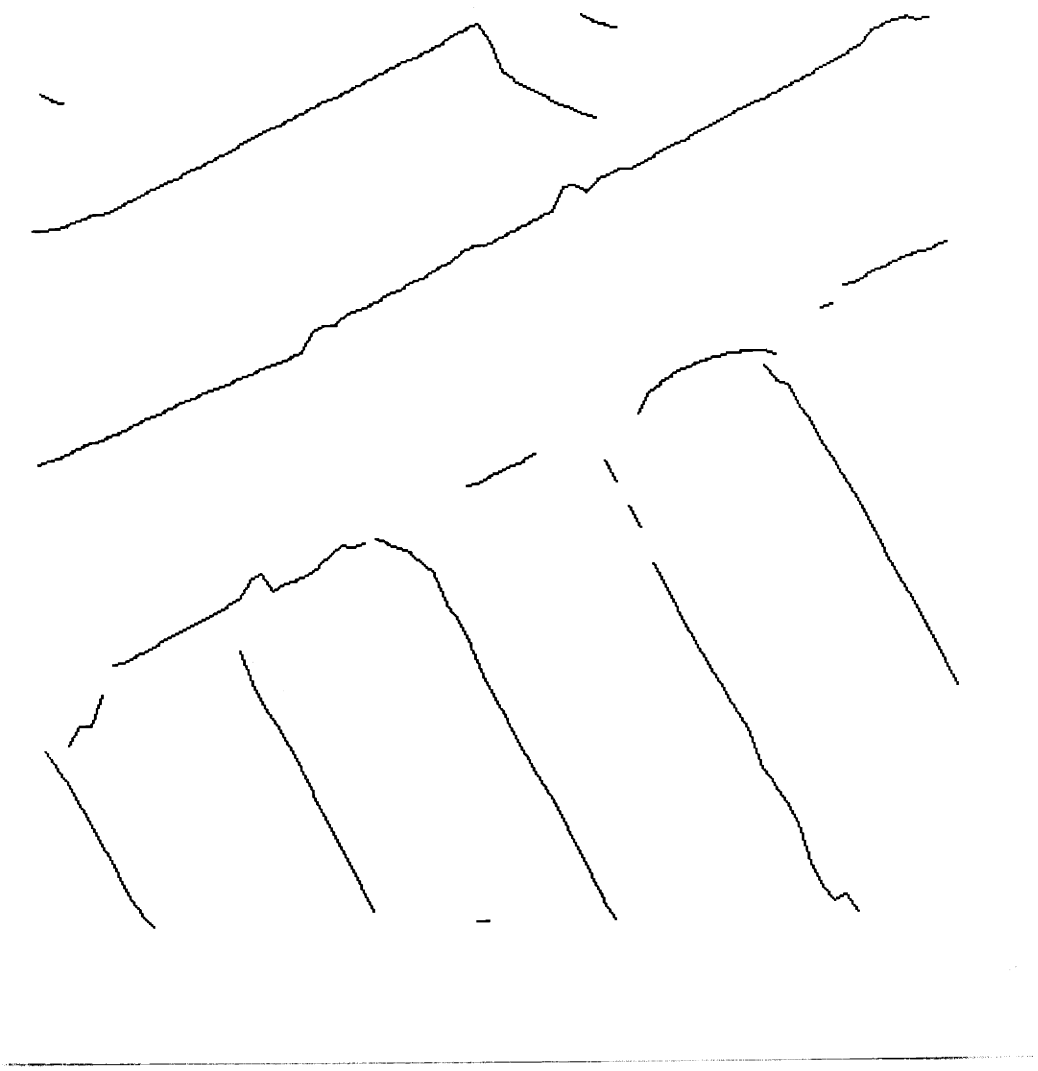

Figure 6.11: Output of Clustering-Kalman Filtering Algorithm With Line Averaging

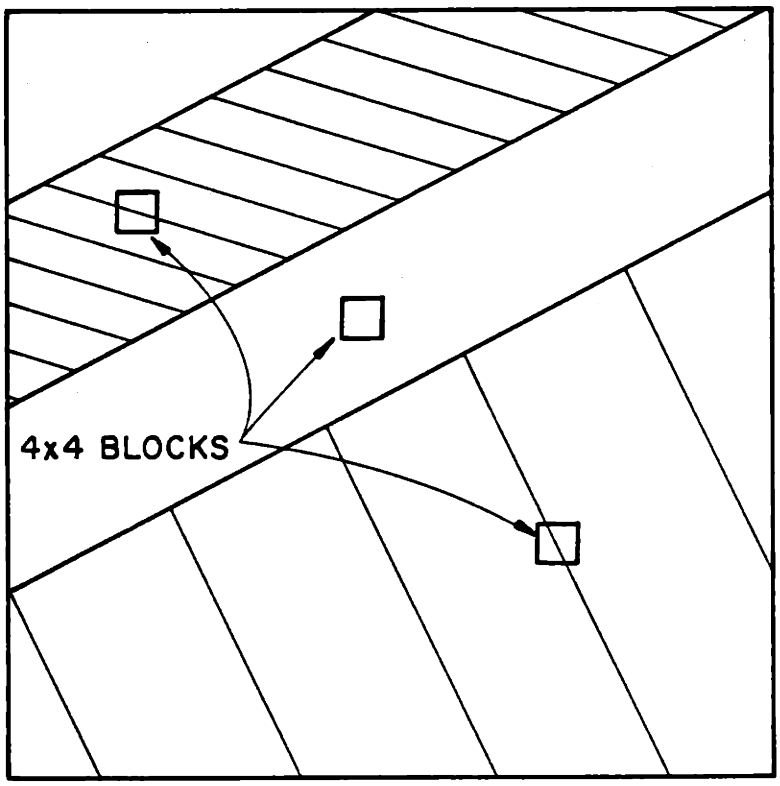

Figure 6.12: Block Variance Computations

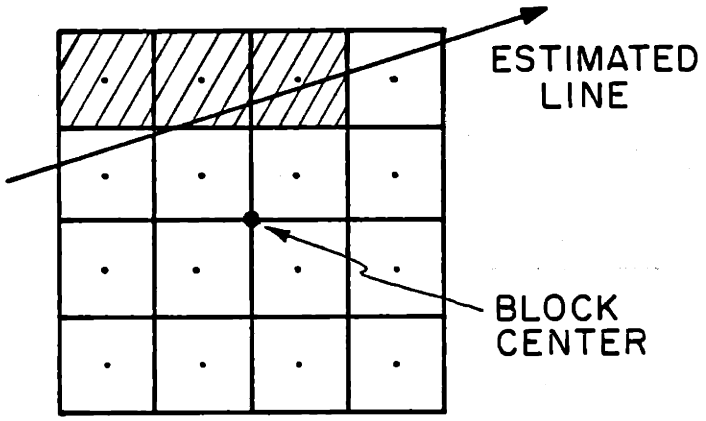

Figure 6.13: Cross-Hatched Pixels Not Included In Block Variance Computation

This is due to errors in our estimates of bed boundary locations. However, these boundaries occupy extremely narrow regions incompatible with finding small scale structure.

6.4 Analyzing Small Scale Features

The previous section detailed a method for finding regions with small scale bedding structure. In this section we analyze the small scale structure of a segment of the data in Figure 6.3. For this purpose we examine a 40×40 block of the data in Figure 6.3. This block of data is illustrated in Figure 6.15. The analysis proceeds in a manner identical to that in Section 6.2, however, without performing any block averaging.

First, the local dip is estimated using the SPROJ algorithm. The estimates are illustrated in Figure 6.16. As can be seen, the dip estimates are very good in the region where the small scale beds are located. This is in spite of the fact that the signal-to-noise ratio is much smaller than it was when the same analysis was

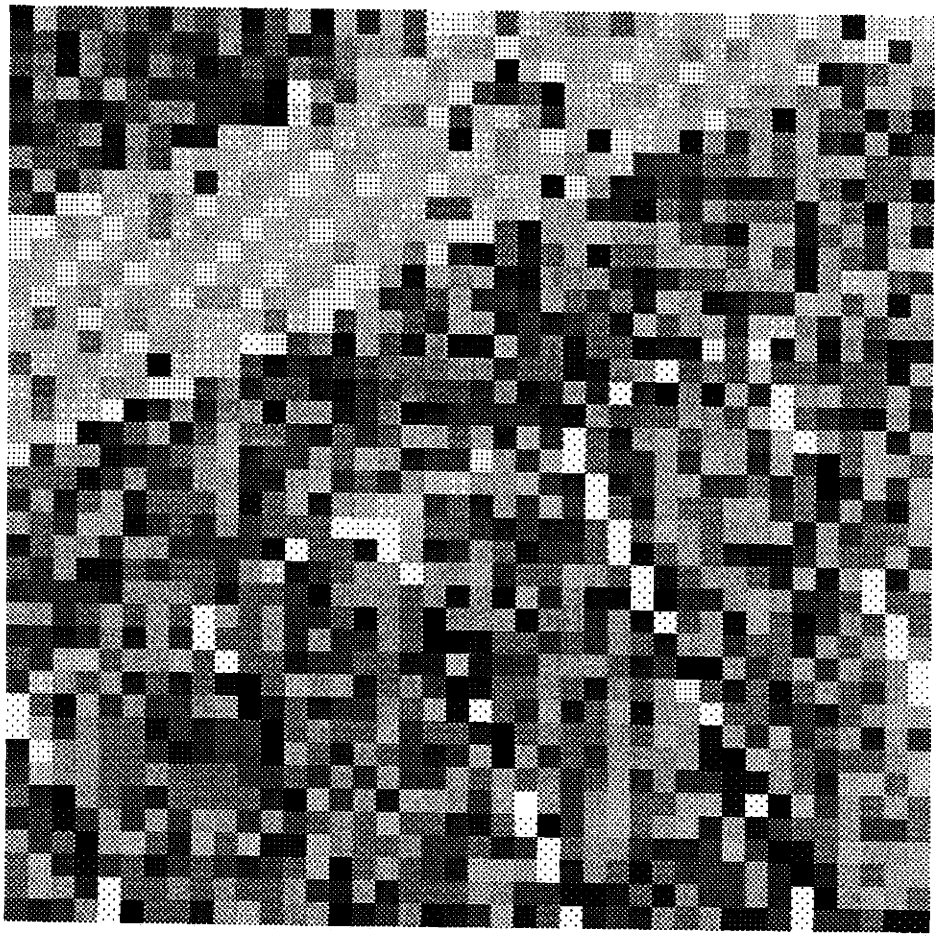

Figure 6.14: Block Variance Computation For Synthetic Data

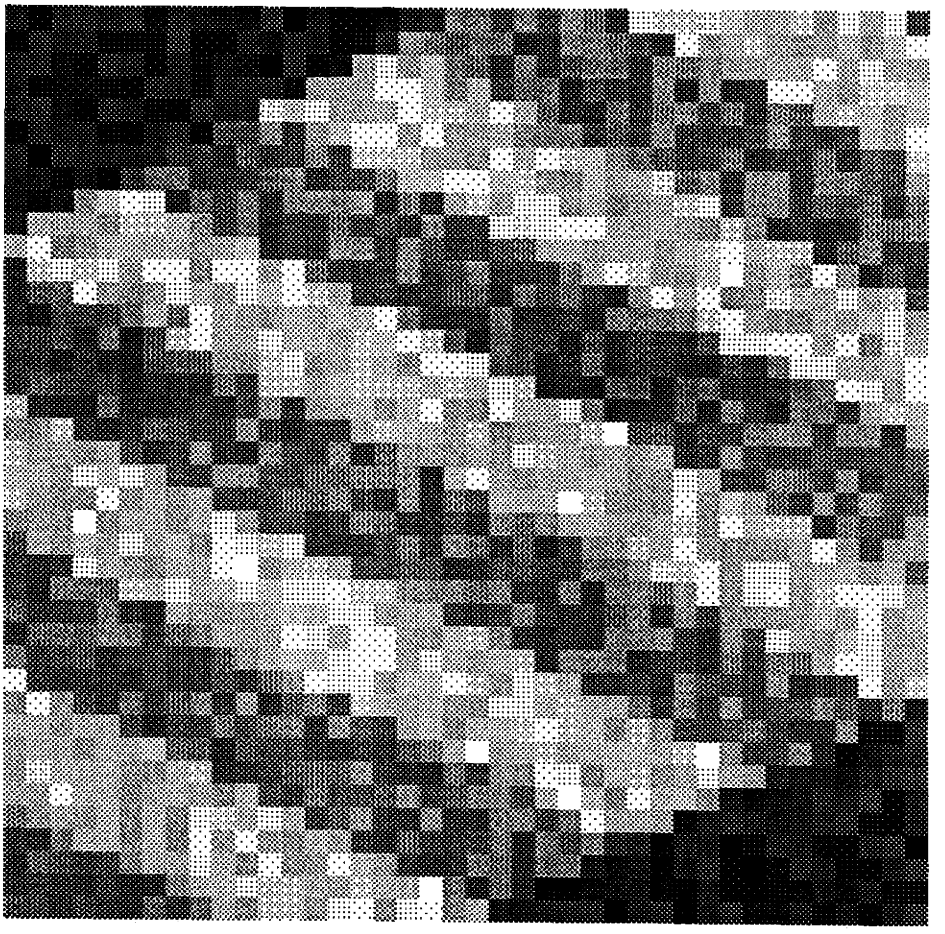

Figure 6.15: A 40×40 Block Of Data Containing Small Scale Bedding Structure

performed in Section 6.2 on the large scale data. In the upper left hand corner and the lower right hand corner of the data in Figure 6.15 are large scale beds which encroach on the small scale bedding features. These beds are of higher contrast with the average value of the small scale beds than the small scale beds are with themselves. Consequently, when a window of data contains part of one of these large scale beds, the dip estimate conforms to the slope of the large scale bed. This is clearly seen in the dip estimates of Figure 6.16.

Figure 6.17 shows the associated local edge estimates. These estimates cluster very nicely about the small scale bed boundaries as well as at the boundaries of the large scale beds. Many singletons exist within the large scale beds which are a result of estimating edges where none exist.

Figures 6.18 and 6.19 illustrate the output of the Clustering algorithm followed by that of the Kalman filtering algorithm. The Clustering algorithm uses the parameters $t_e = t_d = .9999$. Figure 6.20 is a line averaged version of Figure 6.19 with $l_c = 8$ and $l_d = 3$. Each of these procedures performs as expected and the resulting global line estimates in Figure 6.20 are very good.

6.5 Some Real Data

This section applies the methodologies used earlier in this chapter on a real data set. This data is illustrated in Figure 6.21. The data is a 160×160 array with a complicated large scale structure. As can be seen, the small scale structure which exists in the data has a small signal-to-noise ratio.

Figure 6.22 is a block averaged version of Figure 6.21. The blocks are 4×4, and consequently, the data in Figure 6.22 is a 40×40 array. Figures 6.23 and 6.24 are the local dip and edge estimates of the block averaged data. The local dip estimates capture the primarily flat bedded features of the data. The local edge estimates are fairly well clustered about large scale bed boundaries even though the signal-to-noise ratio of the block averaged data appears to be low.

Figures 6.25 and 6.26 are the outputs of the Clustering followed by Kalman Filtering algorithms. Figure 6.27 shows the line averaged version of Figure 6.26

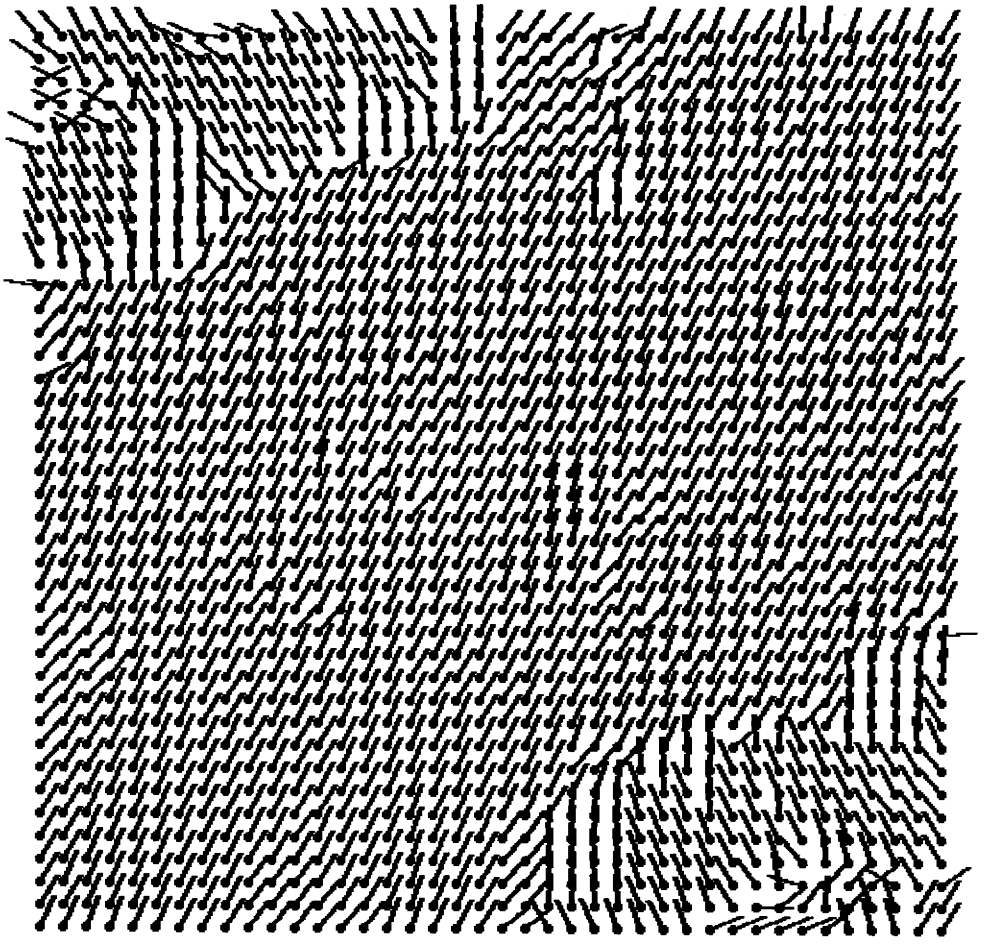

Figure 6.16: Local Dip Estimates For Small Scale Data Features

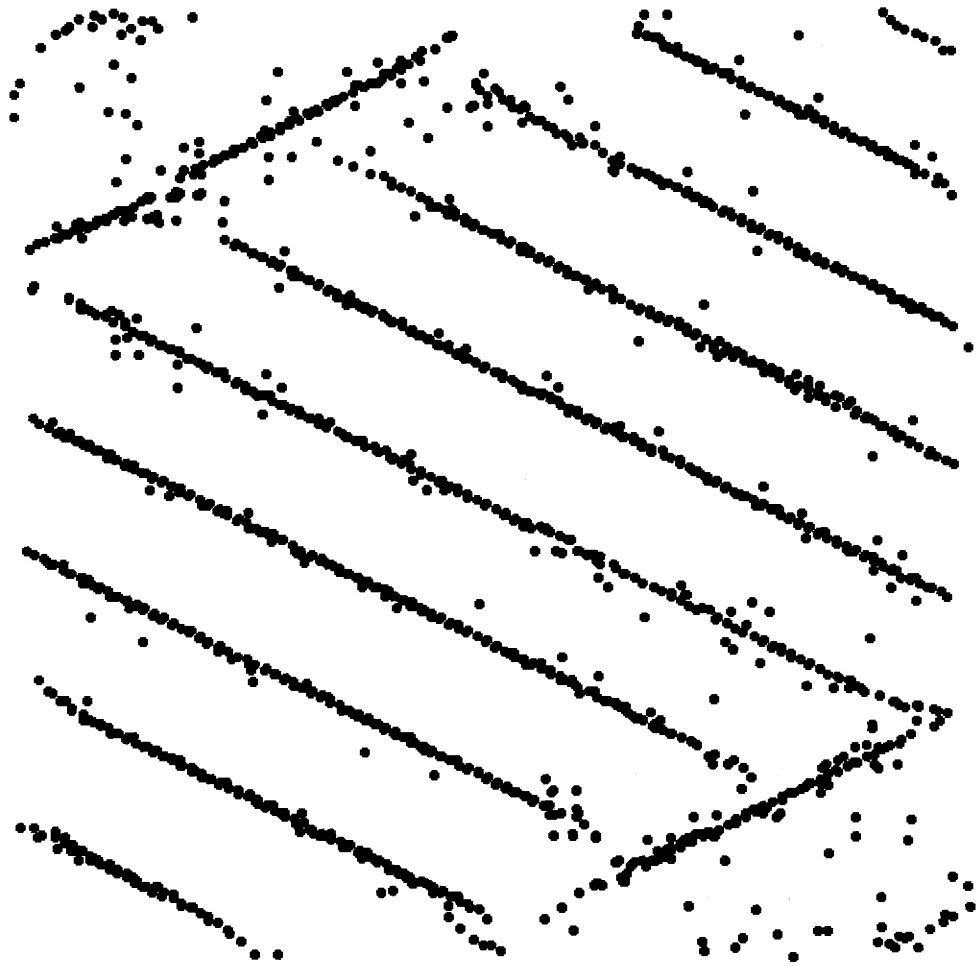

Figure 6.17: Local Edge Estimates For Small Scale Data Features

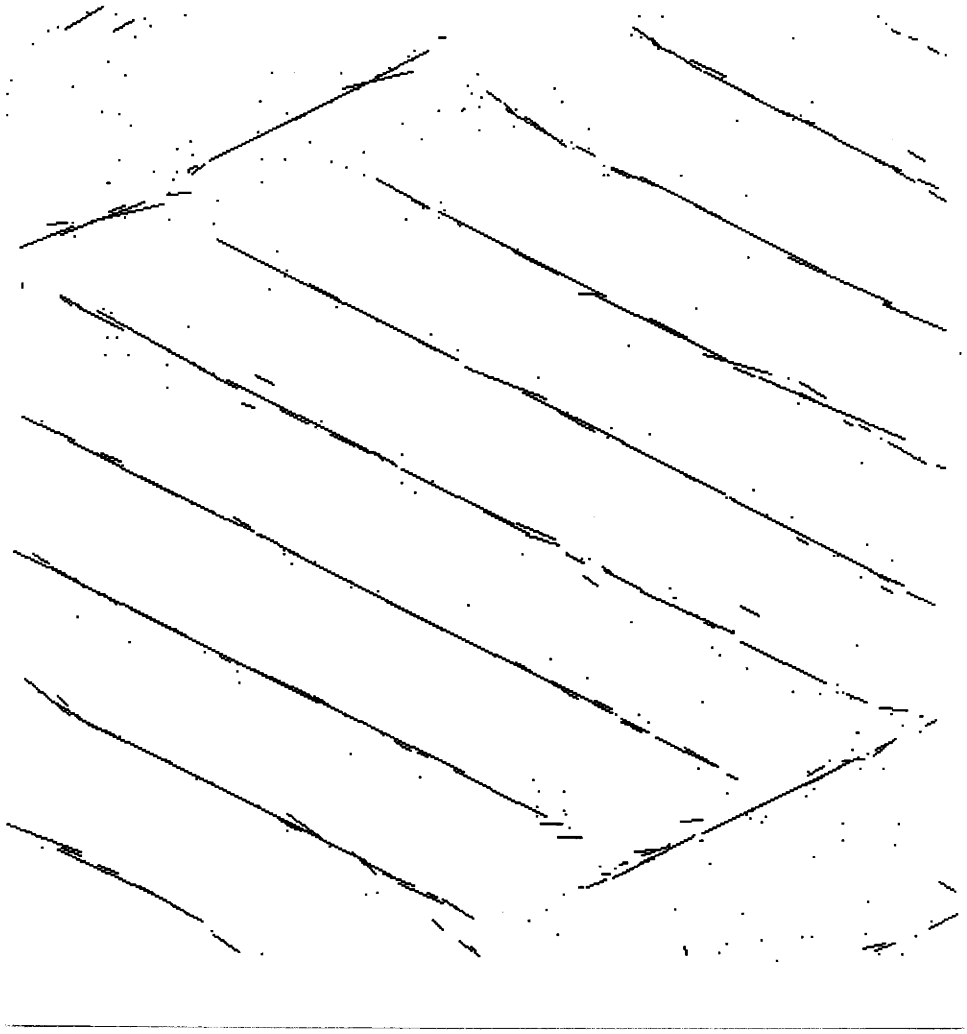

Figure 6.18: Clustering of Local Edge Estimates

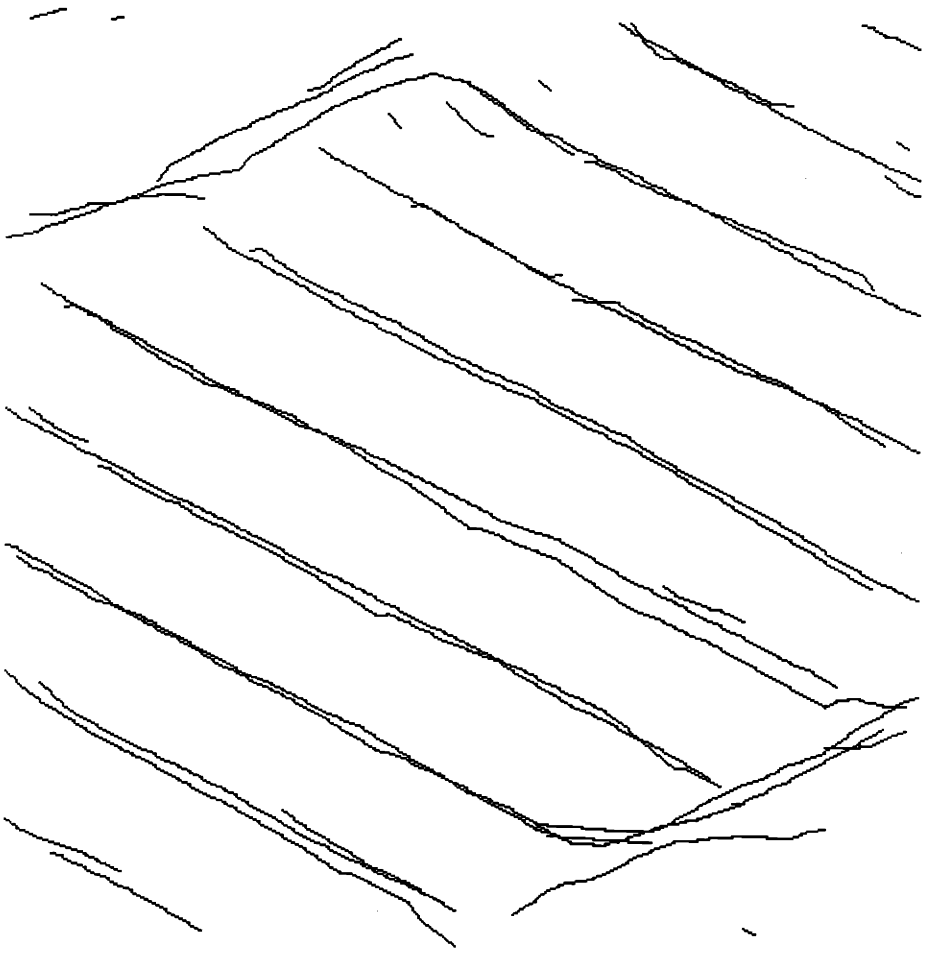

Figure 6.19: Kalman Filtering of Cluster Output

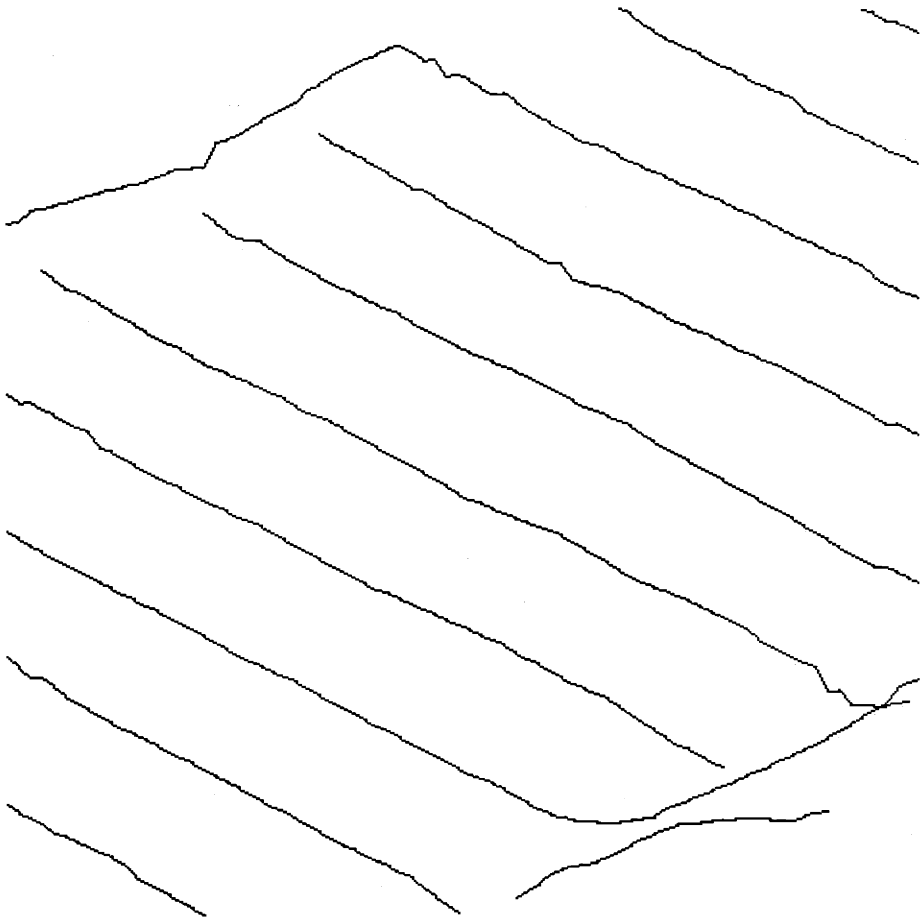

Figure 6.20: Line Averaging of Kalman Filter Output

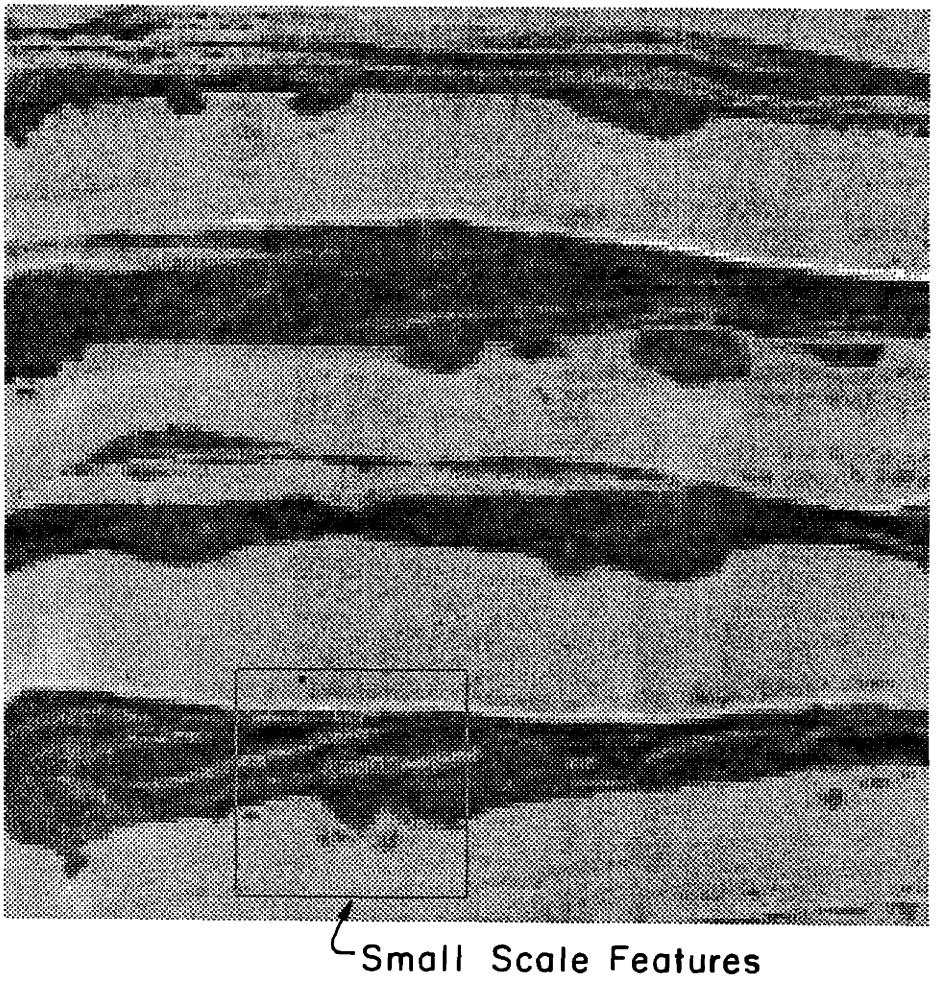

Figure 6.21: A 160 × 160 Array Of Real Data

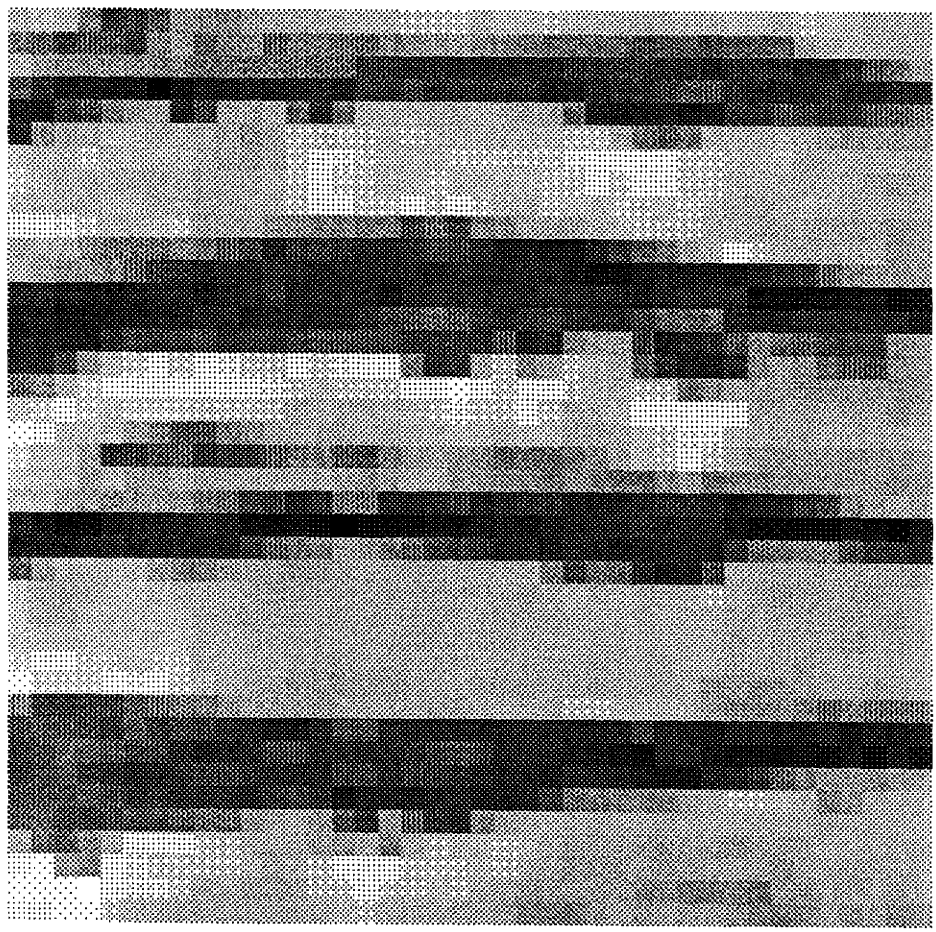

Figure 6.22: 40×40 Array Of Block Averaged Data

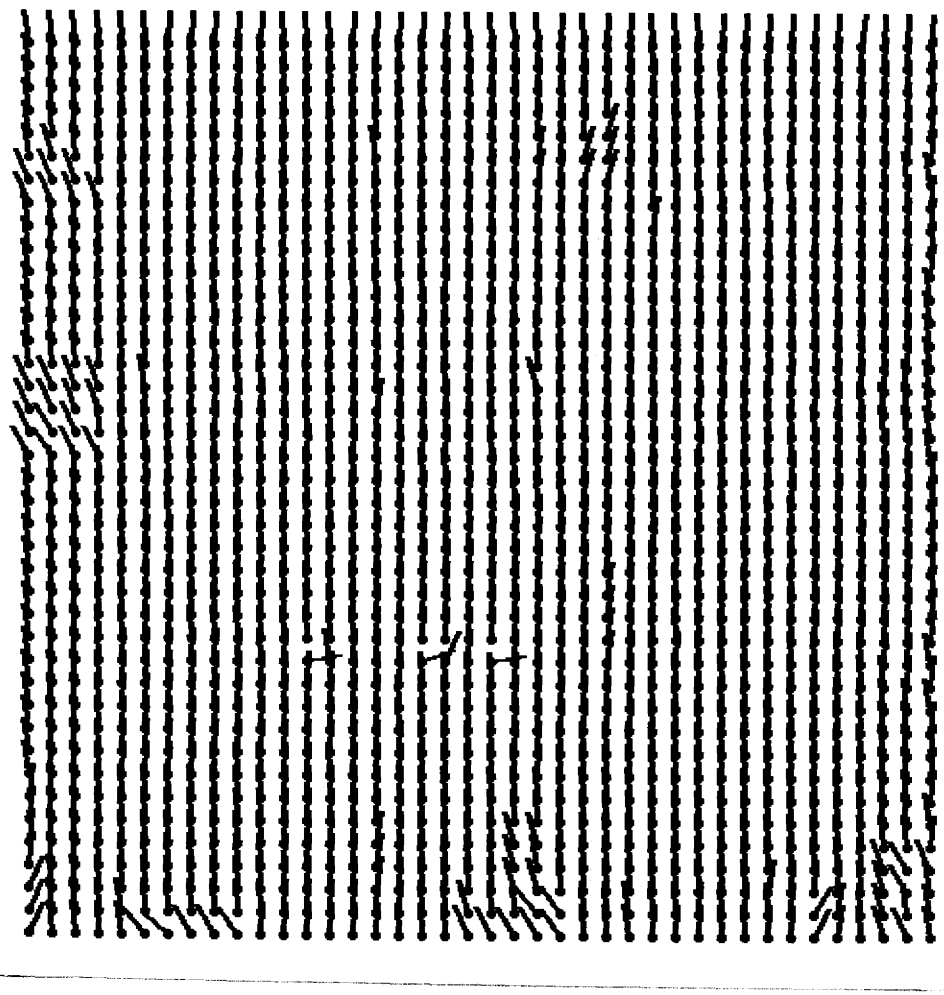

Figure 6.23: Local Dip Estimates Of Block Averaged Data

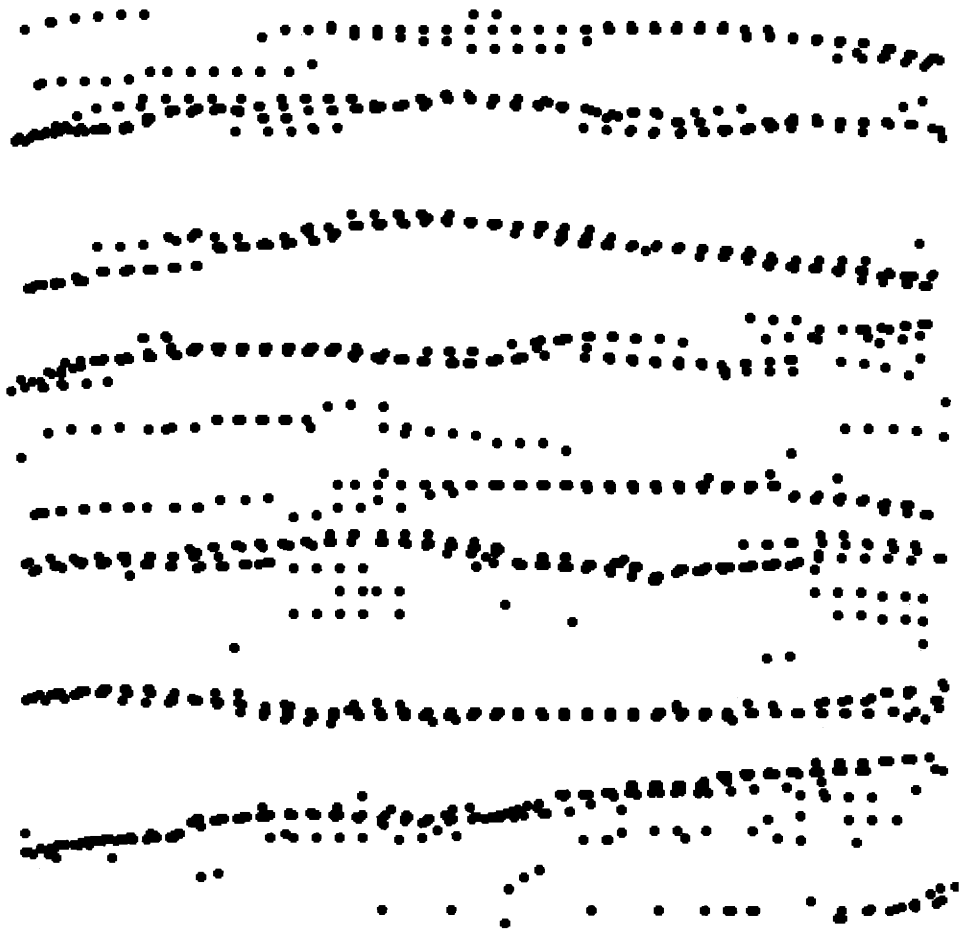

Figure 6.24: Local Edge Estimates Of Block Averaged Data

superimposed on the data of Figure 6.21 (note that the edges have been rescaled to fit the original data set). The line averaged global edge estimates track the bed boundaries remarkably well considering the underlying planar boundary assumptions of our models and the non-planar nature of the data.

Figure 6.28 illustrates the block variance computation performed on the data of Figure 6.21. This figure shows regions of high variance one of which is outlined by a black square. This 10×10 block of variance computations corresponds to the 40×40 array of data from Figure 6.21 illustrated in Figure 6.29. As can be seen in Figure 6.29 some small scale bedding structure exists in the large scale dark bed which cuts through the center of the array.

Figures 6.30 and 6.31 illustrate the local dip and edge estimates, respectively, for the data in Figure 6.29. The local edge estimates are most effectively clustered about the upper and lower boundaries of the large scale bed in the data. In particular the large bump on the lower side of this bed has local edges which are predominant. However, some clustering of edges can also be seen along bed boundaries of small scale structure within the large scale bed.

The small scale beds are more apparent in the output of the Clustering algorithm illustrated in Figure 6.32. Also, these beds are emphasized in the subsequent Kalman filtering output illustrated in Figure 6.33. The final illustration shows the result of line averaging the Kalman filter output of Figure 6.33. The line averaged data is superimposed on the small scale data of Figure 6.29 and displayed in Figure 6.34. The results show that even with a very complicated data set that the methods used can track the bed boundaries in both large and small scale beds. However, looking at the lower portion of Figure 6.34 indicates that some of the edge tracks in the small scale data may be spurious. This is reasonable since small scale data tends to be of lower contrast and consequently it is more difficult to track edges in this data. The results in Figure 6.34 could easily be post-processed to discriminate between edge tracks which are due to high contrast data and those which are due to low contrast data.

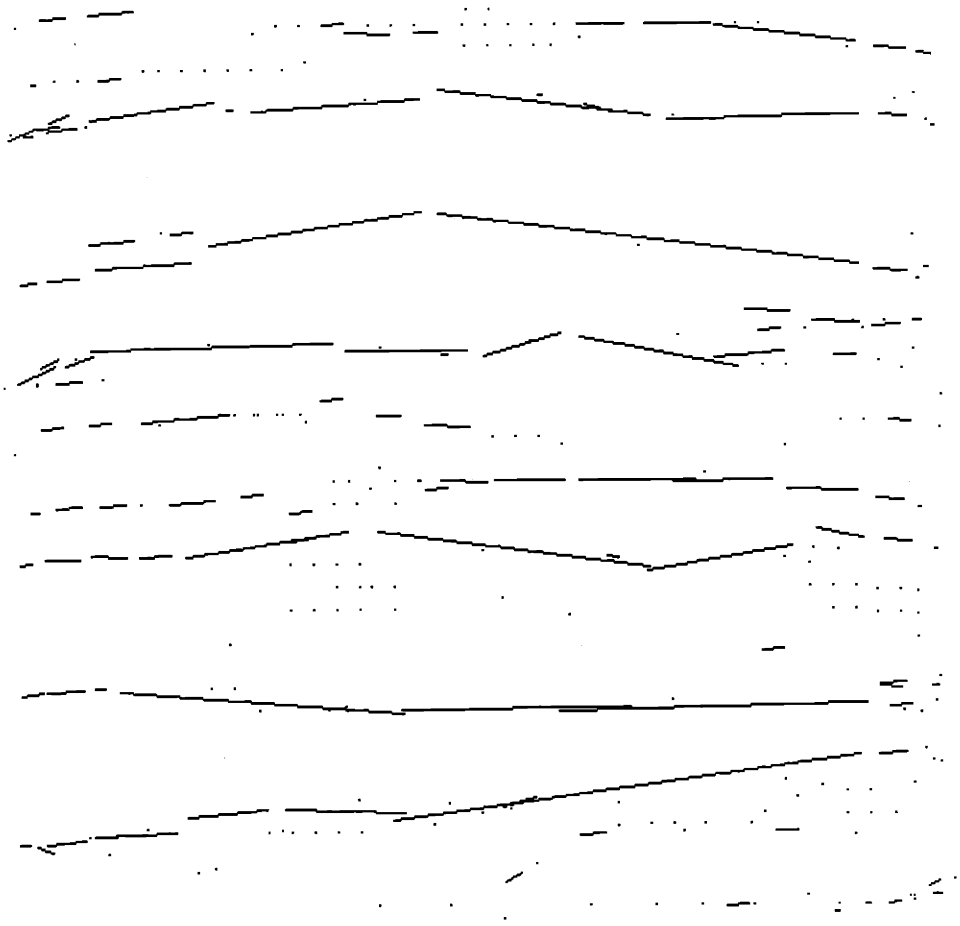

Figure 6.25: Clustering of Local Edges

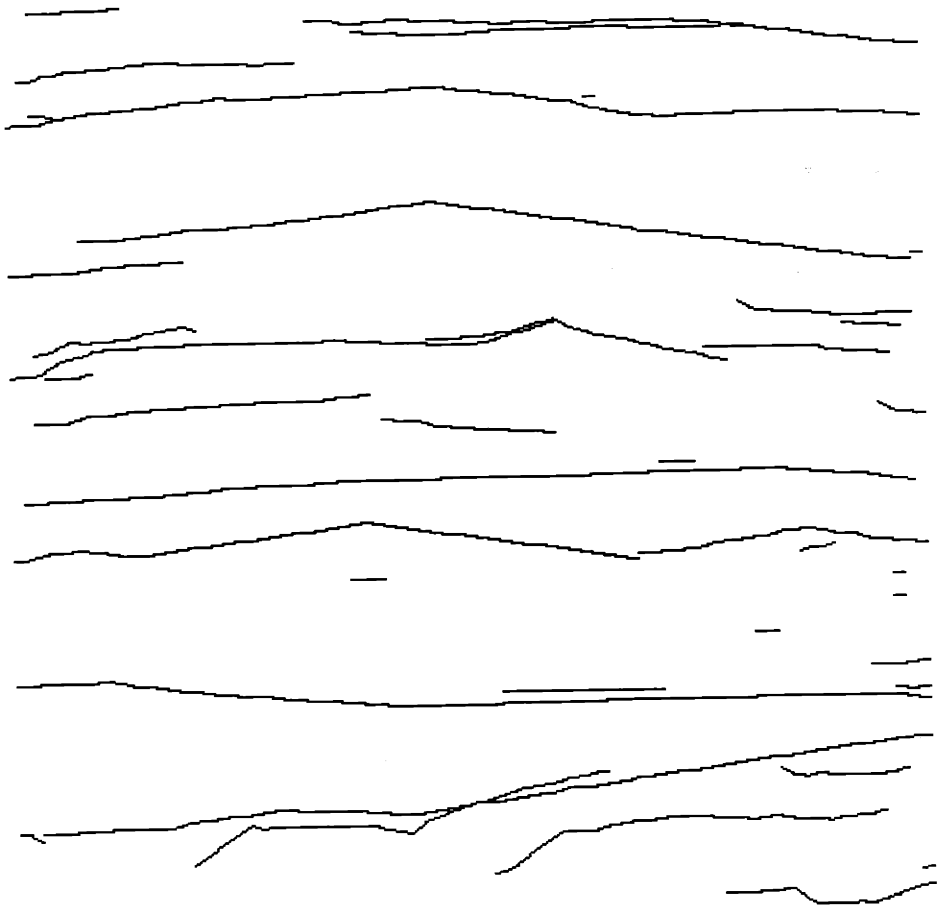

Figure 6.26: Kalman Filtering of Cluster Groups

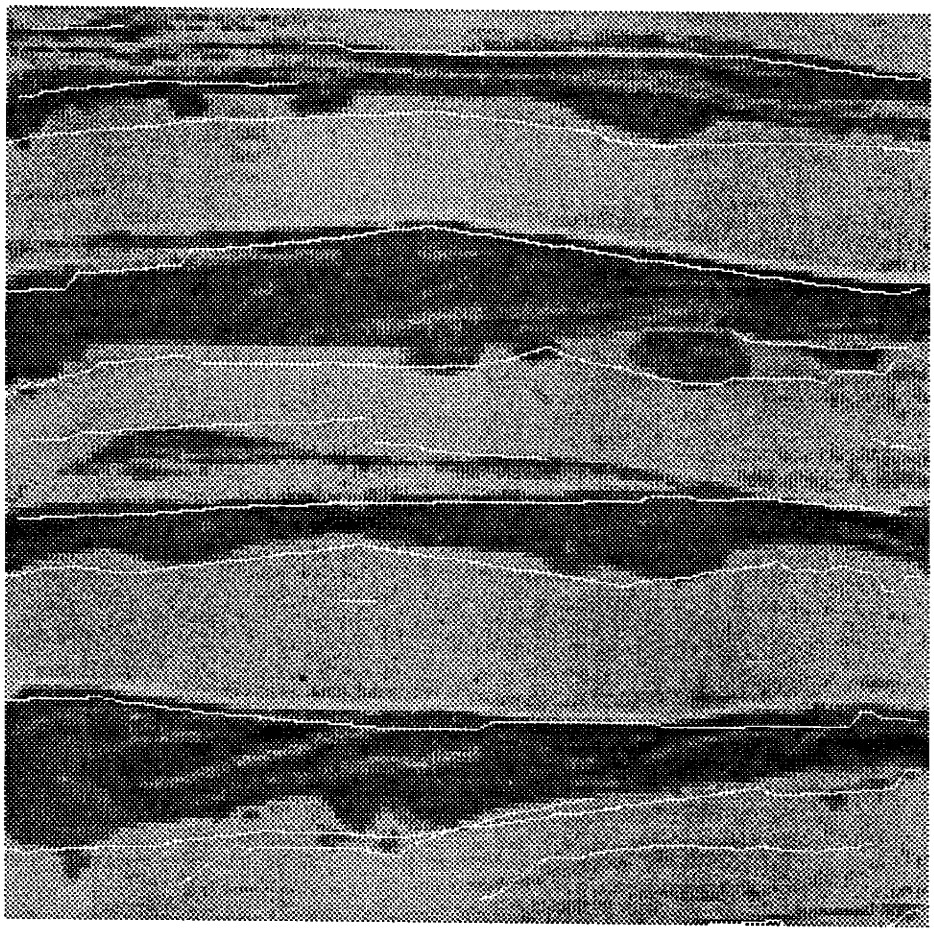

Figure 6.27: Overlay of Global Line Estimate on Real Data

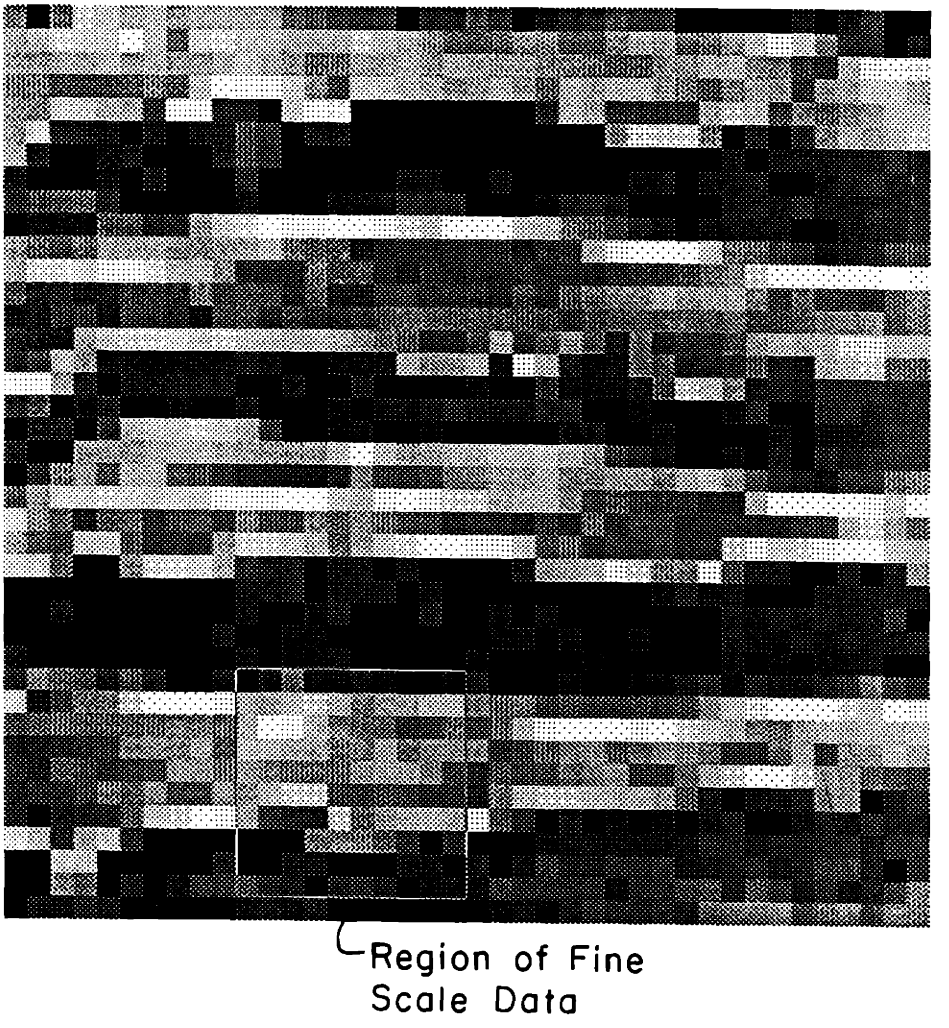

Figure 6.28: Block Variance Computation

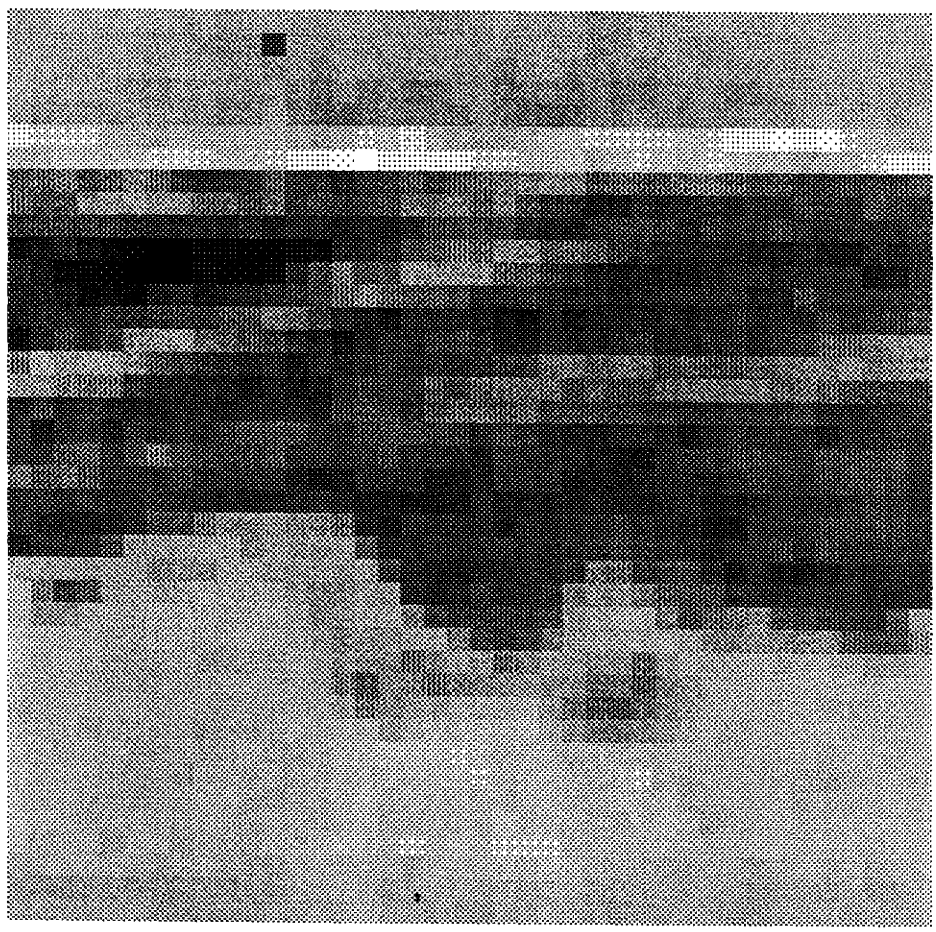

Figure 6.29: 40 × 40 Array Of Data With Small Scale Structure

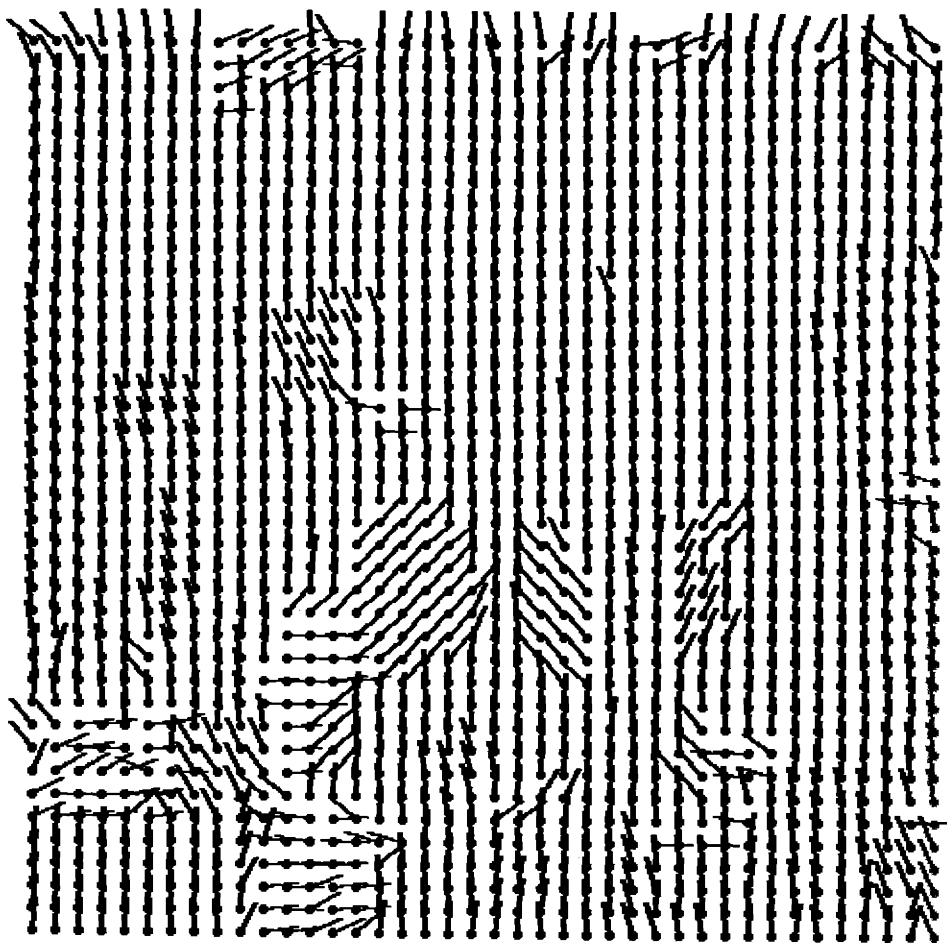

Figure 6.30: Local Dip Estimates For Small Scale Structure

Figure 6.31: Local Edge Estimates For Small Scale Structure

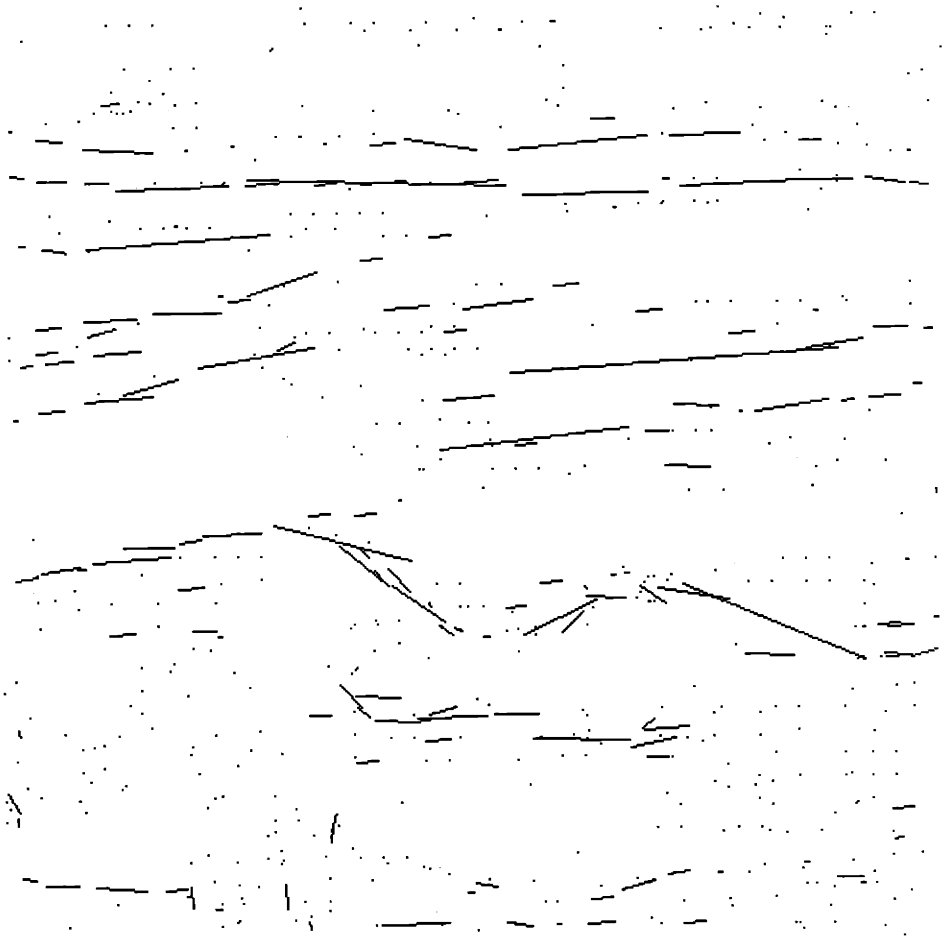

Figure 6.32: Clustering of Local Edge Estimates

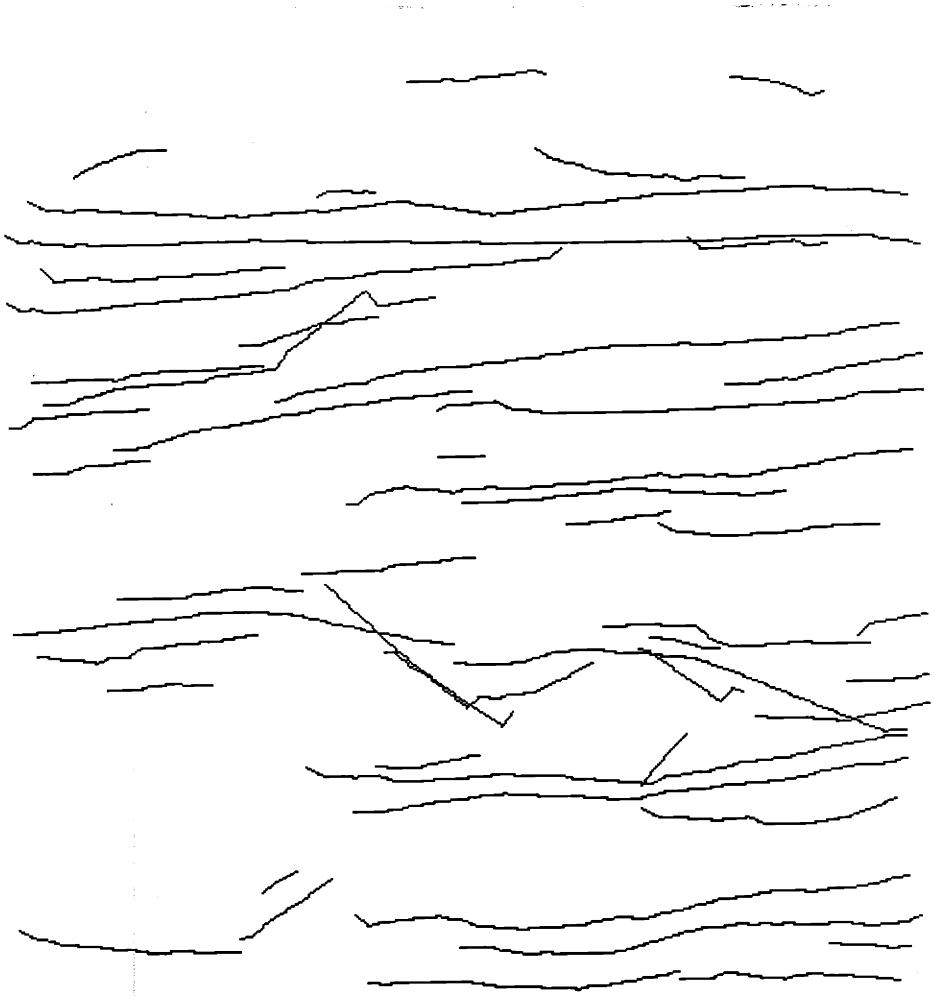

Figure 6.33: Kalman Filtering Output

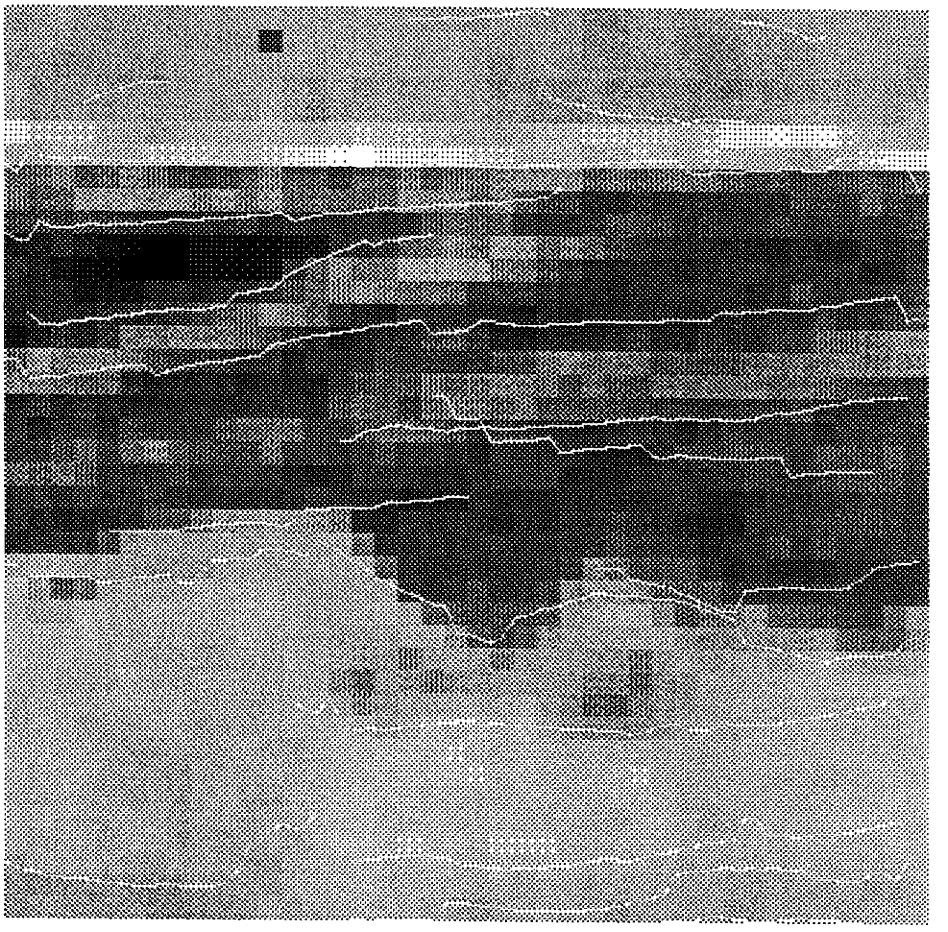

Figure 6.34: Overlay of Global Line Estimates on Small Scale Data

6.6 Conclusions

In this chapter we have applied some of the techniques developed in this thesis to the automated processing of complicated data sets. In particular, a piece of synthetic data is introduced which has features at two spatial scales and contrasts. Through the introduction of block averaging and computation of block variances it was shown how complicated data sets could be analyzed in a hierarchical fashion. Large scale structures are analyzed first and then utilizing some of the information extracted from the large scale structure, small scale structure is located and analyzed. The chapter concluded with an example of the techniques used on real data.

Chapter 7

Conclusions

The research in this thesis has been presented in the chronological order of its development. It reflects a logical flow of ideas stemming from the problem description presented in the introduction. We recapitulate the results of this thesis and identify some of the remaining problems which we feel could develop into productive areas of future research.

Chapter 3 presents a simple binary MRF model which reflects many of the characteristics of the problem described in Chapter 1. The model can generate sample functions which contain layers at prescribed inclinations and bed thicknesses. It was the variety of layered structures which could be generated using these models which convinced this researcher that MRF models would make effective tools for our problem.

The drawback of the model in Chapter 3 is that it relies on observations of level crossings to identify beds. That is, if the chosen level is zero, the ML estimation algorithm based on the model looks for coherent changes in sign to identify bed boundaries. This in turn yields information about dip and bed thickness. While our Chapter 3 signal processing algorithms work well when the data displays this essentially high contrast, alternating structure, it does not work well for layered data which has more subtle variations from bed to bed.

There are several layered data problems which the models of Chapter 3 might be eminently suited for. For example, fingerprint data consists of layers alternating about a single level value. Also, tree rings are of a similar data type. It is likely

that other naturally occurring data sets also exhibit the characteristics modeled by the MRF's of Chapter 3. Consequently, the models of Chapter 3 are well suited for the extraction of primitive features which would serve as the inputs to more complicated pattern recognition algorithms.

One possible path leading away from the results of Chapter 3 is a search for modifications of the models used in that chapter which would help overcome the level-crossing problem. One possibility is to try to incorporate a local level value as a parameter in the model which could be jointly estimated with dip and bed thickness. A second possibility is to perform the estimation procedure successively at several contrast values followed by some post-processing which would integrate the results.

Our own approach, however, was to develop another set of MRF models which capture more of the basic nature of the problem formulation set out in Chapter 1. Chapter 4 is the result of this reformulation.

The model of Chapter 4 captures more of the essential characteristics we expect to see in layered data. That is, the model explicitly incorporates the idea of a two-way correlation structure. In one direction there is long spatial correlation and in the orthogonal direction there is much shorter spatial correlation. A major difference between the model of Chapter 4 and the model of Chapter 3 is in the ability of the Chapter 4 model to characterize the dip direction without relying on the properties of data on both sides of a bed boundary. The Chapter 3 model implicitly relies on a change in sign (level) to characterize bed boundaries, and, consequently, to identify the dip direction. The Chapter 4 model characterizes dip by the orientation of the major and minor correlation directions. In this regard the Chapter 3 model looks for changes in the data characteristics in just one direction, the direction of dip. The Chapter 4 model looks for changes in both the dip direction and orthogonal to the dip direction. In fact the key feature of the model that is developed in Chapter 4 is that it defines dip as the direction orthogonal to the direction of maximal correlation. Thus, the Chapter 4 algorithms look primarily for large positive correlations orthogonal to the dip vector. This is in marked contrast to the Chapter 3 algorithms which look primarily for large negative correlations

along the dip vector.

The Chapter 4 model has additional properties worthy of note. The mathematical tractability of the Chapter 4 model has allowed us to develop a sequence of fast algorithms. These algorithms provide us with computationally efficient signal processing algorithms and they also provide additional intuition into the nature of the model. Furthermore, the covariance structure of the Chapter 4 is roughly separable. The usual definition of a separable covariance structure is that it can be expressed as a product of two one-dimensional covariance functions. That is, for $K(\underline{x},\underline{y})$ a separable two-dimensional covariance function there exists two one-dimensional covariance functions $R(\underline{x})$ and $Q(\underline{y})$ such that $K(\underline{x},\underline{y}) = R(\underline{x})Q(\underline{y})$. Our Chapter 4 model is of this form only with an additional change of basis. That is $K(\underline{x},\underline{y})$ is separable when we perform the rotational transformation corresponding to the dip on the x-y coordinate system.

Our implementation of the Chapter 4 model in terms of signal processing algorithms took two forms. One of the implementations is in the form of a MAP estimation algorithm which relies on SA to find the estimate. The other implementation is an ML estimation algorithm. Neither of the two implementations yields a reduction in the quantity of data.

Part of our objective is to compress the data by extracting higher level features. Furthermore, an essential aspect of estimating dip near bed boundaries is to identify these boundaries and to account for them in our algorithms. Both of these objectives are addressed in Chapter 5 which concentrates on estimation of bed boundaries. The local and global aspects of this problem have been formulated as an edge detection problem. The unique aspect of our own formulation of the problem is in the incorporation of the prior knowledge of the edge orientation (due to the dip estimates). Consequently, our local edge signal processing algorithms only need search for the edge displacement.

We could think of the local edge estimation problem as one in which both the edge orientation and location are part of the unknown aspects of our model. From this perspective we would not estimate orientation (dip) first and then displacement. Rather we would attempt to optimally estimate both simultaneously. This task

would require some new model which might combine the model of Chapter 4 and the model of Chapter 5. The other possibility is to perform an iterative improvement on the dip and edge estimates. That is, given the most recent dip estimate find the best edge estimate and vice versa. In any case we could expect some improvement in the performance of both the dip and edge estimation algorithms by combining the two estimation procedures.

The global edge estimation procedures presented in Chapter 5 are based on well known signal processing procedures. The clustering and Kalman filtering algorithms demonstrate the tractability of global edge estimation within the framework of our local dip and edge models. The resulting global edge estimates compress the data and estimates to a smaller set of bed boundaries. The bed boundaries contain the pertinent dip information of the data and they serve as a stepping stone to the goal of higher level interpretation of the data.

The final issue dealt with in this thesis is that of the efficacy of our estimation procedures when applied to data containing features at several spatial scales. Chapter 6 suggests a procedure for hierarchically processing the data. First, the coarse scale structure in the data is processed followed by the identification of regions in the data where fine scale structure may exist. The fine scale structure is then analyzed in a manner similar to that of the coarse scale structure. The method we employed relies on block averages of the data to reduce its spatial scale.

The results of Chapter 6 show how to decompose the dip estimation problem by scale. Furthermore, this spatial scale decomposition is an important component for extracting higher level features from the data (such as patterns of dip which can be used to identify geological structures).

The results contained in this thesis cover a significant amount of material in the development of models and signal processing for low level features in layered data. There are several logical paths which could be developed as a result of this work. The following discussion suggests four possible directions for future research.

The first possibility concerns the implications of ARC function models with regards to the problem of beamforming [8]. There is a large body of literature concerning methods for steering phased arrays for the purpose of estimating direc-

tion of arrival of plane waves. This problem has many of the same characteristics as the layered data problem discussed in this thesis. Consequently, determination of the formal relationship between the ARC function models of Chapter 4 and beamforming problems would be of great interest.

The second possible line of research concerns the relationship of the anisotropic random field models developed in Chapter 4 and isotropic random field models. The models used in Chapter 4 can be viewed as warped versions of isotropic random fields. That is, the models of Chapter 4 can be obtained by stretching and rotating the coordinate axes of an isotropic random field model (say by a linear transformation). This observation suggests the possibility of examining other classes of anisotropic random field models obtained by other warping operations on isotropic random fields.

The third suggestion for future research concerns error analysis of the signal processing algorithms developed here. The models of Chapter 4 are mathematically tractable. This is also true for the edge models in Chapter 5. Consequently, it should be possible to analyze analytically the errors associated with the signal processing algorithms developed in these chapters.

Finally, of great interest is the search for a cohesive framework for extracting high level features from layered data such as specific dip patterns, sudden changes in dip patterns, etc. An important line of future research is to incorporate the models and algorithms presented here into a higher level scheme for modeling and estimating these features.

Bibliography

[1] Basseville, Michèle; Espiau, Bernard; Gasnier Jacky. "Edge Detection Using Sequential Methods for Change in Level—Part 1: A Sequential Edge Detection Algorithm," *IEEE Trans. Acous., Speech, and Signal Proc.*, vol. ASSP-29, no. 1, pp. 24-31, Feb. 1981.

[2] Besag, J. E. "Nearest-Neighbor Systems and the Auto-Logistic Model for Binary Data," *J. R. Statist. Soc.*, series B, vol 34, pp. 75-83, 1972.

[3] Besag, J. E. "Spatial Interaction and the Statistical Analysis of Lattice Systems (with discussion)," *J. R. Statist. Soc.*, series B, vol. 36, pp. 192-226, 1974.

[4] Campbell, R. L. Jr. "Stratigraphic Applications of Dipmeter Data in Mid-Continent," *Am. Assoc. Petr. Geol. Bulletin,* vol. 52, no. 9, pp. 1700-1719, September, 1968.

[5] Cross, George R.; Jain, Anil K. "Markov Random Field Texture Models," *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. PAMI-5, no. 1, pp. 25-39, Jan. 1983.

[6] Davis, Randall; *et al.* "The Dipmeter Advisor: Interpretation of Geologic Signals," Preprint, MIT Artificial Intelligence Laboratory.

[7] DeGroot, Morris H. *Probability and Statistics* Redding, MA: Addison-Wesley Publishing Company, 1975.

[8] Dudgeon, Dan E. "Fundamentals of Digital Array Processing," *Proc. IEEE,* vol. 65, no. 6, pp. 898-904, June 1977.

[9] Ekstrom, Michael P., *et al.* "Formation Imaging With Microelectrical Scanning Arrays," *Conf. Proc. SPWLA 1986* Houston: 9-13 June 1986.

[10] Geman Stuart; Geman Donald. "Stochastic Relaxation, Gibb's Distributions, and the Bayesian Restoration of Images," *IEEE Trans. Patt. Anal. and Mach. Intell.,* vol. PAMI-6, no. 6, pp. 721-741, Nov. 1984.

[11] Gilreath, J. A.; Healy, J. S.; Yelverton, J. N. "Depositional Environments Defined by Dipmeter Interpretation," *Trans. Gulf Coast Assoc. of Geol. Soc.,* vol. 19, pp. 101-111, 1969.

[12] Hansen, F. R.; Elliott, H. "Image Segmentation Using Simple Markov Field Models," *Computer Graphics and Image Processing,* vol. 20, pp. 101-132, 1982.

[13] Hassner, Martin; Sklansky, Jack. "The Use of Markov Random Fields as Models of Texture," *Computer Graphics and Image Processing,* vol. 12, pp. 357-370, 1980.

[14] Kashyap, Rangasami L.; Chellappa, Ramalingam. "Estimation and choice of Neighbors in Spatial-Interaction Models of Images," *IEEE Trans. Infor. Theory,* vol. IT-29, no. 1, pp. 60-72, Jan. 1983.

[15] Kinderman, Ross; Snell, J. Laurie. *Markov Random Fields and Their Applications,* Contemporary Mathematics, vol. 1, Providence: American Mathematical Society, 1980.

[16] Kirkpatrick, S.; Gellatt, C. D.; Vecchi, M. P. "Optimization by Simulated Annealing," *Science,* vol. 220, no. 4598, pp. 671-680, 13 May 1983.

[17] Marroquin, Jose Luis. "Probabilistic Solution of Inverse Problems," Laboratory for Information and Decision Systems Technical Report, LIDS-TH-1500, MIT, September 1985.

[18] Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; Teller, E. "Equation of State Calculations by Fast Computing Machines," *J. Chemical Physics,* vol. 21, pp. 1087-1092, 1953.