PORE STRUCTURE FROM ROCK IMAGES VIA PATTERN RECOGNITION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF GEOPHYSICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Brian Boru Quinn
September 29, 1989
Rockphysics Laboratory Edition

©Copyright by Brian Boru Quinn 1989

All Rights Reserved

Rock Physics Laboratory Edition - dissLo. 9.89bbg

I certify that I have read this thesis and that in my opinion it is fully adaquate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adaquate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Francis Muir

I certify that I have read this thesis and that in my opinion it is fully adaquate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

A. G. Journel

finished 5.9.89bbg Brian B. Quinn

I certify that I have read this thesis and that in my opinion it is fully adaquate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jon F. Claerbout

Approved for the University Committee on Graduate Studies:

(Dean of Graduate Studies)

DISSERTATION ABSTRACT

PORE STRUCTURE FROM ROCK IMAGES VIA PATTERN RECOGNITION

My goal was producing an image analysis technique, estimating rock's fluid permeability from thin section images of its pores. While I did not reach that goal completely, I did complete an analytical technique that evaluates pore structure, given rock images. The resulting technique may extended to applications beyond the original goal, yet it serves rock-image permeability estimates as well.

Current methods of estimating permeability from rock images study the boundaries of the imaged pores. My work has applied an additional technique that is well known in the computer vision field yet little used on rock data. It is called a medial axis transform. It simplifies two-dimensional, 0-or-1 features into an image of one-dimensional (arc) features. This provides a summary of image contents. I synthesized a pattern recognition technique that extracts these summary arcs into files, sorted according to the feature's topology. These files contain descriptive lists of image contents. They serve whatever tasks demanded the collection of the image in the first place. As a result, the pattern recognition tools I have developed are very general in their utility. For rock images, I felt the main concern of pore-structure pattern recognition was fluid flow problems.

My work provides a new way of extracting rock image features for analysis. It works equally well for pore Boundaries or networks of Arcs, like those derived from thinning of pore images. This versatility has not been found in earlier rock feature extractions. Applications of my technique to rock permeability estimates will involve some extensions to it: an interactive input program for boundary condition application, and a circuit response code like SPICE is needed to fully solve the problem. Output from my technique are data, not analytical results. Different applications may use the lists in different ways, processing lists at reduced cost for complicated, repettitive searches, compared with reapplication of image processing for each search. Medial axis lists give additional measures of pore throat and chamber size throughout the image, which is valuable for permeability and diffusivity estimates from digital image data.

PREFACE

Outlook

As a writer I treat my work as though it were a conceptual colony in the land of artificial intelligence. Here, certain ideas familiar to earth science culture must be grown from scratch their first time, in a place where the resources for growth are abundant. With my desire to build a petrographer's intuition into a machine as a point of reference, I have explored the vicinity of the Petrographic Image Analysis (PIA) colony established by Prof. Robert Ehrlich in the first half of the 1980's. I found sources of useful ideas known to exist but largely untapped by the PIA community, which has built a bridge of statistical support that is used by oil company departments involved with development geology, as well as by commercial core analysis firms. PIA has provided commercially viable pragmatic measures to help these geologists.

My goal is similar but less specific—part of my results serve the same analytical purpose as PIA (pore boundary measures) and another part provides network analysis (pore medial axis measures). By presenting my software in a developmental way, from basic principles to some implementation details, I hope to give a non-proprietary alternative to rock image analysts. I hope further that my work may help suggest areas of useful discussion and needed work in future geologic image analysis, so that more of geologic thought can be made to travel in a machine where human geologists can not, and that drudgery may be driven even further from the daily routine of the world's geologists.

Thanks

I wish to thank three groups of people. One gave me finances, one standards, and one direction. The first group have given me the stipend support that let me live my own life, after I was 18. The second ones gave me a reference standard, in living one's own life, however one sees fit. The third have provided the inventive suggestions that have made my educational journey intriguing.

Since entering college eleven years ago, I have received occasional assistance. This has come from my parents, Wendell Quinn and Mary Lou Kennedy. My grandmother, Mary McCarroll, and stepfather, Allan Kennedy, have also provided occasional assistance, in times of need and in times of enjoyment.

In living the lifestyle made possible by both the people above and my employers, I have learned many skills from those who also have given me emotional and loving support. I have found some things can be more fun when you run with one who is of a like mind in ways diverse, social, and scientific. So in addition to my wife Karen Grove I would like to thank Barclay Kamb and family, my immediate aunts and uncles, my sister and cousins and cars by which I saw all of them, too.

The Third Group

I started in geophysics because of the experience of Jean-Bernard Minster's introductory class. I continued in geophysics through three schools because of the common thread provided by Barclay Kamb, through my employment by him under the NSF Polar Programs' Variegated Glacier Surge field expeditions of summer 1979, '80, '81, and '82. I gained warmer experiences in the Owens valley, northernmost Antelope valley, Coachella valley, western shores of Lake Mojave and Mojave Desert, and Rio Grande rift west of Santa Fe, NM, thanks to Shawn Biehler of UC Riverside. Also through his guidance, I performed seismic work on the USMC bases at Camp Pendleton and Twentynine Palms, California. The Twentynine Palms field work became my senior thesis.

Then at Stanford, Amos Nur helped me think of better ways to do the work that I had done, during five undergraduate years. Six years here have taught me that two by three is more stable than one by six. Still, Amos let me keep my average years spent per college degree at a fraction less than four. Numerical analysis aside, I've enjoyed it.

My first three years of graduate study were seismic ones, doing oil field work on the eastern shores of Lake Maracaibo, estado Zulia, westernmost Venezuela. Times

were studious and thanks to classes taught both in the geophysics department and Electrical Engineering departments at Stanford, I have gone from uncertainty about what was convolution, to concern with compiler tuning and rock pore structure as metaphor for geological imagery.

My fellow students at Stanford have helped me more than any other group of students I have studied with, at four universities and one small technical institution, throughout southern California and the Bay area. I look forward to a continuation of this trend of ever-increasing interaction with my peers. In particular, the students of André Journel and John Harbaugh have helped me in specific parts of my work. A. Desbarats and M. Srivastiva have provided answers to my geostatistical problems—trying to decide how morphology fits into the earth science disciplines. P. Kushner has helped me with general statistical and numerical analysis problems.

Kai Lanz and Ron Lyon provided equipment and support in assembling the Stanford Rockphysics Laboratory's image analyzer. The staff support at Stanford has been fantastic. Thanks to Margaret Muir and the geophysics department staff over the years 1983–1989. Thanks also to Kai Lanz, Rick Ottolini, Ron Lyon, and Jon Claerbout, who have all provided equipment that, without their having declared it surplus, would have never been incorporated into the image analyzer. The original provider of equipment to start the image analyzer was Chris MacAskill of Core Laboratories.

The FMT program was completed in December 1987 because the School of Earth Sciences at Stanford supported the purchase of the vision processor in November 1986. That processor made possible major changes to the Rockphysics Laboratory's IBM PC-AT that had been in use since August 1985. A later contribution of an IBM RT/PC was from David Pollard of the Applied Earth Sciences department. Final programming hardware support from Jerry Harris of the Geophysics department made the CSC program possible in March 1989.

During my stay at Stanford I have enjoyed perpetual funding. Upon my arrival,

this was due to a Geophysics Department Fellowship. Then, I received several years of stipend support from the Stanford Rock Physics Program. Some quarters were funded in part by the Gas Research Institute, via Stanford's Information Systems Lab and later via Stanford Rock and Borehole Geophysics program. Other quarters I received funding from Sohio's Stanford Center for Reservoir Forecasting (SCRF), and the Shell and Mellon grants through the Geophysics Department. SCRF provided a forum for collaboration and idea-sharing among the departments of Applied Earth Science, Petroleum Engineering, and Geophysics. During most of my image analysis research work, I have been supported by the Stanford Rockphysics Laboratory, until this year, when I have been supported by the Cecil H. Green Fellowship while completing my dissertation. I owe special thanks to Amos Nur for maintaining this constant flow of stipend funds through the six years and four summers I have enjoyed at Stanford. This Rockphysics Laboratory Volume edition has benefitted particularly from the constructive suggestions of readers Amos Nur, Francis Muir, André Journel, and Jon Claerbout.

Contents

1 Introduction **2**

2 Pore Structure From Rock Images **8**

2.1 Introduction 9

2.2 Fast Montoto Thinning (FMT) 11

2.2.1 Sobel Neighbor Coding 11

2.2.2 Fast Montoto Thinning 11

2.2.3 Fast Montoto Thickening 12

2.2.4 Cederberg-Sobel Coding (CSC) 15

2.2.5 How to run Fast Montoto Thinning 16

2.3 Cederberg-Sobel Coding 18

2.3.1 Metalines 18

2.3.2 CSC's Basic Ideas 22

2.3.3 What to do at a Unode 24

2.3.4 What to do at a Node 28

2.3.5 How CSC is run 29

2.4 Summary 34

2.5 Glossary 35

3 Application of FMT to Rock Pores **38**

3.1 Re-introduction 38

3.2 Review of Applications Work Done 40

3.3 Discussion 44

4 Reasons To Apply FMT 66

4.1 Quantifying Image Connectivity 68

4.1.1 Percolation Threshold Recognition 68

4.1.2 Mapping Networks Images 69

4.1.3 Fluid Flow Property Measurements 70

4.2 Thinning's Efficient Summaries 73

4.2.1 Pore Throat Size Dispositions 74

4.2.2 Thickening's Additional Connectivity Measurements 76

4.2.3 Boundary Measurement With CSC 77

5 Morphological Concepts and Terminology 79

5.1 Set Theory, Text, Symbolics 81

5.1.1 Set Theory Symbols Definitions 82

5.1.2 Text, Templates, And Sets 84

5.1.3 Fontainebleau's Morphological Symbolism 87

5.1.4 Morphological Presentations Review 96

5.2 Bel-Lan And Montoto's Thinning 99

5.2.1 Point Neighborhood Descriptions 99

5.2.2 Template Tests 102

5.2.3 Bel-Lan And Montoto's Thinning-2 104

5.3 Rosenfeld's Hybrid Presentation 107

5.3.1 Raster Tracking Described In Text 108

5.3.2 Nagging Implementation Details 109

5.3.3 Sequential Vs. Parallel Symmetries 110

5.3.4 Idea Consolidation In Prose 112

5.4 Fontainebleau Morphology Symbolism 114

5.4.1 Description Of Alternatives 114

5.4.2 A Convolution-like Process 127

5.4.3 Obfuscation That May Help 131

5.5 Discussions Of Morphology 134

5.5.1 FMT Is Not PIA Repackaged 135

5.5.2 Erosion Vs. Ablation 136

5.5.3 Thinning Vs. The Skiz 138
5.5.4 Thinning's Lost Information 141
5.5.5 Boundary And Arc Coding 146
5.5.6 Reprise: Morphological Description 147

6 Numerical Morphology Techniques 149
6.1 Some Processing Terminology 149
6.1.1 Raster Arrays 151
6.1.2 Graphical Vectors 154
6.1.3 Vector-to-raster Conversion 158
6.1.4 Raster-to-vector Conversion 160
6.1.5 Faster RVC and VRC 165
6.1.6 Raster Engines Paradigms 169
6.2 Morphological Lattice Concepts 172
6.2.1 Gridded Vs. Continuous 172
6.2.2 Nonlinear Operators 174
6.2.3 Parallel Process Restructuring 176
6.2.4 Neighbor Code Transform 177
6.2.5 Processor Architecture Maps 179
6.3 Look Up Tables 184
6.4 Numerical Approach, Reflections 191

7 FMT Discussion 193
7.1 Throat Aperture Estimation 194
7.2 Unopened Thinning 198
7.3 Collinear Segment Channelization 211
7.3.1 Homotopic Closing 212
7.3.2 Polygon Coding Of Arcs 215
7.3.3 Parabolic Curve Fitting 216
7.4 Global Thinned Arc Analyses 218
7.4.1 Segmentation Problems 218
7.4.2 Intermediate Result Histograms 220

7.2 Unopened Thinning 198
7.3 Collinear Segment Channelization 211
7.3.1 Homotopic Closing 212
7.3.2 Polygon Coding Of Arcs 215
7.3.3 Parabolic Curve Fitting 216
7.4 Global Thinned Arc Analyses 218
7.4.1 Segmentation Problems 218
7.4.2 Intermediate Result Histograms 220
7.4.3 Orientation Histogramming 220
7.5 Arc Vectorization Analyses 223
7.5.1 Boundary Coding Of Everything 223
7.5.2 Coding Thinned Arcs 224
7.5.3 Vectorization 226

8 Conclusions **229**
8.1 FMT is relevant to elongated object classification 230
8.2 FMT helps characterize fat interconnected systems 234
8.3 FMT Needs CSC 240

A The CSC Topology **245**

List of Figures

2.1 Data flow in FMT and CSC 10
2.2 Thinning vs. a blob 12
2.3 The FMT Algorithm 13
2.4 Montoto Thickening 14
2.5 A Thinning-Thickening Cycle 15
2.6 Periodic Table of CSC Morphemes 19
2.7 The need for metalines 20
2.8 Metaline Formation Technique 21
2.9 Metalines vs. Rosenfeld's Conundrum 23
2.10 Topology of Node Types 25
2.11 Topology of Pnode Types 26
2.12 CSC control flow 30
2.13 CSC logic diagram 31

3.1 video data perspective plot, `APAT0` 42
3.2 video data perspective plot, `APAT05` 42
3.3 Binary input to FMT `APAT0` 43
3.4 Binary input to FMT, `APAT05` 43
3.5 FMT result perspective plot, `APAT0` 44
3.6 FMT result perspective plot, `APAT05` 45
3.7 Map view of binary and FMT result, `APAT0` 45
3.8 Map view of binary and FMT result, `APAT05` 46
3.9 Histogram plot of width values for `APAT0` 47
3.10 Histogram plot of width values for `APAT05` 48
3.11 Permeability/Porosity plot for seven samples 49

3.12 Permeability/Porosity plot for three samples with trend 50
3.13 Digital Photomicrograph, Fontainebleau-alpha 51
3.14 Digital Photomicrograph, Fontainebleau-epsilon 52
3.15 Digital Photomicrograph, Fontainebleau-eta 53
3.16 Contour plot, Fontainebleau-alpha image 54
3.17 Contour plot, Fontainebleau-epsilon image 55
3.18 Contour plot, Fontainebleau-eta image 56
3.19 Fontainebleau-alpha image, binary and FMT result 58
3.20 Fontainebleau-epsilon image, binary and FMT result 59
3.21 Fontainebleau-eta image, binary and FMT result 60
3.22 Fontainebleau-alpha image, histogram of FMT results 61
3.23 Fontainebleau-epsilon image, histogram of FMT results 61
3.24 Fontainebleau-eta image, histogram of FMT results 62
3.25 Fontainebleau-alpha image, FMT active area history 63
3.26 Fontainebleau-epsilon image, FMT active area history 63
3.27 Fontainebleau-eta image, FMT active area history 64

5.1 basic logic and set theory symbols 84
5.2 Original Medial Axis Templates 103
5.3 FMT Thickening Structuring Element 143

6.1 A Recognition Technology, Inc. PX-501 processor map 180
6.2 Mark 2 RTI Engine Map 188

7.1 Thinning Leaves 202
7.2 Bel-Lan and Montoto's Benchmark Thinning 204
7.3 Trygstad, et al. Thinning 205
7.4 Cajon Pass Borehole Televiewer Data 207
7.5 Kern River Formation Thinning 209

Chapter 1

Introduction

I present a three part algorithm to convert a type of geological raster data (0 or 1 valued pixels or "pels" in two dimensions) to vector form (a list). This technique can be used to simplify many different earth science problems. For expedience, I mainly discuss its use in the difficult and arguably intractable problem of estimating rock fluid permeability from rock thin section imagery. To avoid intractability, I concentrate on one part of the permeability problem, the use of my algorithm to summarize the connectivity of rock pores viewed in rock images. By this specialization, I strive to define a means of quantifying connectivity of features in geological images at any scale, although I use the terminology of studies at the thin section scale when discussing applications.

The algorithm I present uses the ideas of Mathematical Morphology as presented in books written (1982) and edited (1988) by Jean Serra at the Paris school of Mines in Fontainebleau, France [47, 15]. Serra's work promulgates the field and helps to standardize its symbolism, which I make use of in chapter 5's discussions. In addition to morphology, my technique also uses more traditional methods of image processing from Blum and Nagel (1978) [4], Sobel (1978) [48], and Cederberg (1979) [11], as well as vital developments in digital rock data and model analysis from Ehrlich (1984) [17], Madden (1983) [36], and Bel-lan and Montoto (1981) [2]. Each of the three parts of the data conversion that I present has its own lineage. The first part I call Fast Montoto Thinning (FMT), the second is Micro Loop Simplification (MLS),

and the third is Cederberg-Sobel Coding (CSC).

FMT is a morphological transformation for binary (0 or 1 valued) digital images. It simplifies image objects, clusters (blobs) of 1-valued pels, for instance, into single pel-wide arcs along the spatial median of the blobs, preserving the blobs' topology in the simplified image result. The roots of FMT are the Medial Axis Transform (MAT) of Blum and Nagel (1978) [4] who recognized the wide utility of representing binary image areas as arcs along the areas' medial axes—with a measure of distance from the arc to the former boundary at every point along the arcs. Bel-Lan and Montoto (1981) [2] presented an algorithm to perform MAT on digital images with a general-purpose computer. FMT is an adaptation of their work to the power and constraints of special purpose computers. Montoto (1982) [38] applied his algorithm to thin section analysis of rock microfractures.

MLS is a picture processing technique that operates on binary digital neighbor-coded images (NCI), where each pel is replaced by the values of its eight nearest neighbors in a rectangular grid. It may be thought of as a way of topological "low-pass" filtering, where MLS maps one NCI into NCI's that are identical in their large-scale connections but simpler than the original to reduce to a linear system (matrix form). The realm in which MLS works was described in a paper by Irwin Sobel (1978) [48], who presented neighbor coding as a contour following technique. His NCI's are used in morphological programming (I used them in FMT) as well as in feature extraction (of which contour following is an example). Although I believe MLS is an original idea, it exists in Sobel's realm.

CSC is a feature extraction technique that operates on NCI's to yield a list of an image's contents. Because of its meticulous nature, it is helpful to simplify the small-scale details of an image that add complexity but are not needed in later analyses of the extracted lists. This need for the simplest representative listing was the reason for MLS. As the output of my technique, these lists may be filtered to yield a linear system of equations for an image of a rock pore network, or used to

extract image features by their size, boundary roughness, orientation, or position in the image. CSC is my hybridization of NCI's with Cederberg's (1979) [11] single pass boundary following technique. He presented an algorithm to follow all boundaries in a binary image by a single pass through the data. That way, the image need not be held in memory but can be streamed through in a raster (TV scan) sequence.

In applying image analysis to rocks, I follow the developments of Ehrlich (1984) [17] who has used pore shape analysis to estimate fluid permeability by regression of image statistics with bulk rock property measurements, gaining some predictive power when given the image statistics alone. But by stressing concern with rock effective properties more than commercial development, I follow even more the work two other investigators. Montoto (1982) [38], used image analysis to characterize microfracture connectivity from thin section data. Madden (1983) [36], used three dimensional resistor lattices to study renormalization of fracture connectivity (resistor disconnections) via a physical model. Both Montoto and Madden have studied the problem of connectivity. Montoto has used rock data in two dimensions, and Madden has used a physical model in three, but both are concerned with the study of fluid flow properties in rock.

I present in this thesis what I believe is a tool for evaluating connectivity. To attack a problem that is susceptible to individual effort, I have restricted my work to two-dimensional and Boolean (0 or 1) data. The techniques I present can be used for either network modeling of a pore image or for the popular textural analysis of pore shapes.

I approach the problem of rock flow property characterization in the following way: acquire digital thin section data and process it to produce a description of how each pore appears to be connected to every other in the image. For high porosity samples, an interconnected network of pores can be evaluated as though it were a resistive circuit, with current imagined to flow from one part of the image, through pores, to another part of the image. For low porosity samples, where pores appear

to be isolated in thin section, Ehrlich's techniques may be used to characterize the texture of pore boundaries (roughness aspect ratio, etc.) and identify classes of pores.

These texture measures can be statistically related to bulk (three-dimensional) laboratory flow property measurements. Both the network modeling and textural approaches require lists of image contents (that I call Pore Structure) for their application. My work describes the steps needed to go from a digital image of pores (1 is pore, 0 is matrix) to lists of either the boundaries of these pores, or of the network described by their medial axes (an image of arcs with width values at every point along their length). The Pore Structure list can be used for geological analysis without further image processing—this is its benefit as a general-purpose image processing tool.

The technique I present, the FMT-MLS-CSC sequence of processing, can be related to Madden's physical model by considering an image representing a slice through his resistor lattice. After processing, the Pore Structure yielded by CSC from this image would be a data structure that could be sorted to numerically express network connections, given any two points in the image between which the connection is to be measured. For a given lattice, the connectivity (resistivity in this analogy) can only be defined between two points. This is analogous to the two places where Ohm-meter probes would be attached in the physical lattice, without which there is not resistance measurement. The data structure that CSC produces can be used to specify these connections for any given point without re-processing the image, only sorting CSC's output list.

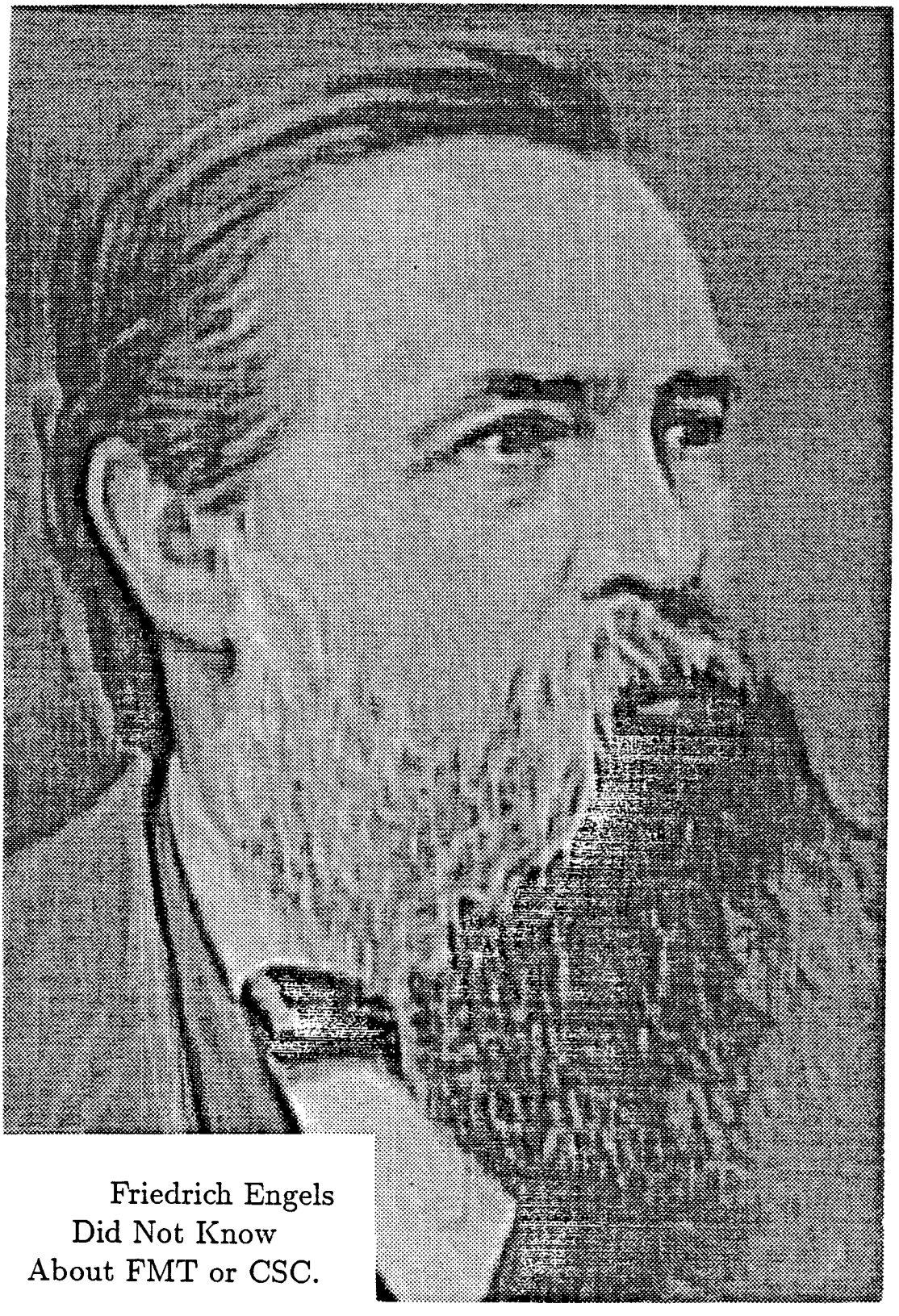
The MLS and CSC tools I present adapt the microfracture image analysis of Montoto to more general classes of pore images. Originally, only the elongated porosity of fractures was thought to be amenable to thinning [2]. By integrating the extraction power of pore boundaries (for low aspect ratio features) and medial axes (for high aspect ratio features), the FMT-MLS-CSC technique can extract Pore

Structure from images with any variety of pores. In the process, not only texture, but also network properties may be evaluated for a given pore image.

I have made special effort to develop processing techniques that improve computational performance without requiring "supercomputer" resources. Where these resources do exist, however, my attention to memory frugality could be extended to form parallel processes and begin to attack the three-dimensional (Boolean) connectivity problem as a parallel set of two-dimensional problems.

In this thesis, I introduce in Chapter 2 the three parts of my technique: FMT, MLS, and CSC, generally in sequence. Since FMT need not always be applied, as when pore boundaries are to be extracted, I separate FMT discussion into one section and MLS-CSC into another. Chapter 2 covers an introduction of the concepts behind FMT, MLS, and CSC, as well as some corollary operations, like the pseudo-inverse of FMT, that make it a useful image processing technique apart from rock pore image applications.

In Chapter 3 I discuss some applications of FMT (without subsequent MLS and CSC use) to rock pores. In Chapter 4 I present some of the reasons for use of FMT. In Chapter 5 I review the foundations of FMT in Mathematical Morphology and discuss alternative approaches to the presentation of morphological algorithms, a meta-discussion of my FMT presentation. In Chapter 6 I turn to the pragmatic side of my tool-building, the computational approach I have used for image processing, particularly in adapting the algorithms to use array processing wherever possible. Chapter 7 contains a discussion of the uses, some untested, that I see for FMT. Chapter 8 contains my conclusions about FMT and some speculation about the uses that await the FMT-MLS-CSC trio.



Friedrich Engels
Did Not Know
About FMT or CSC.

Chapter 2

Pore Structure From Rock Images Via Pattern Recognition

ABSTRACT

This chapter describes a technique to process rock images for automatic production of numerical models based on either dielectric response (pore boundaries) or diffusion equation (pore network) models. The algorithm provides an optimized vectorization of two-dimensional, 0-or-1 valued, raster grids; the result is a succinct description of such images' contents. The vectorized data is intended for use by a specialized routine that solves a linear system of equations, based on the image, in response to boundary condition specifications. The porous rock dielectric response can be modeled from the results of Cederberg-Sobel Coding (CSC) of an image, since it lists the outlines of each and every imaged pore. The diffusion solution utilizes the pre-processing step of Fast Montoto Thinning (FMT)—reducing irregular image areas to median lines of like topology; these are single pixel-wide arcs whose values record the former aperture or width along every point. The diffusion approach treats the image as a resistive mask problem, modeled as an irregular network of a resistive overlay, constant in thickness but varying in width to match the pores. For this problem, FMT is first applied, then CSC is used to encode the thinned arcs—just as it encodes boundaries for the dielectric response problem. CSC's generalized boundary coding allows many topologies to be identified, so that it may extract boundaries, thin lines, or a mix of them, using different pattern recognition templates for each desired topology. This makes the technique a way of symbolically describing heterogeneity in gridded data, common to much of exploration geophysics.

2.1 Introduction

The two algorithms described in this paper transform data from raster form to descriptive forms automatically, for two-dimensional, 0-or-1 valued data, particularly digital rock pore images. One of the algorithms measures the image attributes known from the petrographic image analysis work of Ehrlich (1984) [17], and also models in two dimensions the porous rock dielectric response. Used together, the two algorithms describe the image's contents in a generic form for expression as a linear system of equations, defined by the pattern of pores in the image. This is described in section 2.3. The input (0-or-1) image is treated as a two-dimensional flow system like that studied by Trygstad et al. (1986), but modeled as a resistive mask that matches the pore outlines. This is similar to a two-dimensional slice of Madden's (1983) [36] randomly disconnected three-dimensional resistor lattice model of rock microfractures.

The purpose of these algorithms in the exploration context is to solve heterogeneous diffusion problems as systems of current (mass) balance and voltage (pressure) drop equations set up from symbolic lists rather than from direct processing of a finely gridded system. Using the techniques presented here, the topology of a pore image creates a symbolic description of the pore structure that leads to a linear system of equations.

The first algorithm is a morphological one, from the field promulgated by books written (1982) [47] and edited (1988) [15] by Jean Serra. The continuous-space description of the algorithm's approach is from Blum and Nagel (1978) [4], while the digital image approach is from Bel-Lan and Montoto (1981) [2]. To implement the algorithm on an image-oriented array processor, I incorporated the neighbor coding ideas of Irwin Sobel (1978) [48] with Bel-Lan and Montoto's work to produce the first algorithm, which I call Fast Montoto Thinning (FMT).

The second algorithm is a pattern recognition tool to extract lists of image contents. It is based on Cederberg (1979) [11], and extracts pore boundaries in a

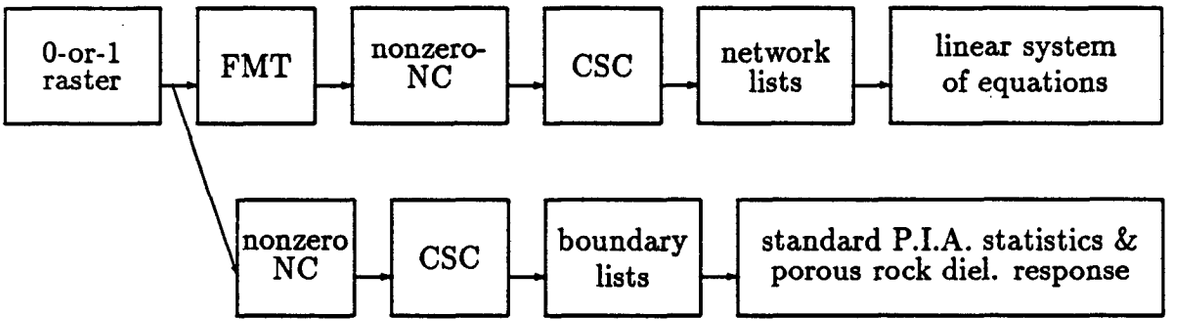


Figure 2.1: The flow of data in the use of FMT and CSC. Sobel's neighbor coding (NC) is a filtering step for input to CSC. If an image is not already binary, a Boolean result, such as a test for nonzero values, is used instead. Network lists lead to a linear system of equation through sorting, and petrographic image analysis (PIA) standard measurements are made on boundary listings.

single image pass. To adapt it to the thinned line images from FMT, I defined a new algorithm to accept input images that are in Sobel's neighbor coded format. The flow of data is diagrammed in Figure 2.1. This new algorithm can be used by itself for modeling porous rock dielectric response (via pore boundary lists) or applied to FMT-processed pore images to be modeled via the diffusion equation as a pore network. I call the algorithm Cederberg-Sobel Coding (CSC).

FMT, discussed first, is a transformation from a 0-or-1 valued digital image of pore areas to an integer-valued gridded image containing only pel-wide arcs with topology like the original pores. CSC, discussed last, is a transformation from a neighbor-coded digital image, (described in the following section) to structured lists of every pore in the image, described in Section 2.3. The complexity of these lists depends much more on the complexity of pores or thinned arcs in the input image than on the fineness of its gridding.

2.2 Fast Montoto Thinning (FMT)

2.2.1 Sobel Neighbor Coding

Both FMT and CSC use as a unifying concept a system of neighbor coding wherein a binary test (one that returns a Boolean true/false value) is made for each of a given pel's eight neighbors on a rectangular grid. The tests are saved in an ordered rotation around the pel using convention outlined by Freeman (1974) [23] and presented as a computational tool by Sobel (1978) [48]. The result is eight bits of data that can be used in place of the original pel in computations on neighbor-coded images.

Efficient use of neighbor coded images means restricting the area of tests to a 3×3 neighborhood. Morphological transforms can require attention to the response for each different neighbor code, so one doesn't readily suggest using 16 independent neighbors—2^8 responses are much easier to define than 2^{16}. What I did in order to implement FMT on an array processor was to adapt the Bel-Lan and Montoto medial line detector sets explicitly to each of the 256 possible neighbor codes.[2, p. 39] I summarize neighbor code responses of both algorithms in the Appendix.

2.2.2 Fast Montoto Thinning

A thinning algorithm transforms a pore image, in my case a binary one, into a simpler image of like topology. The transformed image is computed by reducing the original image features to single-pel wide arcs along their medial axes, at equal distance from any boundary of the feature. The arcs are computed iteratively by removing the outermost layer of pels facing some direction, which changes cyclically with every iteration. Any pel found to be part of a single-pel wide feature is preserved, with the iteration count as its value. This reduction is performed without disconnecting or changing the topology of the object, as illustrated in Figure 2.2. I did not encode Bel-Lan and Montoto's 1981 algorithm for numerical comparison. All my work on rock images was done using the FMT algorithm, which I microcoded for Stanford's Recognition Technology "vision engine". The algorithm is summarized

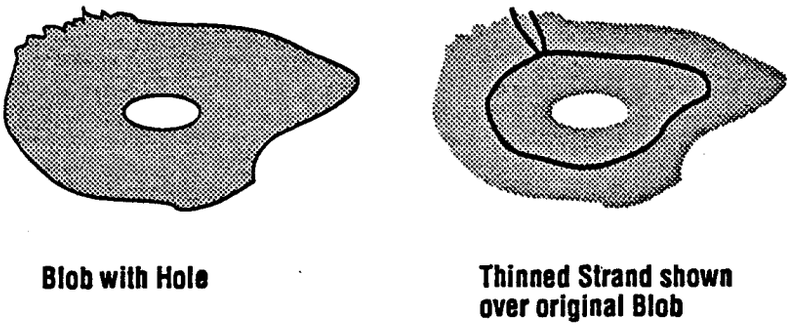


Figure 2.2: The effect of thinning on a blob. The blob on the left has a hole. When thinned, this topology is reflected in the arc that remains. Boundary roughness also generates arcs. These arcs must connect to the loop to preserve topology in the thinned arcs, and also contain width values along their length.

in Figure 2.3. It will be discussed in detail in section 2.2.5.

2.2.3 Fast Montoto Thickening

The 1981 algorithm had a somewhat involved notion of how width values should be saved [2, pp 39–40]. Instead of this, FMT simply records the iteration number at which a pel is saved. No information is lost by this, because FMT iterations proceed four times as often as the 1981 algorithm's iterations. With the north, east, west, south border checking sequence, width w corresponds to iteration z as:

$$\mathrm{w} = (\mathrm{z}+1)/2 \tag{2.1}$$

Normally, this width conversion is not used to replace values along the arcs. The integer value recorded at each point in a thinned arc is the index of the maximally inscribable square's size within the blob at that point. If these numbers are saved, the arcs can be used to recover most of the original shape with Montoto thickening. Montoto thickening is like Mac-Painting[1], using the inscribed square given in

[1]Apologies to Apple Computer

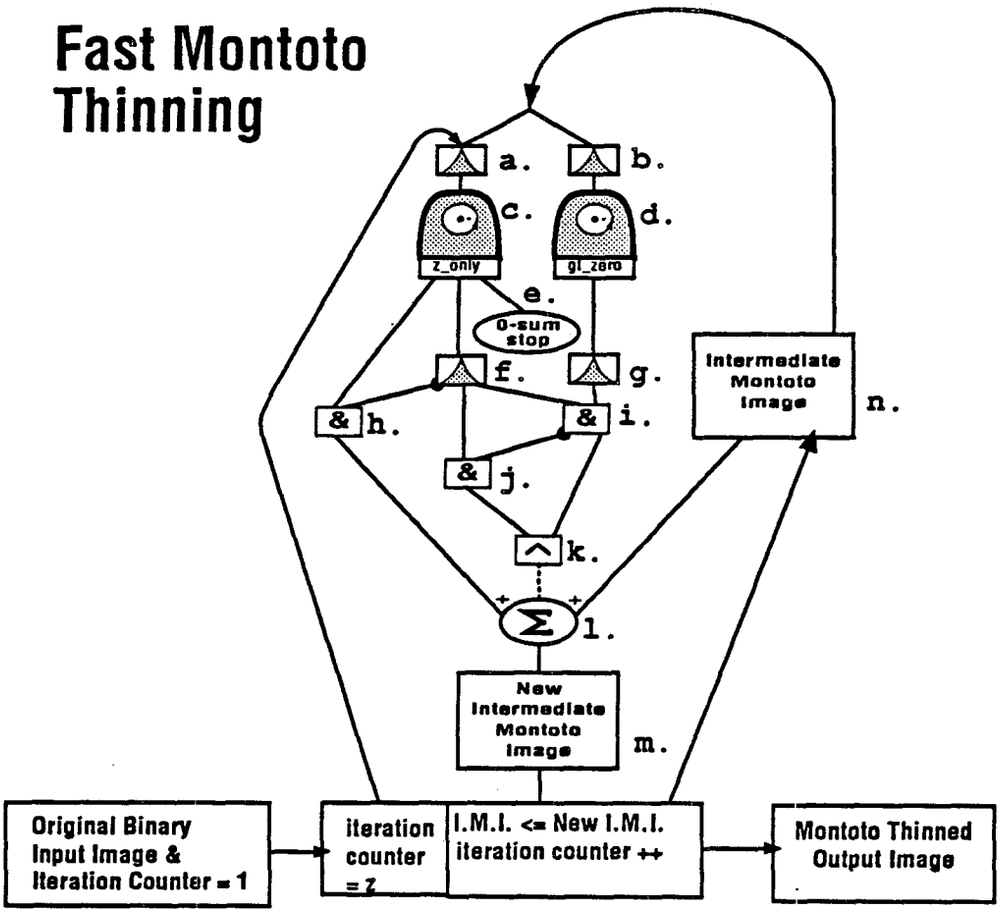


Figure 2.3: The flow of data in fast Montoto thinning. The binary image arrives at the lower left. The iteration counter, $z = 1$ at that time. The image is copied into an intermediate buffer (`n`) and enters the algorithm at the top. Parallel look-up tables (`a,b`) map the input pels into either 0 or 255. Look-up (`a`) is 255 only for pels $= z$, while look-up (`b`) maps any nonzero pel to 255. Parallel neighbor coding routines (`c,d`) accept streams from (`a,b`) to produce `z-only` and `gt-zero` neighbor coded image streams. Time bomb (`e`) counts nonzero pels in `z-only` and terminates FMT when none equal 255, signaling that intermediate image (`m`) should be copied to the output. Parallel look-ups (`f,g`) apply boundary pel checks (`f`) and medial line checks (`g`). Look-up (`f`) applies direction tests in sequence (`north, east, west, south`) from z modulo 4. Parallel logic (`h,i`) flags pels that are active and not border (`h`) and pels that are border and medial lines (`i`). Logic (`j`) flags pels that are border and not medial lines. Control logic (`k`) either preserves a pel if (`i`) is true or zeros a pel if (`j`) is true, otherwise it does nothing. Accumulator (`l`) responds to control from (`k`), and streams image copy (`n`) and flag (`h`) into the iteration's result. If (`h`) is true, the pel is incremented. If time bomb (`e`) is still ticking, output (`m`) is copied to image (`n`), z is incremented and FMT grinds onward.

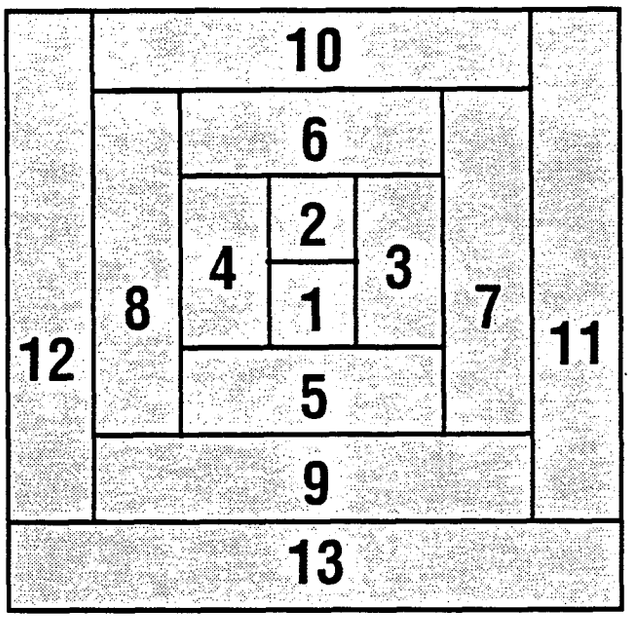


Figure 2.4: The structuring element for Montoto thickening. Pels are stroked with all portions of this template $\leq$ their FMT width value, the iteration at which they were preserved along thin arcs.

Figure 2.4 as a paintbrush to stroke the thinned arcs as paths.

Because I have not incorporated every step that was included in the 1981 algorithm, I call this Fast Montoto Thinning, to emphasize its difference from the Bel-Lan and Montoto algorithm. FMT is not an invertable transformation. Although thinning and thickening with FMT and Fast Montoto Thickening will not recover all shapes exactly, the cycle will return a better representation of elongated shapes than of more rounded objects. As illustrated in in Figure 2.5, an elongated sinuous shape can be recovered exactly while rounded blobs are recovered as a blocky facsimile of their original shape.

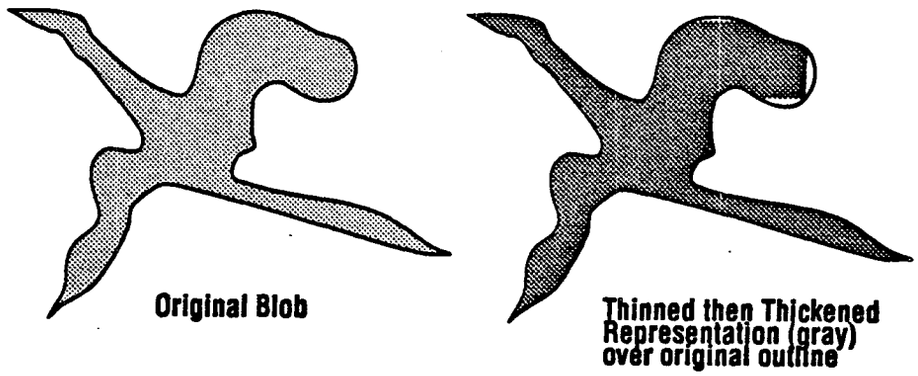


Figure 2.5: The effect of a thinning-thickening cycle using FMT. Elongated parts of the blob are recovered exactly, but more rounded areas become blocky. This effect shows that thinning does not avoid the need for boundary encoding of the original blob, when that blob is not elongated.

2.2.4 Cederberg-Sobel Coding—Background

The end result of the processing sequence I present in this paper is still data, but in a symbolic rather than gridded form. It is suitable for forming a set of linear equations that model the pore image as a resistive network, using the pores as a mask. Topological properties define how many equations are needed to specify the network that represents the image's heterogeneity. To adapt Cederberg's technique to accept neighbor coded input, I mapped each neighbor code value into an unambiguous topological entity that the CSC algorithm could recognize and respond to in a consistent way.

I call my feature extraction algorithm Cederberg-Sobel coding (CSC) since my extensions to Cederberg's work were made feasible by the problem-solving structure provided by Sobel's neighbor codes. This coding is used once as a pre-processing step for all 0-or-1 valued images input to CSC. The CSC algorithm, like image component labeling, is fundamentally a sequential process[45, p.241] since it must

extract features that may be placed randomly in the image.

Cederberg's 1979 algorithm required three scan lines of the image to be held in memory—this same information is made available by a single stream of neighbor codes. The neighbor code steam, however, can be filtered to adjust CSC's efficiency in coding thinned arc topology—which was not possible with the three scan line input. This technique simplifies the equations extracted from the image for diffusion equation modeling of pore networks. Its approach is discussed in the section on metalines.

2.2.5 How to run Fast Montoto Thinning

All steps of the FMT algorithm are described in Figure 2.3 and I refer to its labels in the following paragraphs. I used my array processor to perform neighbor coding in parallel. The Recognition Technology processor that I used does not have hardware (array scatter-gather) to provide the neighbor codes. But bit-slice memory access suffices. The neighbor coding steps (`c,d`) in Figure 2.3 are done by eight passes through the image to build up the neighbor codes. This takes 8/30 s to run on a 512×480 image. Special hardware would be eight times faster.

The FMT result is built up from an initial 0-or-1 valued image, through integer-valued intermediate images at every iteration. The neighbor codes record Boolean tests that I implemented as look-up table values, (`a,b`) that identify neighbors with value $pel = z$, the current iteration count, and $pel \geq 0$, respectively. I refer to the neighbor coded image results of these tests as (`z-only`) and (`gt-zero`). The FMT algorithm has three steps, applied globally, that follow calculation of the neighbor codings that are their input.

(1) The first step I call `bpc-t2c` or border pel check with thinning-2 check, from Bel-Lan and Montoto's terms. It tells whether a pel is on some side of a pore, and is performed by look-up table (`f`). Look-up (`f`) digests (`z-only`) neighbor codes to yield Bel-Lan and Montoto's equations (1,2′,3′,4–7) [2, p.39] in one step. The

directional test sequence north, east, west, south, corresponding to their equations (4–7) is expressed as a sequence of four look-up tables of 256 byte length, chosen by z modulo 4.

(2) The second step I call `mlc` or medial line check procedure. It is a look-up table filter of neighbor codes that applies Bel-Lan and Montoto's eight medial line templates [2, Fig.1,p.39] in one step. This step depends on the (`gt-zero`) neighbor code image. I present the `mlc` look-up table in the Appendix, as those neighbor codes labeled "m".

(3) The third step I call `iip` or increment interior pels procedure. It is a four-part step involving (`h,i,j,k`). The fate of a pel that is saved on an arc is decided by (`k`) from both (`i`) and from (`j`). Test (`i`) flags pels that are on a medial line and part of the iteration's border set. Because (`g`) works from (`gt-zero`), it finds medial line pels saved at any iteration. Because (`f`) digests (`z-only`), border pels are found only on the actively ablating borders of the remaining pores, remembering that FMT is iteratively removing most outermost pels. Thus, (`i`) marks which pels are both on medial lines and active border, which means they must be added to the arcs they connect to. Meanwhile, test (`h`) marks pels that are part of the active area and that are *not* part of the border set. These are the so-called interior pels which give step (3)its name. With its identifying flag, (`h`) is used to increment by one the active feature area so that it can be identified in the next iteration. Test (`j`) identifies pels that are border pels for iteration z and not on a medial line. These pels are zeroed. The control test (`k`) listens for flags from tests (`i,j`). Normally, it does not disturb accumulator (`l`) as it combines, pel by pel, the intermediate image and the occasional $pel = pel + 1$ signal from (`h`). On signal from (`i`), (`k`) will preserve a pel as part of a thinned arc. On signal from (`j`), it will zero the pel. The results of from (`l`) become the new intermediate image (`m`).

At the end of each iteration, the total number of pels identified in `z-only` is summed at (`e`); if zero, FMT has finished. The new intermediate image is shipped

out as the result. If any pels equal z, then another iteration will be started. The new intermediate image is copied from (m) to (n), $z = z + 1$, and so it goes.

2.3 Cederberg-Sobel Coding

Cederberg (1979) [11] described a way to follow binary image boundaries in raster scanned data, rather than holding image data in memory throughout the scanning process. Sobel (1978) [48] described a way to follow contours using neighbor code images. I have synthesized a version of Cederberg's work that follows boundaries and thinned lines, using Sobel's principles. I refer to the new algorithm as Cederberg-Sobel Coding (CSC). It produces two types of output listings. One output list contains the coordinates of nodes—either the last pel in a raster scanning (left to right and top to bottom) of the pore, or an endpoint of a thinned arc. The other list contains paths—either a boundary around a pore or an arc between two nodes. The exact form of this output, a data structure, is not specified here, since it is application-dependent. The information needed for extracting these lists a single pass through the neighbor coded image is described in this article.

CSC's maps the 256 possible neighbor codes into four classes: nodes (arc endpoints, and pels where three or four arcs meet), pnodes (meaning pseudo-node, any max- or minpoint where two arcs meet), links (any pel on an arc but not a node or pnode), and interior pels (residing within blobs and surrounded by arcs). Interior pels are not coded by CSC. The elements sort well into a periodic table, shown in Figure 2.6. The mappings from Sobel neighbor codes to these topological classes is presented in the Appendix.

2.3.1 Metalines

Since CSC's output is meant for systems of linear equations to model the a pore image as a network, needless complexity must be removed from the output lists wherever possible. To do this, I devised a neighbor-code to neighbor-code mapping

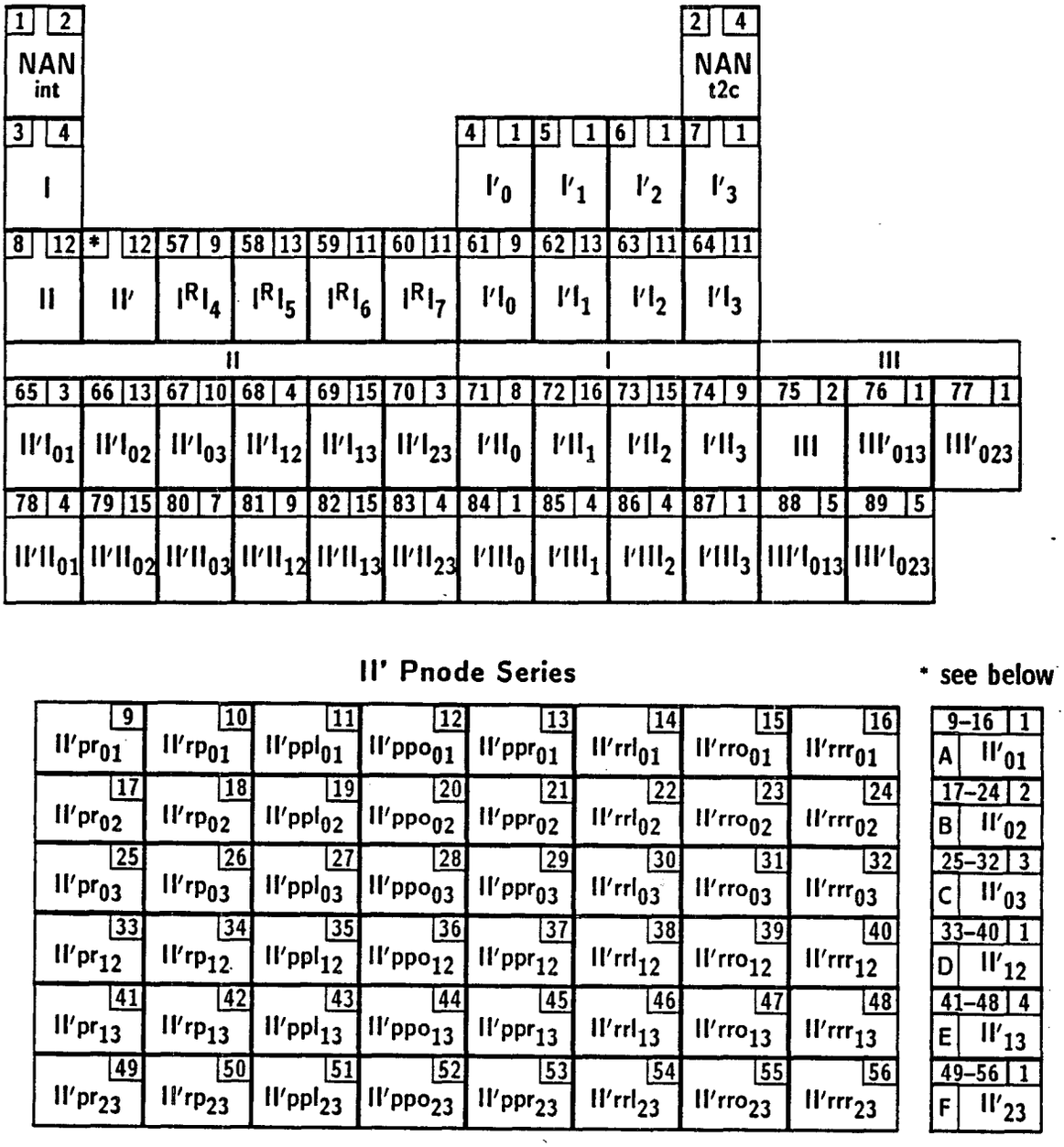


Figure 2.6: A periodic table of topological elements for neighbor codes input to Cederberg-Sobel compilation. Small upper left boxes contain serial numbers, upper right boxes the number occurrences of that type in the 256 total neighbor codes. The first row are interior points with no connecting arcs. The second row are endpoints, with one connecting arc. The third row are pnodes and links, with two arc connections. Types 9–56 are all type II′ pnodes, in a separate tabulation. Types 57–60 exist only by reversing lists of types 61–64. The fourth and fifth rows are nodes with three and four arcs connecting. Names give arcs arriving, a prime (′) symbol, and the number of arcs leaving. The rest of the name gives what links are appended at minpoints, from the Cederberg directions {0,1,2,3}, meaning {east, south-east, south, south-west}.

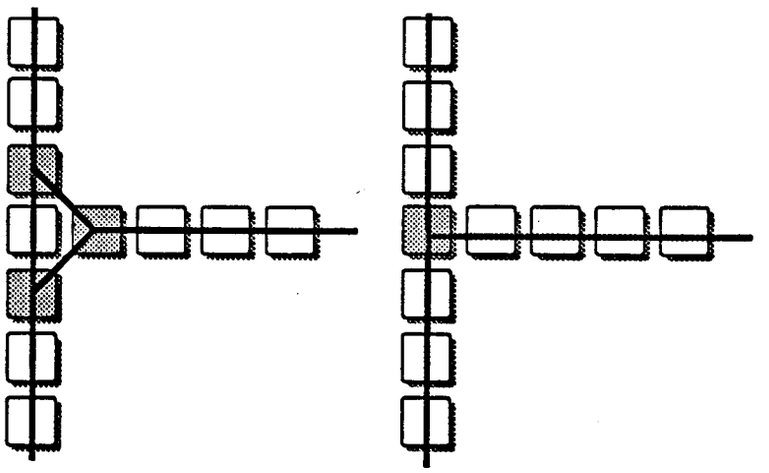


Figure 2.7: The need for metalines: Nodes shown in gray, arcs shown as black lines. On the left, one topology requires that the 'T' arrangement recognize diagonal paths wherever they exist. In this context, however, it is far more economical to describe a system of equations for the four way topology on the right. For the pels' neighbor codes, a context adjustment can make the neighbor codes of the three node pels in the left arrangement blind to their diagonal neighbors. If this is always done, the one node on the right can be recognized, even if the image is generally eight connected.

that filters the input neighbor coded image into a simpler topology. This step I call the formation of metalines, since it produces an image that describes arcs with a filtered neighbor coded image, rather than a global neighbor coding of the arcs themselves. This simplifies a common problem, shown in Figure 2.7.

Neighbor coded images are ideally suited to a context check (are two nodes adjacent?) of neighboring nodes as a look-up filtering of *neighboring* neighbor codes. When passed through a look-up table to flag nodes, the neighbor coding that results is a coding of context information in a 5×5 neighborhood, compressed into a standard (8-bit) neighbor code. Then, two adjacent nodes may have their diagonal

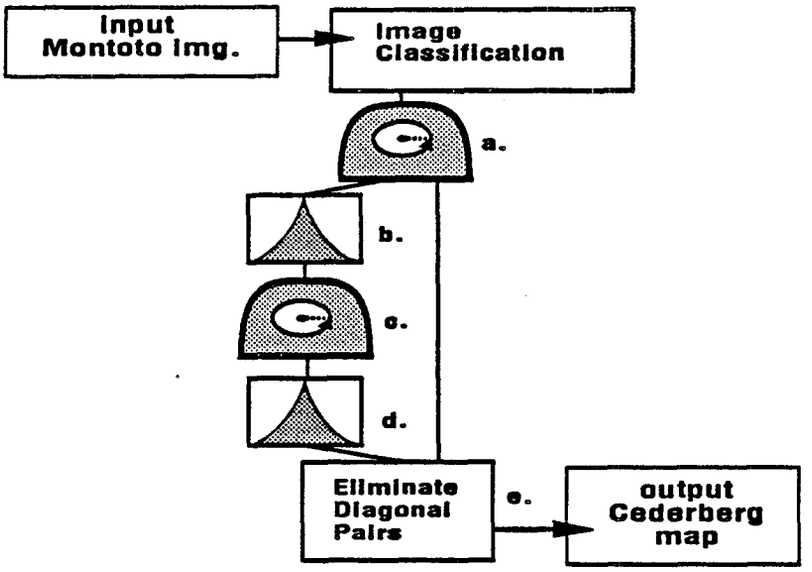


Figure 2.8: The technique for forming metalines begins with an image of thinned arcs, or arcs with blobs. Image classification yields a binary mask for the FMT result. Neighbor coding is applied to this binary at (a). Its result is filtered to flag all nodal classes with a look-up version of Appendix A. This new binary is neighbor coded at (c), then filtered through a look-up (d)to identify clusters of nodals that can be simplified. Diagonal connections are nulled in (e) for nodes adjacent to two other nodes, for south-east and south-west neighbors, and between south and east neighbors to yield the output.

neighbor code bits nulled. Altering a pel's neighbor code may remove the pel from the node class, without changing the accuracy of the output list.

The calculation of metalines is inherently sequential, rather than global. The changes made in the neighbor coded image change a pel's nodal nature, and affect subsequent actions. The two neighbor codings that are used as input to the process may be computed globally.

The range of adjustments possible with metalines is illustrated in Figure 2.9. Graphical representations of the neighbor code values in a 5×5 neighborhood emphasize the simplifications possible for a pathological arrangement of 8 binary arcs converging to one pel, or four lines crossing at a pel. The problem was presented in

reference to thinning by Rosenfeld and Kak (1982) [45].

2.3.2 CSC's Basic Ideas

In terms of the output listing format, nodes can retain full coordinate information, but their arcs need only be chain-coded. The chain code represents an arc path as an offset from the previous pel, rather than full image coordinates. For numerous arc pels, this results in a more compact output listing, if needed.

Minpoints create challenges, although they do not appear in output files. Topologically unimportant arc minima, these pnodes must be treated specially in a raster scan, with a concern for their genetic history. They are required to tie together seamlessly the two arcs that meet at that pel in order to produce a single arc in the output files. Doing this correctly requires selecting one of eight possible responses. The response depends on what kind of nodal (node or pseudo node) is at the far end of each arc, and thus what must be linked together. This linking modifies the output listing. parenthetical text statement.

When minpoints tie together two arcs, they will usually append some links to one arc, taken in reverse order from the other arc. Reversing the order of links in an arc list represents the four non-causal scanning directions [11, p. 233]. In other words, although a left-to-right, top-to-bottom scan can only follow arcs in the east, south-east, south, and south-west directions, an arc leading south-east, when reversed, will appear to lead north-west. In this way, all arc directions can be represented by a raster scanning process.

I use a simple technique to name CSC's nodals. The number of arcs arriving (downward) at a pel is given a primed Roman numeral like I′, II′, or III′. The number leaving the node is given a Roman numeral I, II, or III, as illustrated in Figure 2.10. The action taken at nodal pels affects CSC's directed list of arcs. This directed list is not part of the output file set. Rather, it is an internal bookkeeping device commonly used in compiler programs. Its purpose is to mediate the disparity

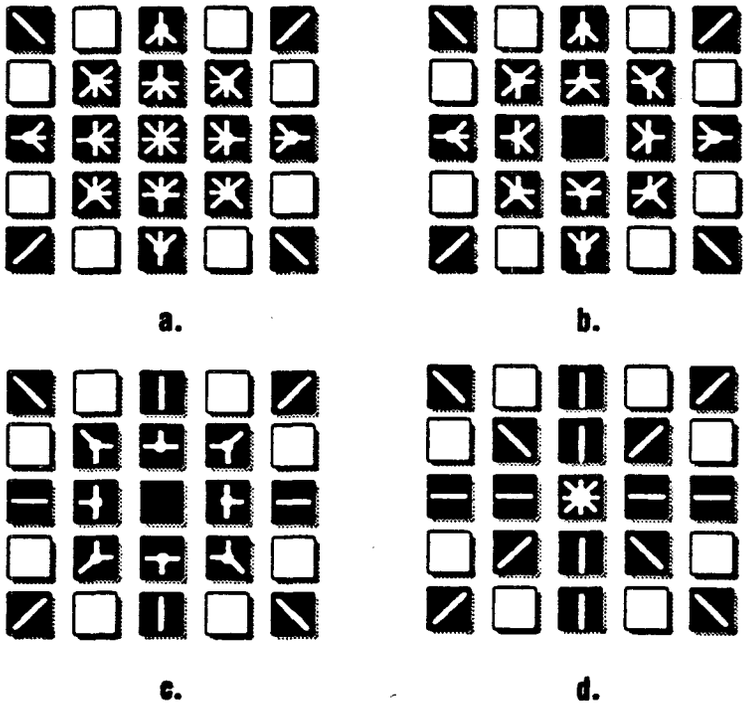


Figure 2.9: Here metalines are put to the task of Rosenfeld's conundrum, where 8 arcs arrive at a single pel. Each block represents a 5×5 window of a neighbor coded image. All four blocks show the same eight arcs in black. The neighbor code values are represented graphically as white sticks leading toward each encoded neighbor. Nodal neighbor codes are given a dot at their center. (a) This block illustrates the neighbor codes returned by a global process or by hardware. All arc pels except the four corners and very center are nodes, and eight triangular loops abut the north, west, east, south arcs around the central nine pels. (b) The central pel neighbor code connections are nulled to illustrate the redundant 4- and 8-way connections that must be considered by a boundary follower program. (c) This block shows neighbor codes from b. after micro loop simplification. Diagonal neighbors exist but are not neighbor coded, greatly simplifying the topology but still coding the same arcs. (d) This block illustrates how metalines would support a topology that allowed 8 arcs/node. All redundant neighbor connections are nulled. One node remains.

between data flow, where pels arrive in raster sequence, but output lists must be appended asynchronously. The maintenance of a linked list of active arcs is an implementation detail that is treated well in compiler texts such as Tremblay and Sorensen[51], but which I will not discuss much. A I'II node pel requires that it be added to the list of nodes, that the currently active arc be terminated, and that two new arcs be created in its place, with the last newly created arc remaining active before proceeding to the next pel.

When discussing unodes, the troublesome II'nodals, it is simpler to always call the lower-numbered arrival direction "left" and the higher-numbered one "right," as an adjective for either the arc or what it connects to. The left and right arcs are connected to nodal entities. What these left and right nodals are, and where they are in the image, together determine which of the eight actions is performed.

2.3.3 What to do at a Unode

Usually, both arcs arriving at a unode are removed from CSC's linked list of active arcs. That way, they will not be expected to have pels appended to them on the next scan line. Sometimes, as described below, they are removed from the linked list but one survives and is laterally linked to some other active arc through cancellation of a type II pnode. For the sake of discussion, I call type II pnodes by the name pinodes as in πnodes and call type II' pnodes by the unodes moniker. The surviving arc is extended through the unode, through a pnode at the top of the other arc, and then down the pnode's other arc to an actively growing end. This is illustrated in Figure 2.11, with the II'ppl example in the middle of the figure.

If the left and right arcs connect to real nodes, three algorithmic choices exist. The nodal type is II'rr for real and real, in the top row of Figure 2.11. First, if both are real, what are the serial numbers of the nodes at the far end of the arcs? If both arcs attach to the same node, they will form a loop when joined, type II'rro as in real to real loop. Second, if the left node's serial number is lower than the right

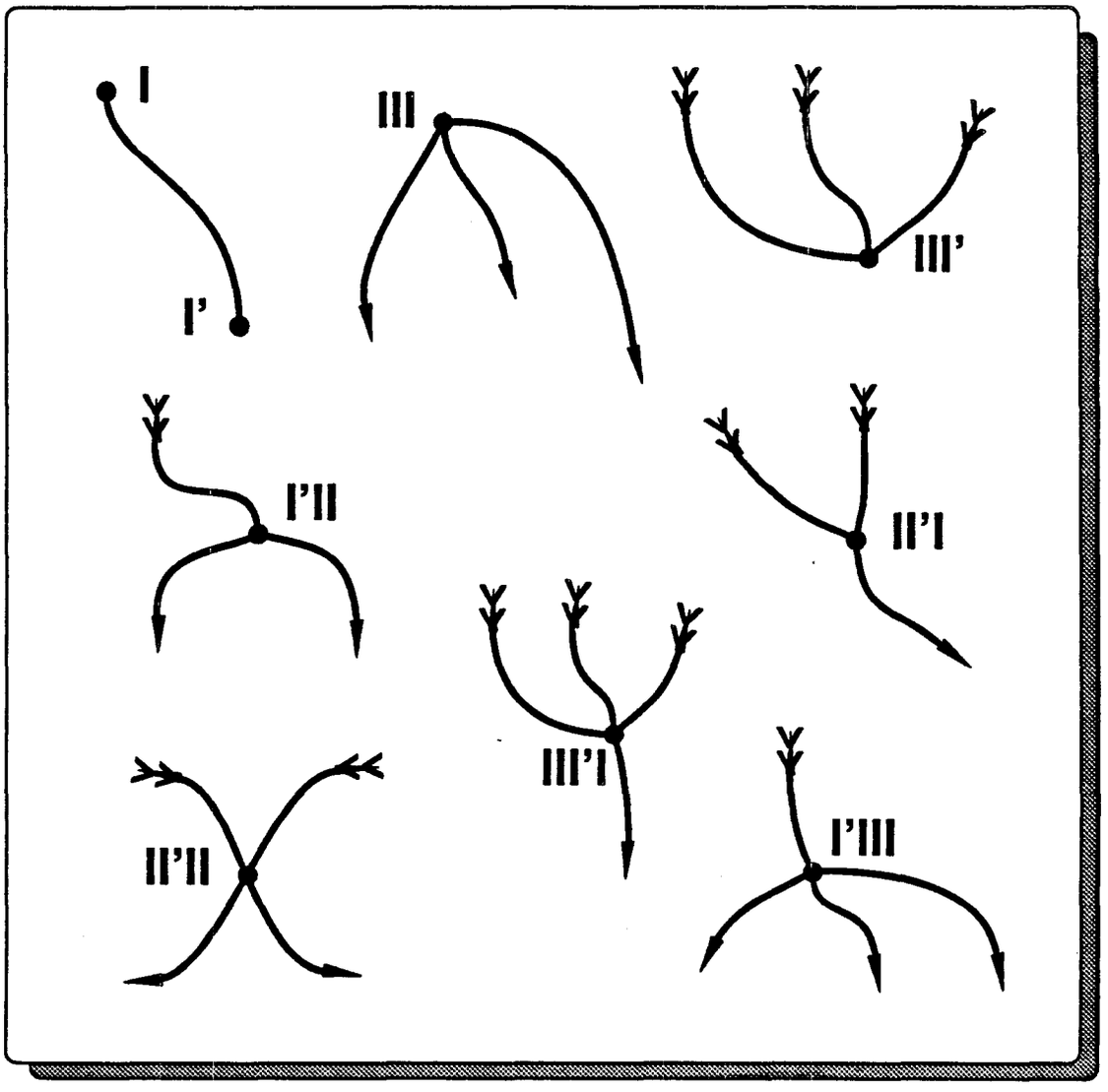


Figure 2.10: The nine node topological types are shown here. The top row shows the endpoint types I and I'. These begin and end single arcs, and are considered complete nodes since they do not require action other than starting or ending an arc. The other complete nodes are types III and III', also in the top row. The five compound nodes are shown below that row, and consist of types I'II and II'I, where three arcs arrive at a node pel, and types I'III, II'II, and III'I, where four arcs arrive at a node pel. All other nodals besides these nine are pnodes, and have only two arcs attached, just like any link along an arc. Pnodes are shown in Figure 2.11

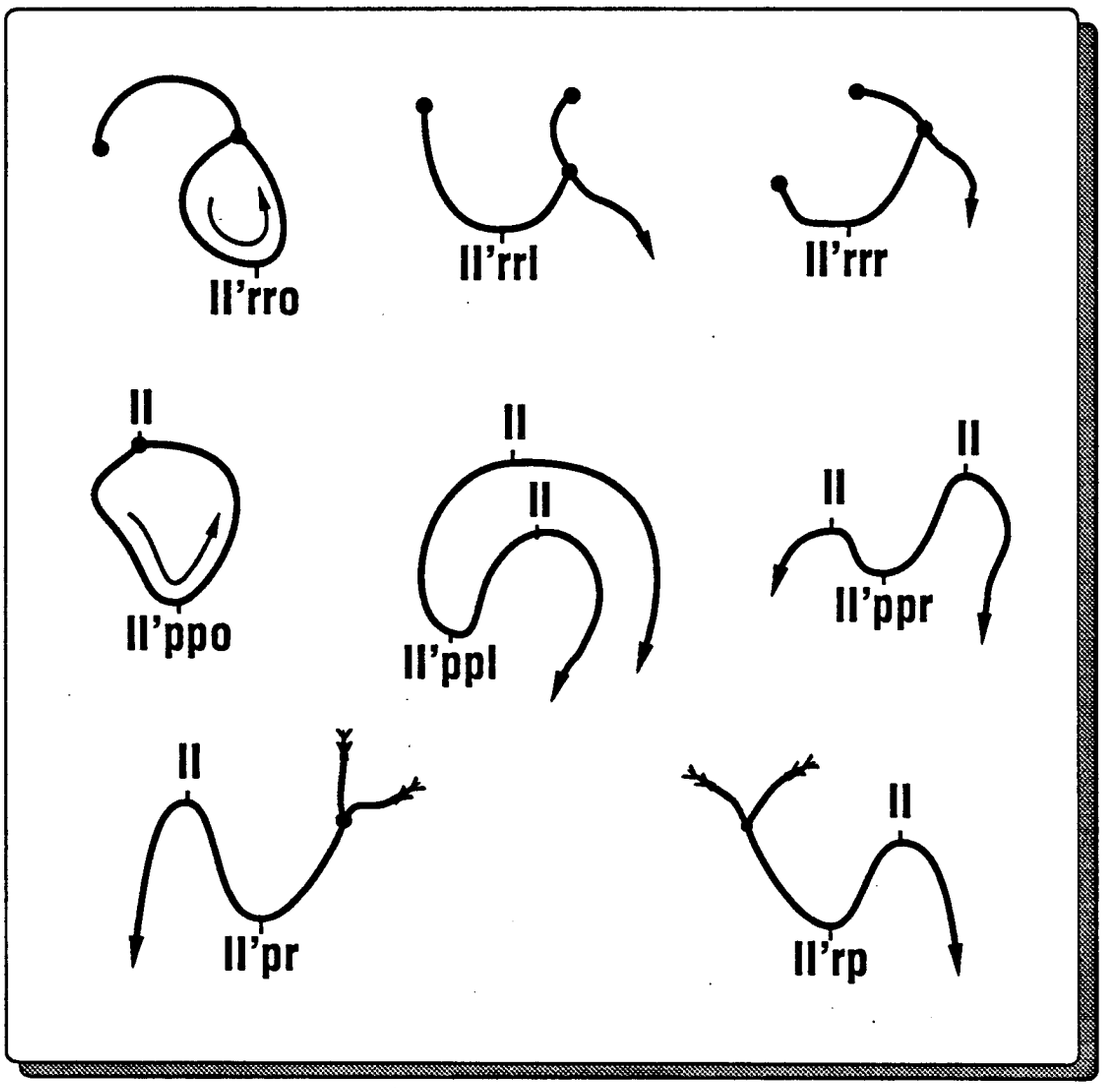


Figure 2.11: The nine pnode topological types are shown here. Type II, also called a pinode (πnode), is shown at the maxpoints in the middle row. It occurs at pels where two arcs are started in raster coding, but is not important in, nor should it affect the output listing. The top row shows three of the eight II′ type pnodes, all of which are also called unodes. The top three connect to nodes (rather than pnodes) on both their left and right arcs. Type II′rro forms a loop below a real node, type II′rrl and II′rrr connect two nodes where the left and right node, respectively, are higher up in the image. In the middle row, types II′ppo forms a loop around a blob, where both left and right arcs started from the same pinode. Types II′ppl and II′ppr join two pinodes, with the left and right pinodes, respectively, higher in the image. Types II′pr and II′rp in the bottom row illustrate unodes that join a real node on the right and left, respectively, and a pinode on their other arcs.

node's, the left node was found earlier and its arc, by CSC convention, defines the forward direction. Append the left arc with a reverse listing of the entire right arc, and record this connection in the records of each node. This is type II′rrl for real to real with left higher. Third, if the right node's serial number is lower, append the left arc, reversed, onto the right arc and record the connection in the left and right nodes, this is type II′rrr, for real to real with right higher.

As with real nodes, if the left and right arcs connect to pnodes, three choices exist. They are illustrated in row two of Figure 2.11. Firstly, if both left and right pnodes are the same, not only do the arcs form a loop, they define a complete boundary. The pnode record at the top is converted to a real node's record in the output files and the arcs are joined. By convention the right arc is reversed and appended to the left arc, forming a counter clockwise loop at all type II′ ppo pnodes, named for pseudo to pseudo loop.

Secondly, if the left pnode is further up the image than the right, then the left pnode is left intact, and further, more involved changes transpire. The right pnode, because it is further down the image, is subordinated by the left pnode. As shown in type II′ppl (ppl symbolizes pseudo to pseudo with left higher.), the lower of these pinodes ([) is connected by the unode (]), and one pinode ([), the top one, remains. Symbolically, ([] [) becomes ([). The arc that went through the unode is extended by reversing the right arc up to the lower pinode, then appending that pinode's other arc onto the newly extended one, and changing the ownership of this arc in CSC's linked list. The right pinode can then be recycled. What was three arcs is now one. Where once were two pinodes and one unode is now one pinode. The similar situation, trading the unode's left for its right, holds for type II′ppr (ppr symbolizes pseudo to pseudo with right higher.).

Thirdly, if the unode is connected to one node and one pnode, then the pnode is canceled as explained in the preceding paragraph, and a single longer arc is created from the three arc segments. The node remains with its arc still active, to be

appended in later scan lines. These are called types II′ pr, for left pseudo, right real, and II′ rp, for left real, right pseudo connections. These complete the set of eight possible actions at unodes. Fortunately, pinodes need only start two arc records. All resolution of nodal connections takes place at unodes.

2.3.4 What to do at a Node

Some nodes simply start one arc or three, or end one arc or three. These are types I and III, and I′ and III′. I call these the *complete* nodes, since they evoke a one-time action from CSC and then are complete. They do not cause the same bookkeeping problems as pnodes. CSC's action at a complete node is primarily to create a node in the output index file, save relevant information such as the grid coordinates and thinned arc width in that record, and arrange to record the serial numbers of arcs that end at that point. The node records also contain what other nodes they are connected to at each arc's other end, if the arcs' ends are known at that time. As such, complete nodes evoke a change in CSC's internal directed list of active arcs, and may write arcs to output lists, if it ends them. The directed arc list is the queue of all active arcs. As CSC scans across pels which are edges or thin arc links, it adds each to the active arc and advances along to the next arc in this linked list, unless the link just added is in the east direction, in which case CSC does not advance the directed list until it reaches a node or link other than one in the east direction.

Some nodes are both maxpoints and minpoints, in that they end some arcs and begin others. I call these *compound* nodes. Since all compound nodes end some arcs and start others at the same pel, they can be classed in the same terms as other nodals. Since arcs are ended before new ones begun, I make the number of arcs ended the first part of the node's name. One can not properly begin a new arc, adding it to the directed arc list and starting a new queue in the database, before making all the database changes required by the arcs that are being finished.

Two compound node types end only one arc, types I′II and I′III. Two compound

node types end two arcs, types II'I and II'II. In the CSC topology, Type II'III is not allowed because it requires five arcs to be connected at one pel. Five arcs may meet, but in CSC's topology they must do so at at least two adjacent nodes, where one of each node's four possible arcs must connect to its neighbor node. Finally, there is compound node type III'I, ending three but starting one arc.

There are four types of complete nodes and five types of compound nodes, for nine types of real nodes. How many pnodes are there? One type of pinode, and eight types of unodes. There are nine types of pnodes, and nine types of real nodes, for a total of 18 nodal entities.

2.3.5 How CSC is run

Because it treats the raster image as a time series, CSC must know the width of the input image, to control its ability to recognize vertical lines. For extracting features from FMT output images, the input to CSC is a 16-bit image input. A neighbor code image is used to control the identification of nodes and loops, while the FMT image provides the width data along its arcs.

I present two graphical depictions of the CSC algorithm. In terms of CSC's implementation in code, Figure 2.12 is more precise, but in terms of control flow, Figure 2.13 is more accurate. For simplicity, the branches of the `EndStrand` routine are not shown in Figure 2.13.

The start of CSC is most easily seen in Figure 2.12. At the point labeled "IN", control moves to point 1, which is the main program. It calls point 2, which opens the input files and allocates memory for the program to run. The names of the routines symbolized by these points are given in Figure 2.12's caption. For simplicity, Figure 2.13 does not show the initialization steps, but rather shows the input files already streaming data into CSC.

After initializing, CSC repeatedly moves to point 3, and obtains a pel from the input files, using the routine `GetPel`, as shown in both figures. Control then moves

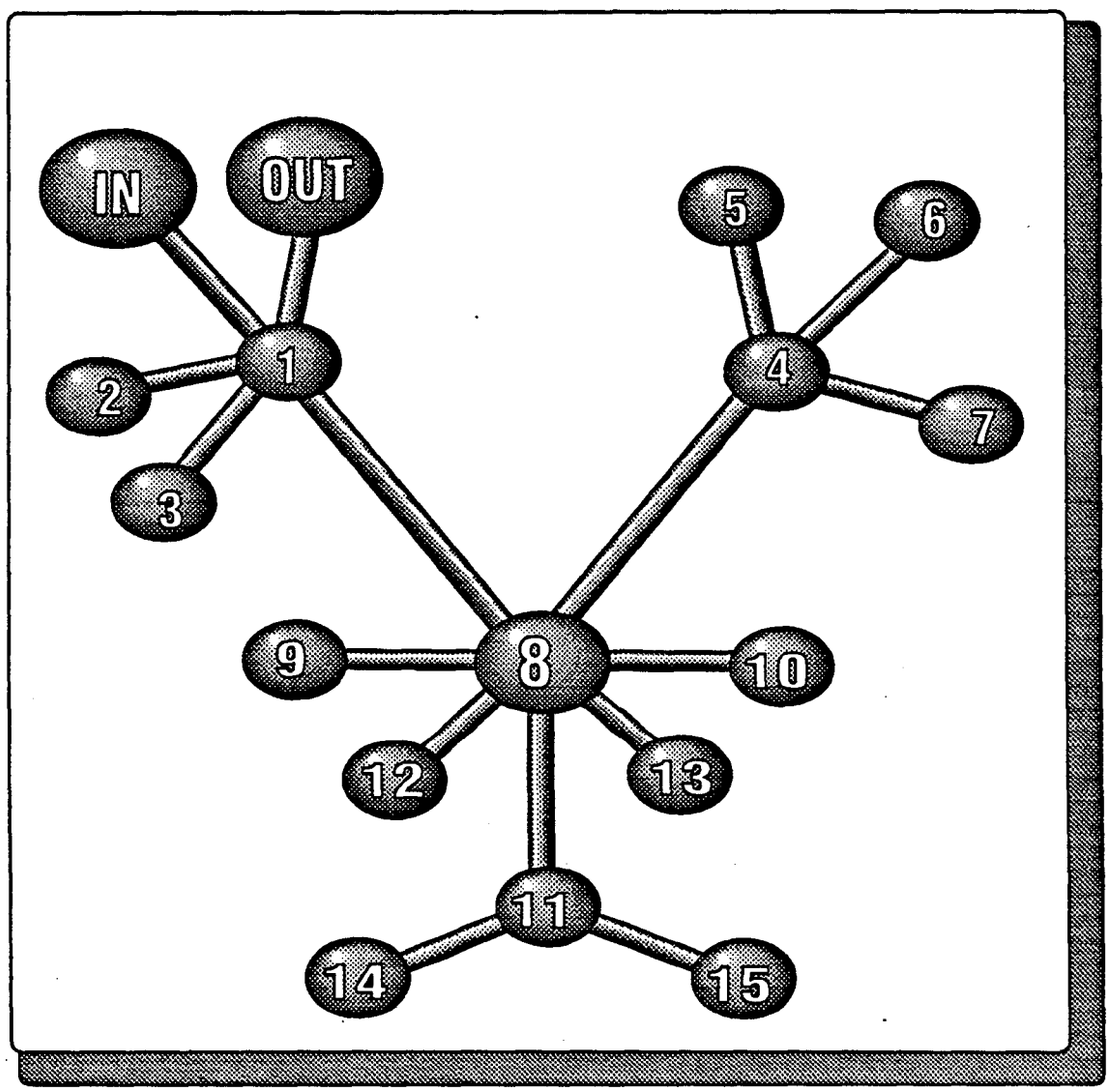


Figure 2.12: The symbol above represents the Cederberg-Sobel Coding algorithm's control flow. The input, IN, invokes the main CSC program at point 1, called `CederSobel`. Point 1 calls initialization routines at point 2, `StartImgScan` then repeatedly calls point 3, `GetPel` to accept input pels one at a time. These are passed to point 8, `DoTheScan`, that handles minpoint functions. In this action, it may call points 9, 10, 11, 12, or 13, known as `CloseNode`, `MakeNode`, `EndStrand`, `MakePnode` or `AddLink`. Point 11 may need to call points 14 or 15, `CloseNode` and `EndPnodeStrand`, independently of other calls to CloseNode. If a pel is a maxpoint, or is a minpoint but also a maxpoint, control passes from point 8 to point 4, `ContinueTheScan`. It may call points 5, 6, and 7, known as `MakeNode`, `StartStrand`, and `MakePnode`. When the entire image has been scanned, control passes from point 1 to OUT, after calling point 2 as `EndImgScan` to free memory allocations and write output files.

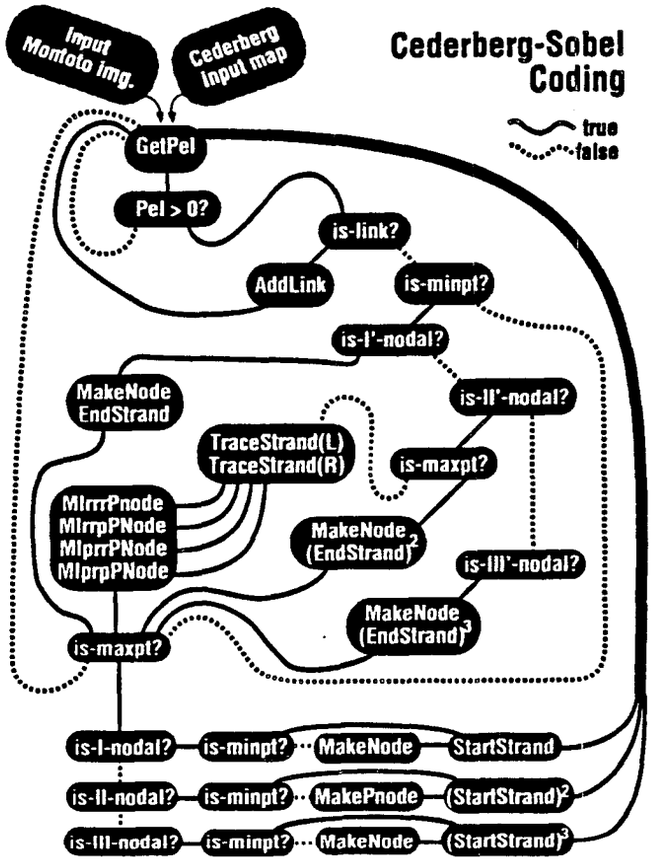


Figure 2.13: This figure represents a schematic diagram of the logic used to run Cederberg-Sobel Coding. Details of `EndStrand` and linked list of arcs have been excluded, but the action of point 12 in Figure 2.12 has been expanded. The input images from FMT and its neighbor coded map are shown as ovals at the top of the figure. `GetPel` repeatedly accepts input pairs from these files, and tests the input map to see if the pel is nonzero. If so, it uses a look-up test to see if the pel is a link. If so, `AddLink` appends the current arc and may step CSC's internal linked list to the next active arc. Control passes through minpoints as described in the text. If a pel is a II′ minpoint and not also a maxpoint, it is a so-called unode, and both strands are traced by `TraceStrand`, then one of four actions is requested from `MakePnode`. After managing minpoint tasks, any maxpoint tasks are attended to, if needed for that pel. If it hasn't been already, `MakeNode` is called, unless a pel is a type II maxpoint and is not also a minpoint, in which case it is a so-called pinode, and `MakePnode` is called. After any required minpoint and maxpoint actions are taken, `GetPel` is called once again and the process repeats until the entire image has been scanned.

to point 8 in Figure 2.12, which operates by the logic shown in the middle section of Figure 2.13, below and to the left of the thick curve connecting three points in the bottom of the figure to `GetPel`. This logic handles all minpoints, where arcs end or meet and output lists are appended. The routines called by point 8 are shown below and adjacent to it in Figure 2.12, and shown by name in the middle part of Figure 2.13.

If a pel is a maxpoint, then it is at the start of one, two, or three arcs. The control passes to point 4 in Figure 2.12, which represents the lowest three rows of Figure 2.13. Here, a new node record will be started, unless the pel also was a minpoint and a node was created for that purpose. Then, a new arc will be added to CSC's linked list so that its pels on the next scan line will have a record in which to be entered. This action is realized in the logic by testing whether the pel should have one, two or three arcs, based on its neighbor code, and then whether it is also a minpoint, since all minpoint actions are handled before maxpoint actions in the CSC algorithm. With a node record guaranteed to exist, one, two or three arc records are started by entry into the linked list, and control returns to point 8, then to point 1 in Figure 2.12, from which point 3 is called. This is summarized in Figure 2.13 by a return to `GetPel`.

The logic of point 8 in Figure 2.12 can be grouped into whether the pel is a link or a nodal, and whether any nodal ends one, two, three or no arcs. If a link, the active arc is appended with the type of link the pel represents.

If the pel is not a link, the logic of point 8 checks to see if it is a minpoint for one, two, or three arcs. If it is none of these, it must be a maxpoint since it is not a link. Control passes to point 4 in Figure 2.12, representing the lowest three rows of Figure 2.13, as described above. If the pel ends only one arc, it is a I′ nodal, and a new node record is created and the arc is closed by writing it to the output files. The bookkeeping at this point is performed by the `EndStrand` routine, shown as point 11 in Figure 2.12. It will selectively close the node record just created, if all its expected

number of connections have been made. Also, `EndStrand` may pass control to point 15 in Figure 2.12, which is not shown in Figure 2.13 for simplicity. That routine, `EndPnodeStrand`, connects the minpoint to other nodes via an intermediate pnode, which requires that both arcs of that pnode are joined to form a single arc. Since `EndStrand` is called as many as three times (at type III′nodes,) and any one or all of those calls may require a call to close a node record or end an arc started by a pnode. These possibilities form the bottom three points of Figure 2.12.

Point 8 ends arcs. For a type II′ minpoint that is not also a maxpoint (i.e. a nefarious unode), control passes to point 12 in Figure 2.12, which will trace each arc and perform one of the eight actions required at a unode. The response is to call the routines `MlxryPnode`, where the `x` and `y` are either r for real or p for pseudo nodals, on the left and right arcs. For this particular action, Figure 2.13 displays more detail than Figure 2.12. The two boxes containing `TraceStrand` and `MlxryPnode` calls in Figure 2.13 together represent the single point 12 in Figure 2.12.

After finishing its minpoint tasks, point 8 checks to see if the pel is also a maxpoint. If not, it returns control to point 1 and then point 3 is called. If so, then control passes to point 4 and the maxpoint actions proceed as described. (It will not be necessary to call `MakeNode`.)

This procedure repeats throughout every scan line in the input images. At the start of every new scan line, CSC's linked list is re-started at the beginning, by setting a pointer variable to the beginning of the directed list. Some error checking makes sure that this pointer has traversed the entire linked list by the end of every scan line, to be certain that every arc that is in the list had some links appended to it. Then, with the pointer at the record for the leftmost arc, the next scan line in the image is begun. This manipulation of the linked list pointer is under the control of the main program, point 1 in Figure 2.12, and is not shown in Figure 2.13. When the entire image has been scanned, point 1 passes control "OUT" of CSC to end its run, as shown in Figure 2.12.

2.4 Summary

The advantage provided by Cederberg-Sobel Coding (CSC) to geophysical exploration is simple but important—it is an automatic tool for extracting symbolic descriptions of digital rock images given a stream of neighbor-coded data. I have begun to apply this to thin-section images where pores are 1-valued and matrix is 0-valued, but as Sobel (1978) [48] described, neighbor codes of real-valued images can be used for contour following. As such, CSC may find use extracting features from borehole imaging tools' data streams, to be passed as descriptive lists to general purpose feature detection or classification routines as part of a pattern recognition study. Still another use, with suitable extensions, may be the listing of features in three-dimensional data, processed as parallel two-dimensional feature extraction problems. Because CSC accepts image data sequentially, many CSC processes could be synchronized by scan line in the data, and processes may communicate to adjacent processes that a feature has begun or stopped and must be followed in another image plane. Without this ability to synchronize processes, feature extraction codes can not run deterministically in parallel, for they would not be constrained to work on adjacent parts of the data volume. This extension would make CSC useful in serial section pore description, and by implication, may provide a foundation for seismic data volume feature extraction.

The advantage to exploration of FMT is that, together with CSC, high resolution digital rock images can be automatically reduced to simpler numerical models. The complexity of what is in the image, rather than its grid size, controls the complexity of diffusion models based on FMT processing followed by CSC extraction. Changing boundary conditions then requires a re-sorting of the descriptive lists, rather than re-processing the gridded data. Particular applications may have different demands on the data structure output from CSC. The author believes the generic solution of pore network problems based on solutions to the diffusion equation is presently the most demanding application yet made of pattern recognition techniques applied to

geophysical concerns. The systems that must be solved after processing a pore image with FMT followed by CSC are far simpler than the original heterogeneous gridded system. This efficiency may lead to routines for interactive boundary condition application that can use the feature listing output by CSC to rapidly return an effective flow property implied by the image. This is a more interesting prospect if applied to reservoir-scale images.

Both Fast Montoto Thinning, to simplify a rock pore image to its topological summary, and Cederberg-Sobel Coding, to extract pore lists as a network for proxy solutions to the gridded (two-d) heterogeneous diffusion problem implied by the image, are important tools in the explorationist's search for symbolic description of heterogeneity in gridded data. Because it requires only sequential access to image data, CSC is a better foundation for three-dimensional feature extraction than alternate approaches that require random access of image data. Many parallel-running sequential data accesses may be synchronized as intercommunicating processes—far more systematically than the same number of random data accesses. This is a vital consideration for future work in three-dimensional feature extraction from seismic data volumes.

2.5 Glossary

0-or-1 An adjective to describe image data that is Boolean, binary, or an indicator function, depending on one's favored vocabulary.

CSC An acronym for Cederberg-Sobel Coding, the second algorithm presented in this paper. It extracts features from neighbor-coded images to yield lists of their contents. These contents are vital data to pattern recognition processing of the images. CSC requires only sequential access to the image data, in raster scan fashion. It is extended from Cederberg's work to accept equally well either pore boundaries or their thinned arcs.

complete nodes CSC node pels that make a one-time modification of the output lists, such as joining two arcs and starting a third at a given pel. This is distinct from CSC's action at a pnode, where temporary nodes store information until arcs are completed at some later time.

compound nodes CSC's nodes that have characteristics of both minpoints (arc or arcs come together from above) and maxpoints (arc or arcs continue below). These may be found in images with thin arcs.

FMT An acronym for Fast Montoto Thinning, the first algorithm presented in this paper. It reduces 0-or-1 valued image areas to arcs along their backbones, skeletons, or medial axes, depending on one's favored vocabulary. A measure of the maximally inscribable square size is recorded along the remaining arcs, which can be used to reconstruct originally elongated features.

maxpoints Cederberg's term for pinodes.

minpoints Cederberg's term for unodes.

neighbor-code A computational shorthand for the contents of a 3×3 neighborhood surrounding a pel. In neighbor coding a 0-or-1 valued image of pores, each pel is replaced by the eight bits of its neighbors, with the east neighbor in the least significant bit, and proceeding counter-clockwise to leave the south-east neighbor in the most significant bit.

nodal A nodal is a topological entity of concern to CSC. It is a broader class than nodes, containing both them and pseudo-nodes.

node A topologically important pel where an arc begins or ends. Also those pels where three or four arcs join together.

pel A short word for pixel.

pinode A short word for πnode. A pnode at a curve maxima, from which two arcs temporarily extend, for bookkeeping purposes only. These are discarded by the time the final output is written.

pnode A short word for pseudo-node.

pseudo-node A false node that exists in sequential access feature extraction. For example, at the top of a pore, two arcs appear to join at a the uppermost pel. The same as curve maxima and minima, also referred to as pinodes and unodes.

raster scan The way that images are drawn on TV screens. A time series is mapped onto a two-dimensional field by filling rows of pels from left to right, in top to bottom order, until the image field is covered.

unode A pnode at curve minima, where two arcs join together in one of eight possible ways. Correctly recording the effects of unodes is the most expensive operation performed by CSC beyond initialization.

Chapter 3

Application of FMT to Rock Pores

3.1 Re-introduction

Given an image representation of a rock physical property distribution, many earth scientists face a dilemma. Many of them would agree that the connectedness of an image, like areas that represent fluid conduits, relates somewhat to bulk fluid flow properties. Most would agree that connectedness in conduit zones is related to those properties. Yet we do not seem to have an accepted numerical measure of this connectedness that could be applied to this volume, as well as volumes at arbitrary scales like this volume. Electrical resistivity formation factor is used for lack of a better measure at the rock core scale and because it has long been measured with logging tools. But given a microscopic sample of pore space, from sidewall cores, cuttings, or more substantial samples, what can best be done with an image of the pore system?

What I sought in my dissertation work was a measure of connectedness—which might be called connectivity—adapted to the heterogeneities that distinguish rocks from ideal media. Existing measures are the cause of the earth scientist's dilemma. On one hand, two point correlation methods (Journel and Huijbregts, 1978)[31] like a variogram are powerful tools for the characterization of anisotropy, but questions of connectivity are asked in reference to images where indirect flow (i.e. around

obstacles) makes two-point correlation methods less diagnostic than when flow is known to occur along linear paths. On the other hand are data-exhaustive processing measures[12, 13] that convert seismograms into image data and help seismic interpreters model petroleum fluid flow systems, through the application of some equations of physics to field observations. While image data may be enhanced or characterized, interpretation of images falls very soon to human observers. Perhaps with appropriate technology, the human intervention can be reduced in these interpretations.

Here, a middle approach is presented, midway between image enhancement and characterization, under the assertion that it is inherently worthwhile to have a connectivity measure that can reduce image heterogeneity of geological complexity to a standard numerical representation. I chose to define this measure in the context of *machine vision*. Machine vision involves acquiring a digital image, processing and analyzing its data, and automatically interpreting the elements of the analysis such that a decision can be made about objects in the image. This way of working reduces voluminous image data to a decision, often simply "yes" or "no." (Haralick, 1986)[27].

A connectivity metric, defined in a general enough sense, could solve a communication problem between geological disciplines, whenever they share a common need to reduce image data from different physical expressions to a common interpretation. Simply put, the goal of heterogeneity representation is to define rigorously a connectivity of objects within an image, for a given application. I suggest that my work is a useful answer to one of the difficulties faced by rock image analysts: "At the theoretical level, it will be a major task to select, among the many quantitative parameters furnished by image analysis, those that are really useful for the petrophysical interpretation of pore spaces." (Bourbié et al., 1987)[8, 7]

One difficulty with work in geological machine vision is the words that describe it. To me, "machine" means computer plus camera plus hardware and software

converters between data types. I believe the terms I use are carefully thought out, and I offer them as useful compromises between terms in common use by scientists and machine vision jargon. Whether or not the use of my terms persists within the earth science community, the concepts they represent will likely become more widely familiar as machine vision techniques are absorbed by earth science researchers. For now, I feel they just make things easier to discuss.

The following section describes the applications work I completed, and provides some details of data collection and processing procedures. It sets the context in which the processed results now available are being related to rock properties. That section, together with its figure captions, details the processing of both synthetic data and three Fontainebleau sandstone samples from the seven in Stanford's rock library.

3.2 Review of Applications Work Done

In the following application I digitized images from a video camera mounted on a petrographic microscope viewing blue dye impregnated 60 micron thin sections at 40X through a red dichroic (high quality) filter to boost the contrast of blue pores from clear grains. The Fontainebleau sandstone samples are clean quartz arenite of Rupelian (early Oligocene[8]) age that range from porosity 5%, permeability 5 millidarcies, to 20% porosity and 1100 millidarcy brine permeability—with no major changes in mineral composition.

As reported in the American Geophysical Union Fall 1987 meeting, an image object thinning transformation (Quinn, 1987.)[41] Fast Montoto thinning was derived from rock fracture image processing work. (Bel-Lan and Montoto, 1981)[2] More recent effort has been directed at a new feature extraction algorithm, designed to be fast and reliable in producing network description files from Montoto thinned image data. The result of this effort is Cederberg-Sobel Coding (CSC), and I hope it will be recognized as an important step in the use of pore image data to estimate bulk

permeability.

The fast Montoto thinning (FMT) and microloop simplification (MLS) codes were mostly written in the C language and as microcode for the Recognition Technology raster engine that I added to an IBM PC-AT to build Stanford's rock image analysis system. In contrast, the network extraction code, CSC, is written in the C language for possible use on many general purpose machines running MS-DOS or Unix, particularly the Stanford Rockphysics laboratory's image analysis system IBM PC-AT, its associated IBM RT-PC/125 computer, and the laboratory's Ardent Titan II mini-supercomputer. CSC's approach combines and extends the raster-scan conversion algorithm of Roger L.T. Cederberg (Cederberg, 1979)[11] with the pel (pixel) neighbor coding scheme of Irwin Sobel (Sobel, 1978)[48] to create a hybrid image processing code to extract either thinned network or polygon boundary lists from raster data. Sobel coding may make use of special hardware to speed processing in its first few steps.

In its input data access, CSC uses a single-pass approach rather than random access, for two reasons. Firstly, Montoto thinned rock pore space images often contain a weblike network of pel arcs. Random access feature extraction is likely to visit each arc pel at least twice when identifying all closed loops. Secondly, non raster scanning feature extraction requires that most or all of the image be retained in memory and accessed randomly while it runs. CSC accesses the preprocessed metaline image and thinned network image pel pairs only once, and sequentially.

The resolution of images to be analyzed will inevitably expand in future research. CSC's single and sequential approach to input data access can be adapted to larger image data sets or even 3-dimensional data more readily than random access boundary following algorithms, through its reduced image memory requirements. Pratt[40, pp 699–705] and Rosenfeld and Kak [45, pp 112-130] provide discussions of random access (omnidirectional) feature extraction.

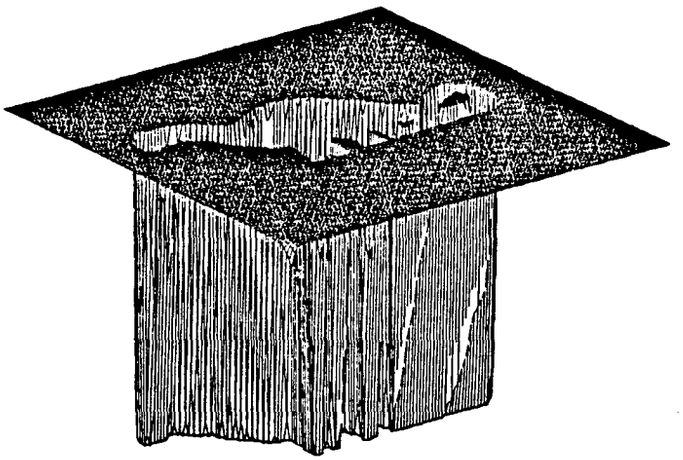


Figure 3.1: Perspective plot of video data digitized from the APAT0 paper cut-out figure. The bright background has saturated the camera to produce the level surface, and the unevenly illuminated figure has lower intensity values. The 512×480×8-bit pel (pixel) image was decimated to a 120×120 grid for plotting. Data is in the range [0–255].

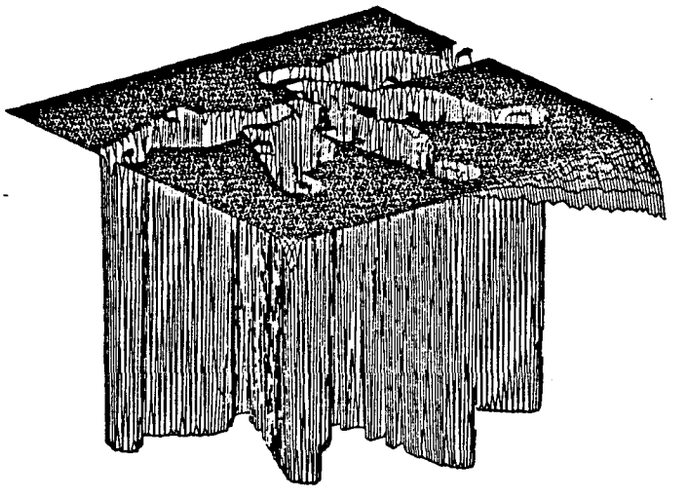


Figure 3.2: Perspective plot of video intensity data for the APAT05 image. Uneven lighting in the lower corners, especially the lower right, appear as unsaturated areas of the image. A threshold segmentation was applied below that level to yield the binary input to FMT. This is a plot of decimated data, on a 120×120 grid, in the range [0–255].

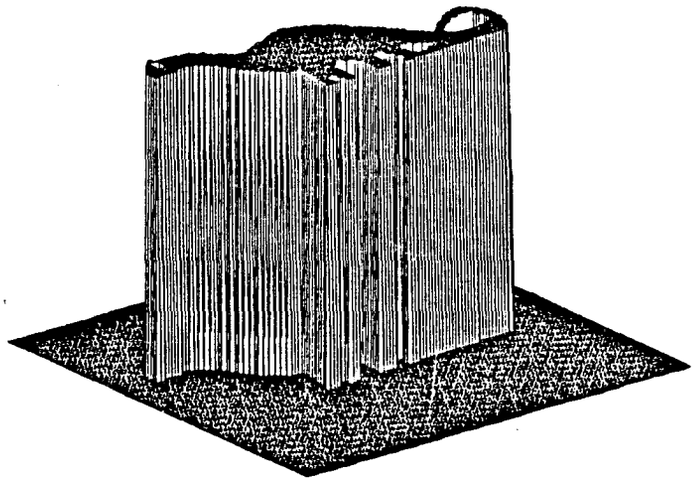


Figure 3.3: Perspective plot of data prepared for input to FMT. Image APAT0 has been segmented to binary values of feature (1) or background (0). The 512×480 video digitization of Figure 3.1 was segmented and 4×4 pel blocks were averaged to yield a 120×120 grid for plotting

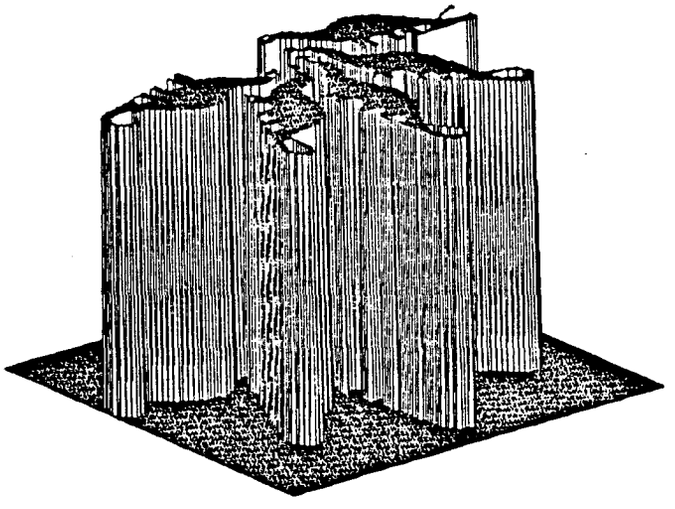


Figure 3.4: Perspective plot of APAT05 image binary, decimated to 120×120 pels for plotting. The image data values range [0,1].

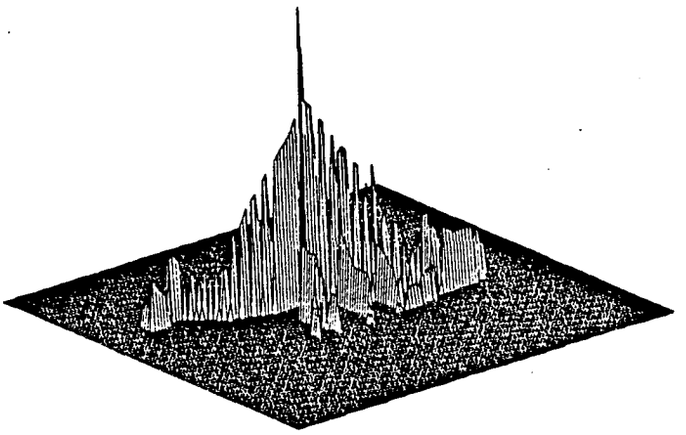


Figure 3.5: Perspective plot of APATO image, after FMT. The transform was applied to the 512×480 binary shown in Figure 3.3, yielding nonzero pels only along medial axis arcs, recording the size of the maximally inscribed square at that point. Arcs along the wider parts of the feature have higher width values after transformation. To produce this plot, 4×4 pel blocks were averaged to make the 120×120 grid for plotting. Values range over [1–180].

3.3 Discussion

The following section describes the application of FMT to Fontainebleau sandstone images. The images are derived from video data shown in Figures 3.1 and 3.2 and consist of segmented "silhouettes", as shown in Figures 3.3 and 3.4, derived from the video data. After thinning, the binary image is transformed as shown in Figures 3.5, 3.7, 3.6, and 3.8. The thinned images can be summarized as as lists of nodes, arcs, node-node connections, and closed loops. If thinning is not applied to the binary image before running, CSC extracts the boundaries and the process reduces to the raster scan conversion of Cederberg.[11]

Histograms of pel values in the FMT thinned results are plotted in Figures 3.9 and 3.10. There is a similar shape to the two scaled histograms, with two moderately large modes below iteration 80 for image APATO and below iteration 45 for image APATO5, with a notch somewhat visible at iteration 43 in APATO and 27 in APATO5. I believe this is noteworthy, since the APATO polygon is both replicated at a different

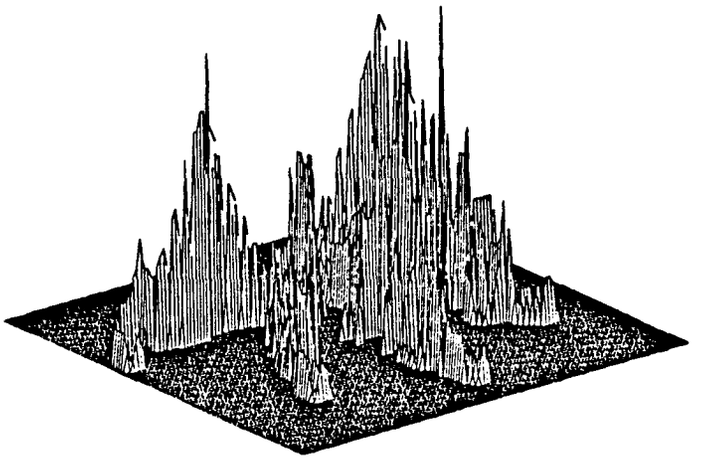


Figure 3.6: Perspective plot of APAT05 image after FMT. Data is in range [1–80].

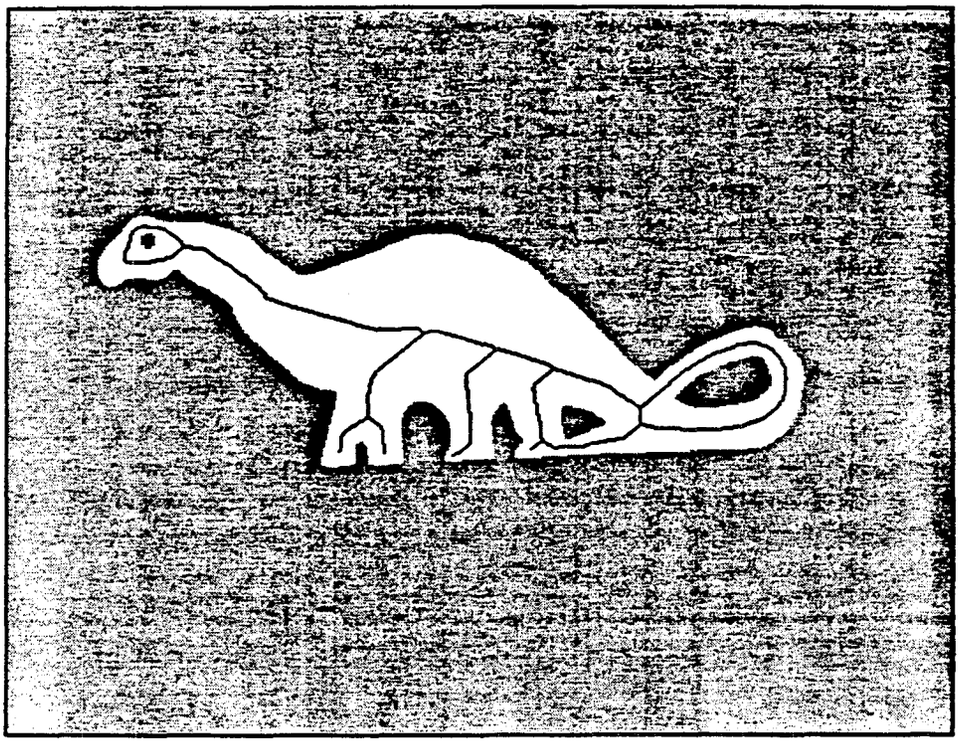


Figure 3.7: Composite plot in map view of both binary (white) and thinned arcs (lines within white) APAT0 image used to make Figures 3.3 and 3.5. The actual 512×480 data was used in this plot. Halftoning is used to represent variations in the width values recorded along the arcs, so that smaller values, as in the head and tail, are darker and larger values, as in the middle of the body, are somewhat lighter.

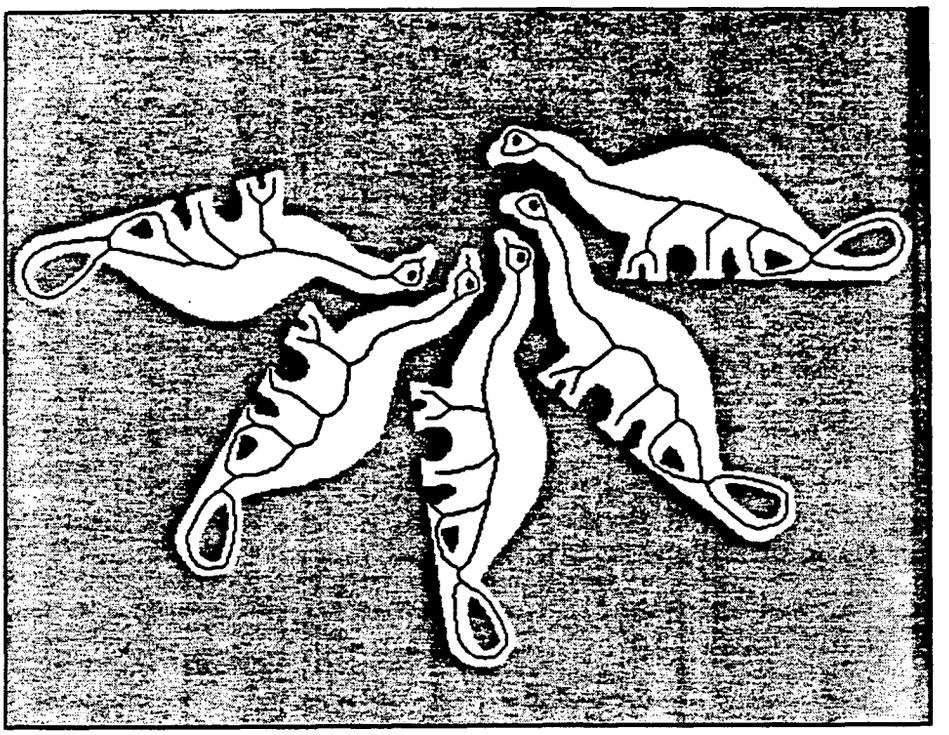


Figure 3.8: Composite plot of the APATO5 image. The APATO paper cut-out figure was reproduced mechanically and digitized with the rightmost figure unrotated, and four copies rotated clockwise at 38° , 70° , 120° , and 164° . The FMT thinning is based on inscribed squares that are aligned with gridding, and the transform is sensitive to boundary roughness. Thus, the medial axis positions within each figure change with rotation. All five figures contain three loops, one forked and one simple peninsula. While the 0° and 164° figures have adjacent loops in their tails, the other three figures have a small single strand extending between them. Only the 70° and 120° figures have a peninsula off the middle loop from the foot furthest back, like Figure 3.7. Because of scaling, the unrotated figure is smoother than its counterpart in that figure and causes the loss of that detail. Also, while the leg peninsulas extend mostly perpendicular within the body, the 0° figure's legs approach the body's axis toward the tail at 45° , while the 164° figure's legs do so towards the head.

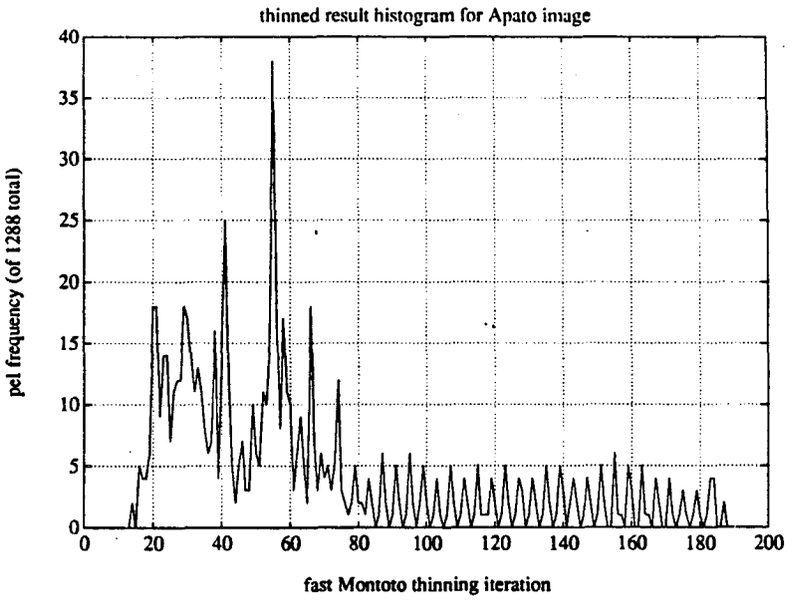


Figure 3.9: Histogram of values in APATO thinned result. A total of 1288 nonzero pels remained, a 95.6% reduction in area from the original binary image. The histogram plots a distribution of maximally inscribable square along all medial axis arcs. A side of the inscribed square is about one half the iteration value shown on the x axis of the plot. The maximum width represented is 94 pel widths.

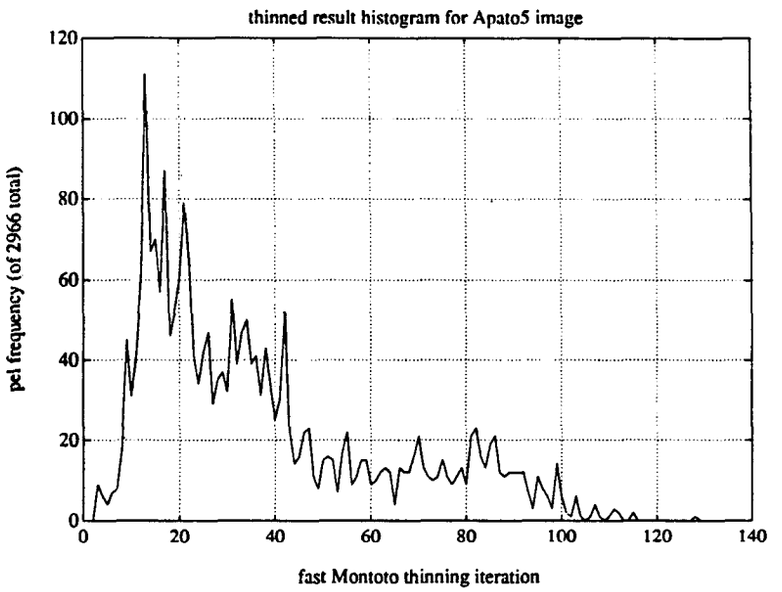


Figure 3.10: Histogram of values in APATO5 thinned result. A total of 2966 nonzero pels remain, a 94.2% reduction from the binary image of Figure 3.4. The maximum width represented is a square 65 pels on one side.

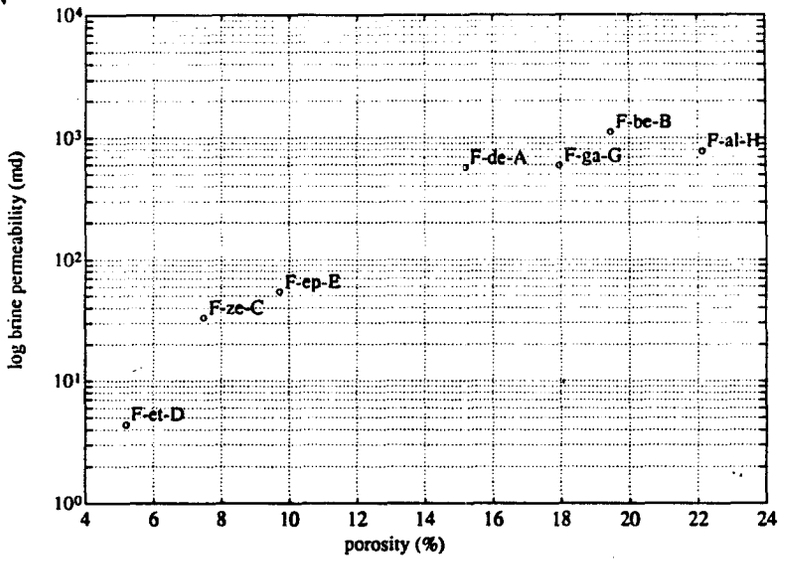


Figure 3.11: Porosity vs. log brine permeability plot for the Fontainebleau samples. The labels refer to the first two letters of "alpha" through "eta", and the capital letters at the end are the notation used by earlier workers including P. Doyen[14], who worked on the same thin sections.

size and rotated in APATOS, yet a simple histogram measure of size distributions reveals the similarity of features between the images. The histograms of FMT thinned rock images look quite different, as shown later.

Moving from synthetic to rock data, I applied FMT to views of three of the seven Fontainebleau samples in the Stanford Rockphysics Laboratory's rock library collection. Sorted in terms of porosity, these Fontainebleau samples are called alpha through eta and shown in Figure 3.11 The three I chose for this work are alpha, epsilon, and eta, re-plotted with a trend line for all seven samples in Figure 3.12. I chose the extreme members to illustrate as much variation in sample permeability as possible, and chose only one intermediate sample because at the time of writing, I did not have a feature extraction code available, which meant that processed results would remain as images—I limited the study to three samples to avoid repetition in this paper's figures.

The three samples are shown in gray level image plots taken from a video digitization at 40X , as shown in Figures 3.13, 3.14, and 3.15. The field of view is nearly the same as shown in the brightness contour plots of each input image, Figures 3.16,

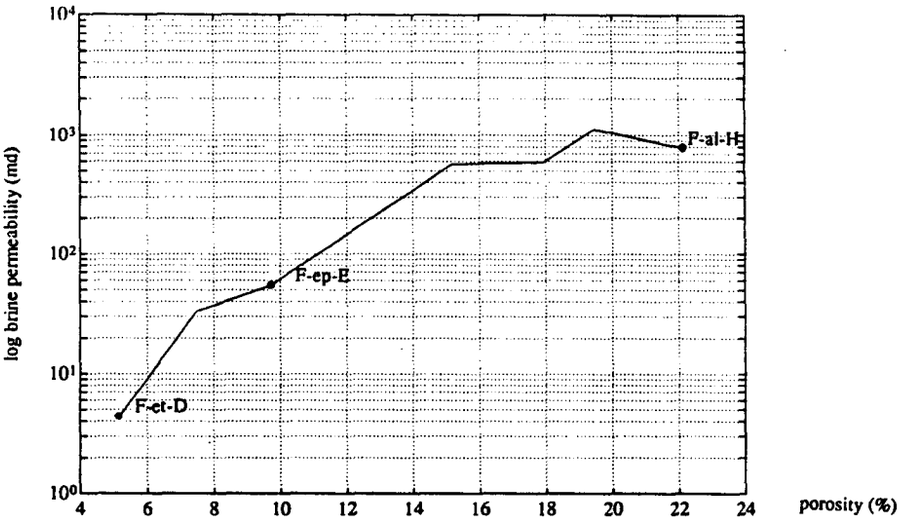


Figure 3.12: The same Permeability/Porosity plot as in Figure 3.11, but showing only those samples used in this study, as well as a trend line through all seven. Because of the trend line, it is apparent that samples zeta, delta, and beta are above a mean trend for all samples, while eta, epsilon, gamma, and alpha are below the trend in this cross plot.

3.17, and 3.18, and is horizontally about 3 mm across. The scales of the contour plots are shown in pels but are calculated for 2×2 decimated data. More precisely, the gray level plots (Figs 3a, 4a, 5a) are 512×480 pel images of 8 bit depth, where pels are 6 microns wide and rectangular with horizontal:vertical aspect ratio of 5:4.

With each of the gray level plots, I present a Montoto thinned representation of pores, when they are defined by gray level values below 240, with brightest white set at 255. Without doubt, the results of FMT depend on the gray-level threshold chosen to define the 0-or-1 pore image. To minimize this sensitivity, I adjusted the optics to produce higher-contrast images. The black-and-white video camera viewed the blue-dyed pores through a red filter, with the microscope illumination near its maximum. This made the edges of the pores more sharply defined at the expense of viewable detail in the rock matrix. Figure 3.21 shows the Fontainebleau F-eta sample, with porosity 5.18%, and brine (bulk) permeability 4.42 millidarcy. This sample is the leftmost point in the plot of Figure 3.12. Figure 3.20 shows the F-epsilon sample, with porosity 9.71%, and permeability of 54.5 millidarcy.

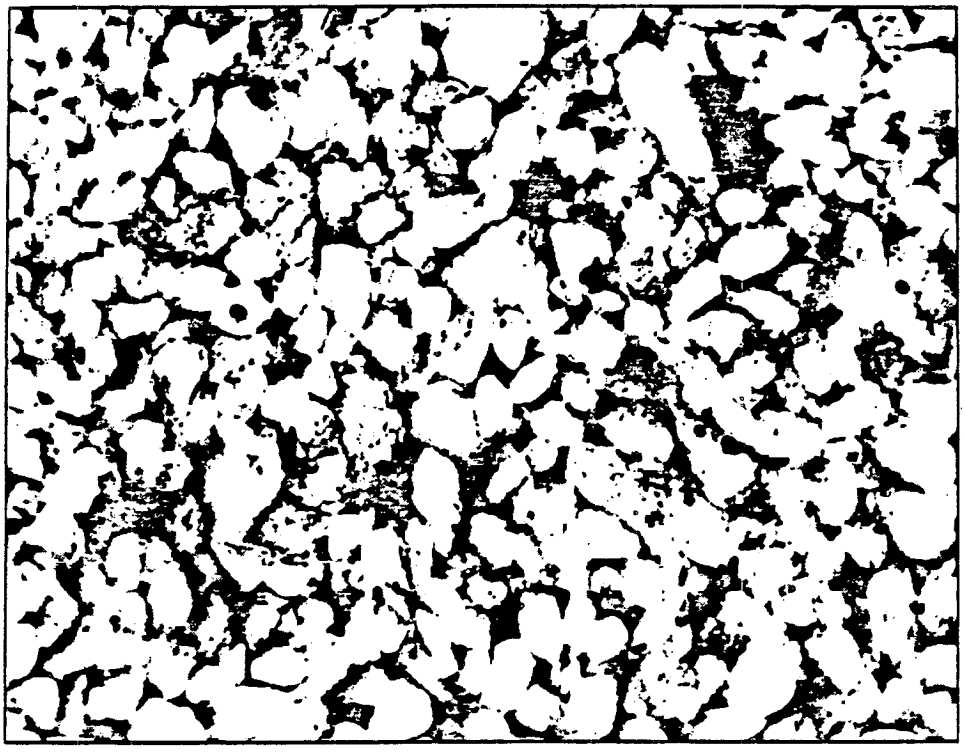


Figure 3.13: Digital Photomicrograph of Fontainebleau sandstone, alpha sample, brine permeability 784 millidarcy, porosity, 22.13%. Digitized at 40X from a video camera signal to yield 512×480×8-bit pels (256 gray levels). Field of view is 3mm across, and pels have H:V aspect ratio of 5:4. Pores are filled with blue dyed epoxy, and a red filter was used to increase contrast for the black and white camera. The thin section is 60 microns thick.

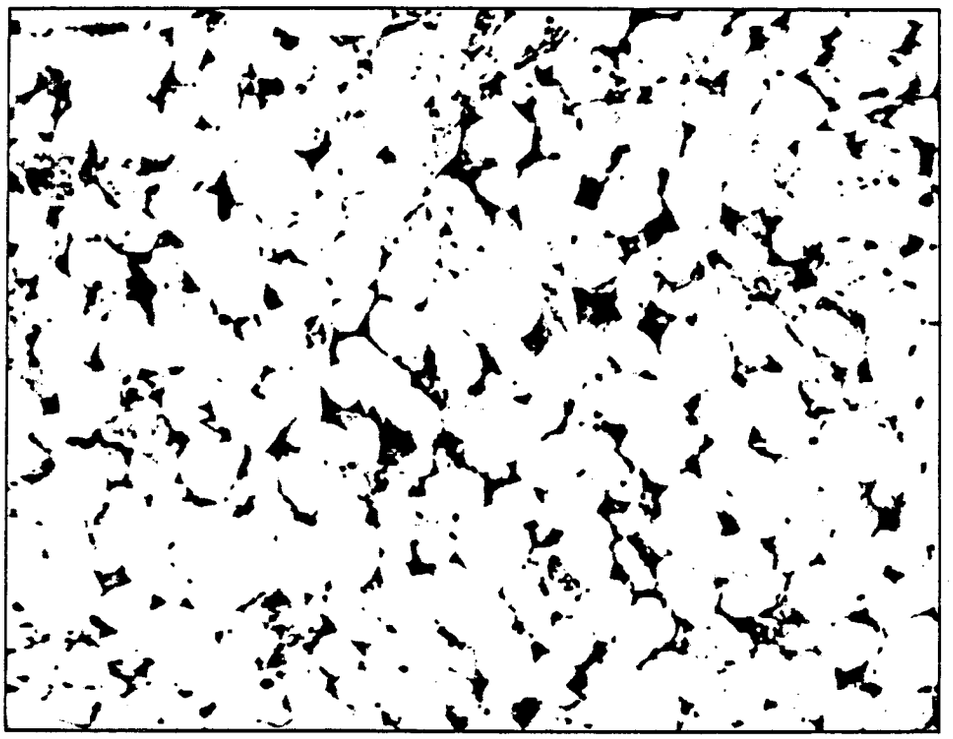


Figure 3.14: Digital photomicrograph of Fontainebleau epsilon sample, brine permeability 54.5 millidarcy, porosity 9.71%. This data 512×480×8-bit pels. 3mm wide field of view at 40X magnification.

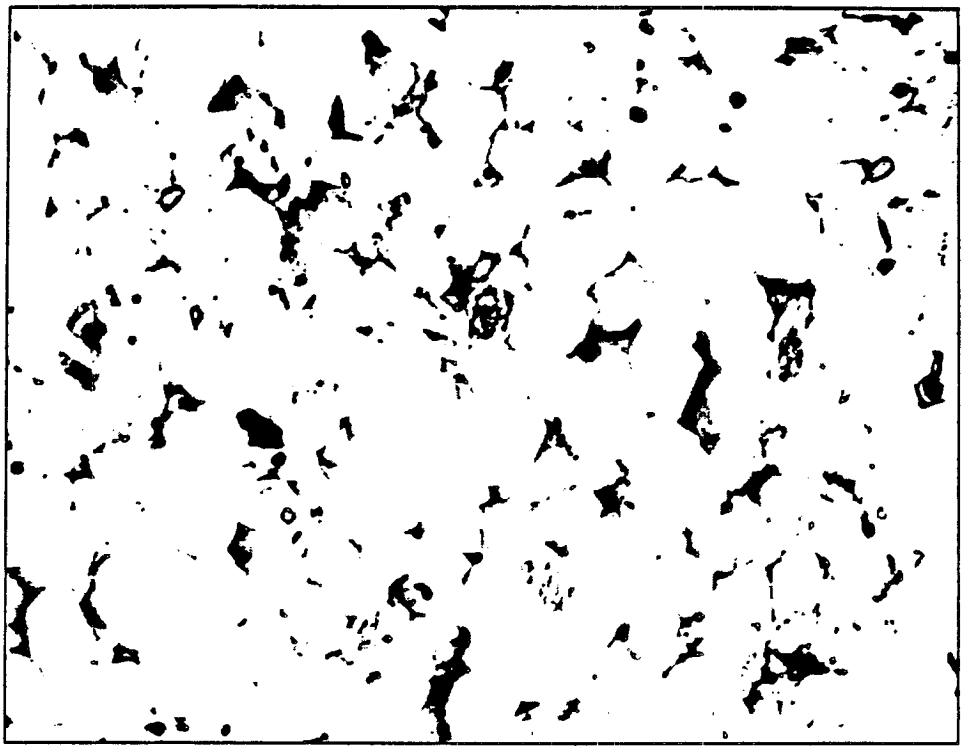


Figure 3.15: Digital photomicrograph of Fontainebleau eta sample, brine permeability 4.42 millidarcy, porosity 5.18%. This image data 512×480×8-bit pels. Field of view, 3mm across at 40X.

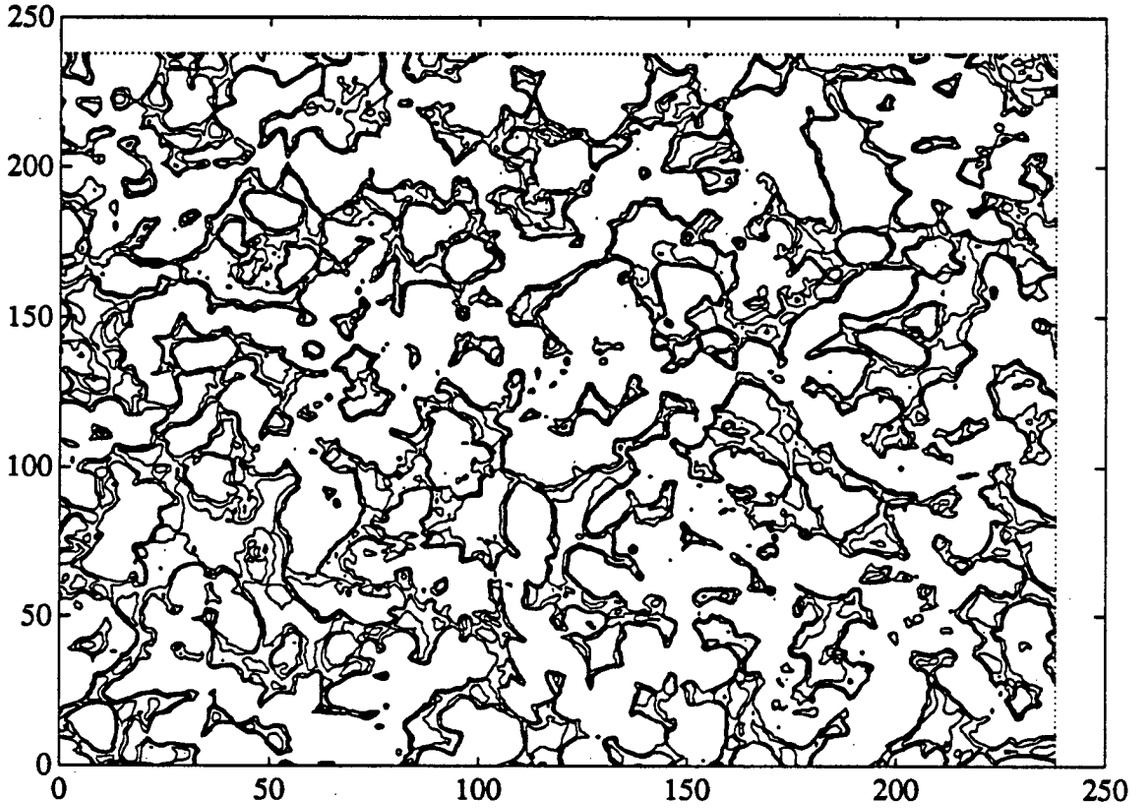


Figure 3.16: Contour plot of Figure 3.13, Fontainebleau-alpha image decimated with 2×2 averaging to yield a 240×240 left-registered view. As contours of gray levels, this plot shows a family of possible segmentations defined by gray level thresholds. The white areas in Figure 3.13 are 255-valued, and the contours here plot into the darker areas of the image as through they were canyons incising a karst plain. Contours are drawn for the 50, 100, 150 and 200 gray values. Axis dimensions in pels, after decimation.

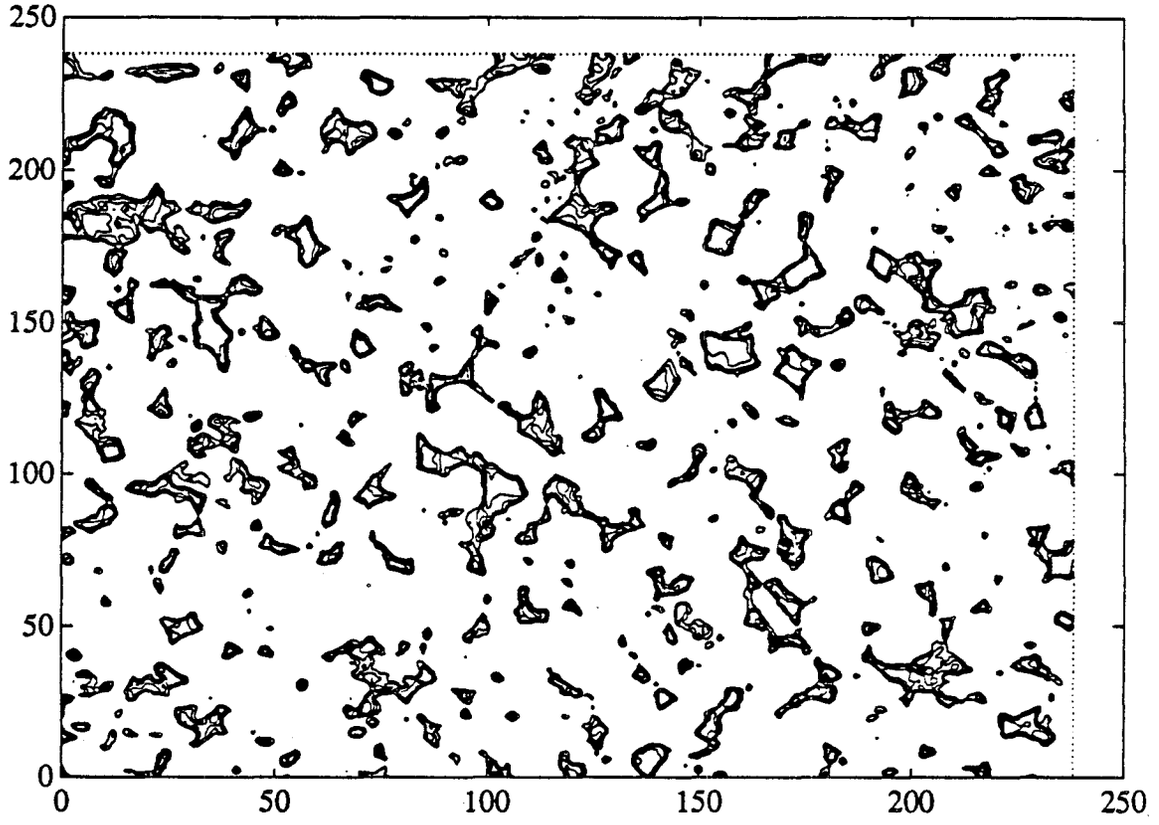


Figure 3.17: Contour plot of Figure 3.14, Fontainebleau-epsilon image, calculated from 2×2 decimated data. Contours are drawn for the 50, 100, 150 and 200 gray values. Axis dimensions in pels, after decimation.

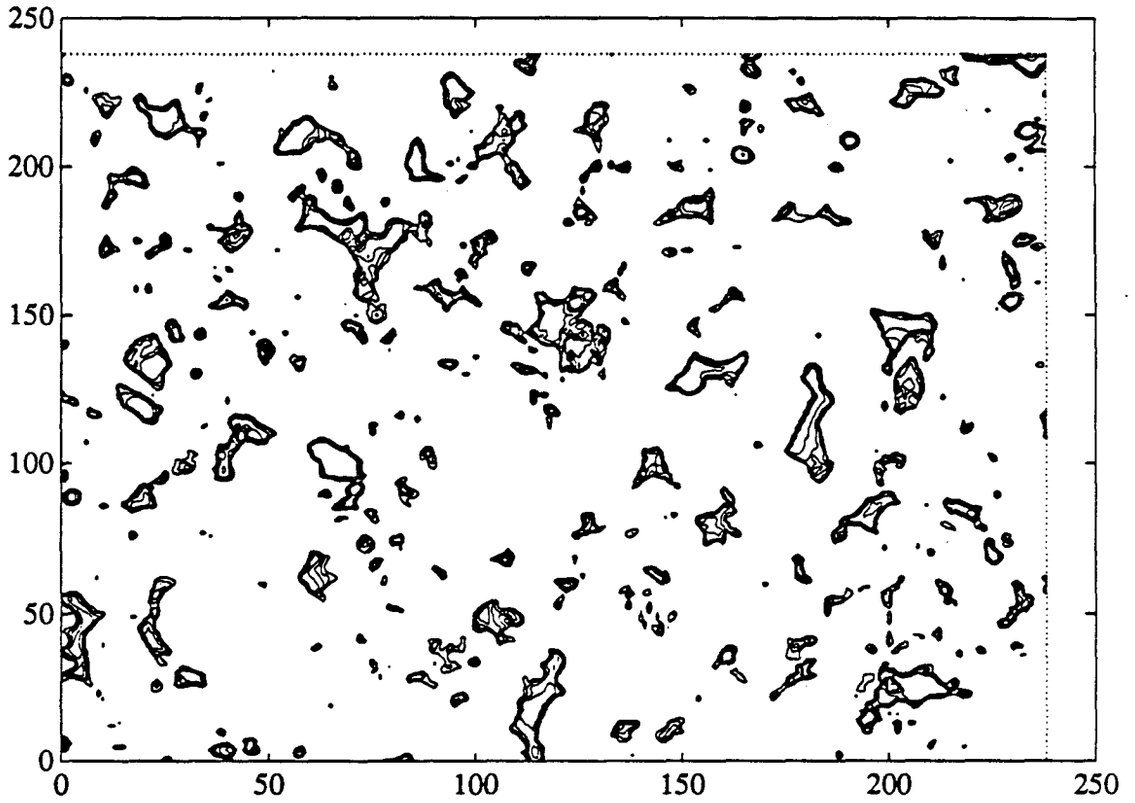


Figure 3.18: Contour plot of Figure 3.15, Fontainebleau-eta image, decimated by averaging to 240×240×8-bit pels. Contours are drawn for the 50, 100, 150 and 200 gray values. Axis dimensions in pels, after decimation.

Finally, the exceptional F-alpha sample, with porosity 22.13% and permeability 784 millidarcy is shown in Figure 3.19, the rightmost point in Figure 3.12.

Qualitatively, the connectedness of the thinned networks in Figures 3.19, 3.20, and 3.21 increases with increasing porosity and permeability. This increase is expressed heuristically in the network that CSC extracts by both increased mean length of connected arcs and increased width values from FMT recorded along them. In Figure 3.19, arcs allow percolation from the left to the right borders of the image. In Figure 3.20, 22 arcs are over 75 pels long. In Fig 3.21, only 8 arcs exceed this length.

Beyond the path lengths, the widths recorded in these thinned networks decrease with decreasing porosity, as shown in the distributions of Figures 3.22, 3.23, and 3.24. The maximum width values found in these networks decreases as well. Greater medial axis values correspond to wider polygons [1] in the original gray level images. The corresponding maximum values for all pels in the thinned networks are 100 microns in Figures 3.21 and 3.24, 150 microns in Figures 3.20 and 3.23, and 250 microns in Figures 3.19 and 3.22.

CSC's output files support a wide variety of statistical measures, well-used in petrographic image analysis of pore boundaries. The work of Ehrlich and his students [18] has listed over 300 measures derivable from pore boundaries alone. When used on FMT-processed images, CSC can also yield circuit representation of the pores after thinning as a linear system. Since these samples are known to vary in brine permeability, and can be seen to improve in connectivity through visual inspection, circuit descriptions of these arcs are certain to reflect this improved connectivity for all reasonable boundary conditions (those that allow percolation and thus permit some measure to be made). I argue that the circuit modeling is a more direct and representative model of the pores as fluid conduits than are the numerous statistical measures of pore boundary characteristics.

[1] Technically, larger sized maximally inscribably squares withing the original binary blobs (polygons)

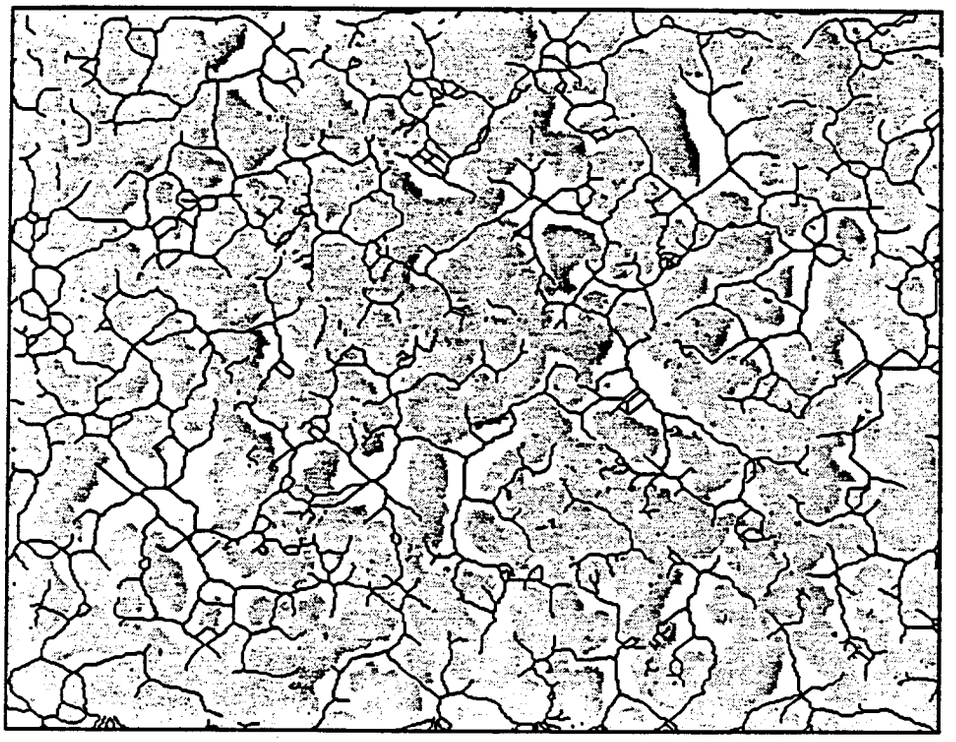


Figure 3.19: Composite plot of a threshold segmentation of Figure 3.13, the Fontainebleau-alpha image. Pels ≤ 240 are shown in white. The varying width values along arcs in the FMT result make them suitable for numerical representation of this image as a network.

Figure 3.20: Composite plot of the (pels $\leq$ 240) segmentation (white) of the Fontainebleau-epsilon image, Figure 3.14, and its FMT computed result (arcs therein). If compared to Figure 3.19, the connected arcs are much shorter and their width values are less.

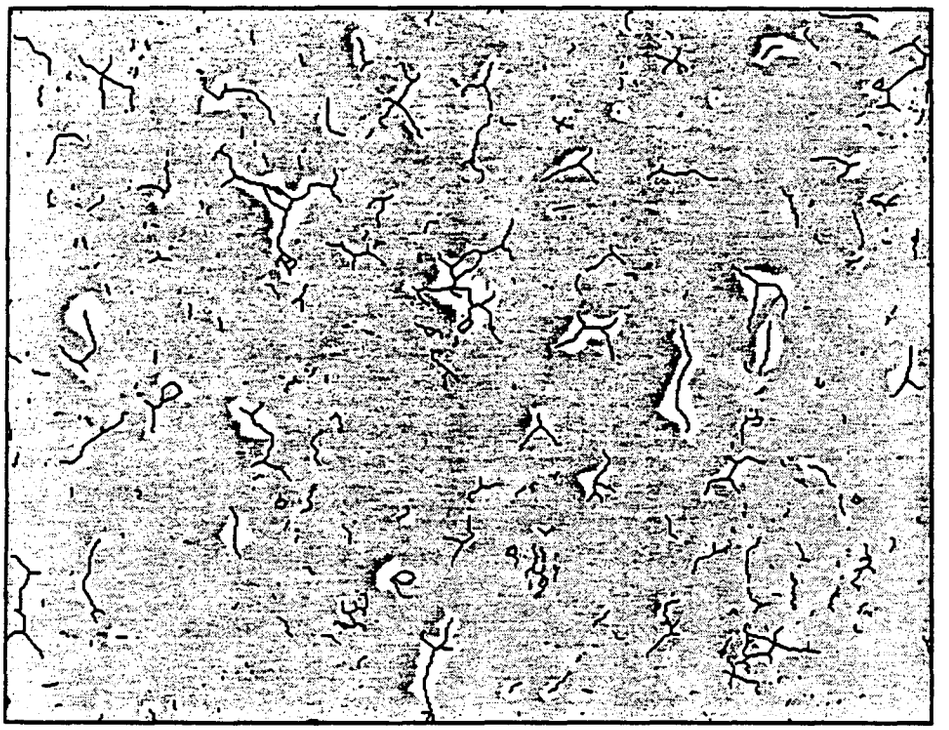


Figure 3.21: Composite plot of the (pels $\leq$ 240) segmentation (white) of the Fontainebleau-eta image, Figure 3.14, and its FMT computed result (arcs therein). This figure is even less connected than Figures 3.19 and 3.20.

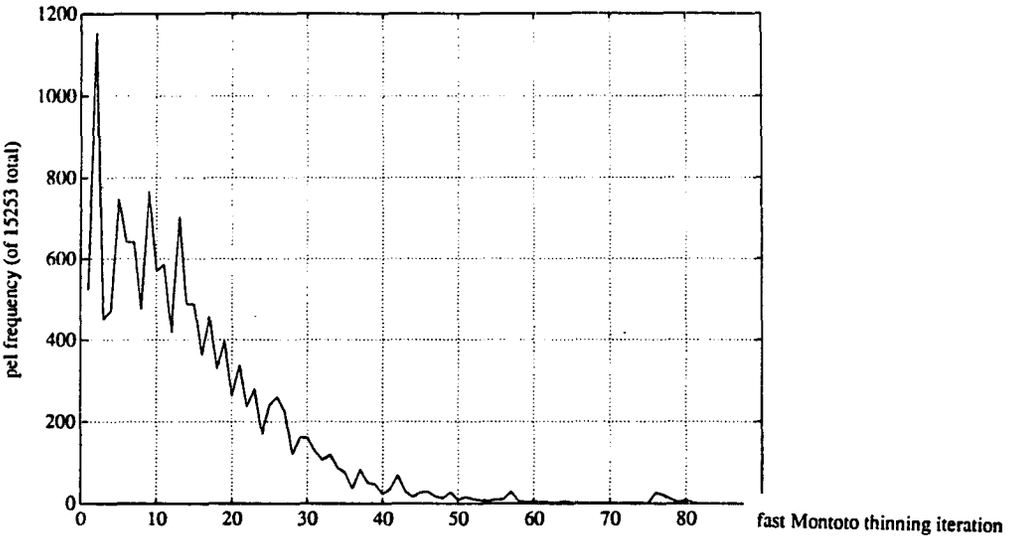


Figure 3.22: A histogram of nonzero values in the FMT thinned result of Figure 3.19. The left spike around iteration 2 is representative of microporosity. The width distribution is oscillatory but fairly flat from there until iteration 13, where it starts dropping as fewer wider pore throats are found, yielding fewer thinned arcs. The oscillations are due to the directional sequence used in the FMT algorithm: north, east, west, south. Apparently, greater numbers of pels were added to medial lines during east- and north-facing erosions, and fewer from south-facing ones. Only the largest pore chambers remain after iteration 40, though the contribution to optical porosity of these values around iteration 76 is substantial.

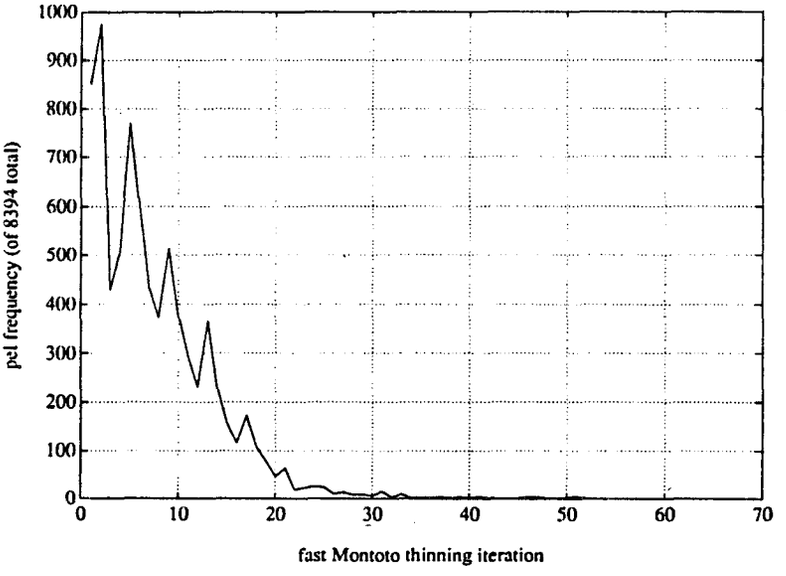


Figure 3.23: Histogram of nonzero values in the thinned result of Figure 3.20. The leftmost spike represents microporosity. Long thin throats with widths less than 10 pels (iteration 20), skew the distribution toward lower values.

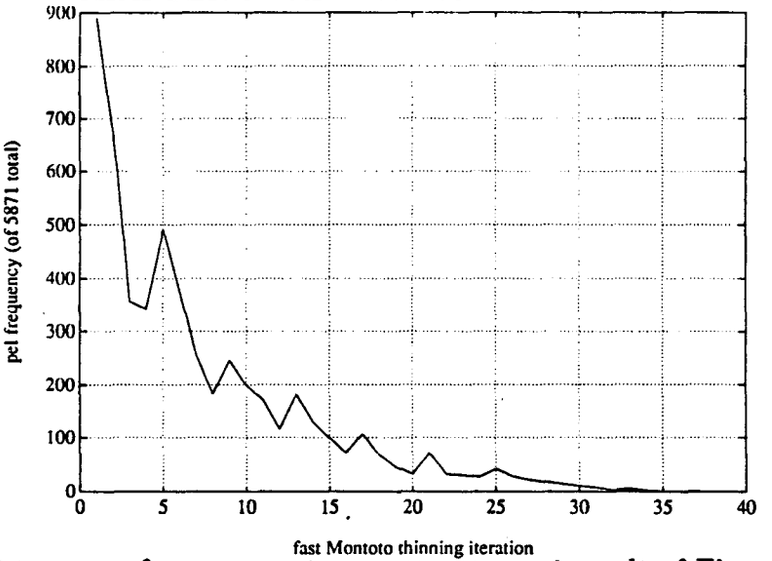


Figure 3.24: Histogram of nonzero values in the thinned result of Figure 3.21. The microporosity fraction, while only some 2000 pels, dominates the first three iterations. Small pores show up at iteration 5. In comparison with Figures 3.22 and 3.23, this plot is less skewed to low values, emphasizing the smoothness and isolation of these pores. It is important to note that FMT's algorithm produces the roughness with period equal to four iterations, not the pores themselves! The roughness is due to anisotropy in the image, but I have not modeled the exact cause.

The remaining active area plots for each figure show the area remaining in the feature after each iteration of thinning[2]. Figures 3.25, 3.26, and 3.27 show the cumulative ablation of active objects during thinning. Starting at iteration 1, the optical porosities of 41.40%, 15.63%, and 10.34% [3] were reduced to zero iteratively by FMT's application[4]. Though the plots in Figures 3.25, 3.26, and 3.27 are scaled, they have distinctive shapes. The eta sample in Figure 3.27 evidently has smoother, more isolated pores because, for its scale, the plot approaches zero much less readily than the alpha or epsilon samples in Figures 3.25 and 3.26.

The remaining active area plots for each figure show the area remaining in the feature after each iteration of thinning. [5] Figures 3.25, 3.26, and 3.27 show the cumulative ablation of active objects during thinning. Starting at iteration 1, the

[2]This is an ablation or directional erosion in a N, E, W, S sequence

[3]The systematic error in optical porosity can be greatly reduced through the use of a more sophisticated classification from gray level to binary image

[4]Even if a pel is saved as a medial axis arc, it is no longer part of the active area measured here.

[5]This is an ablation or directional erosion in a N, E, W, S sequence

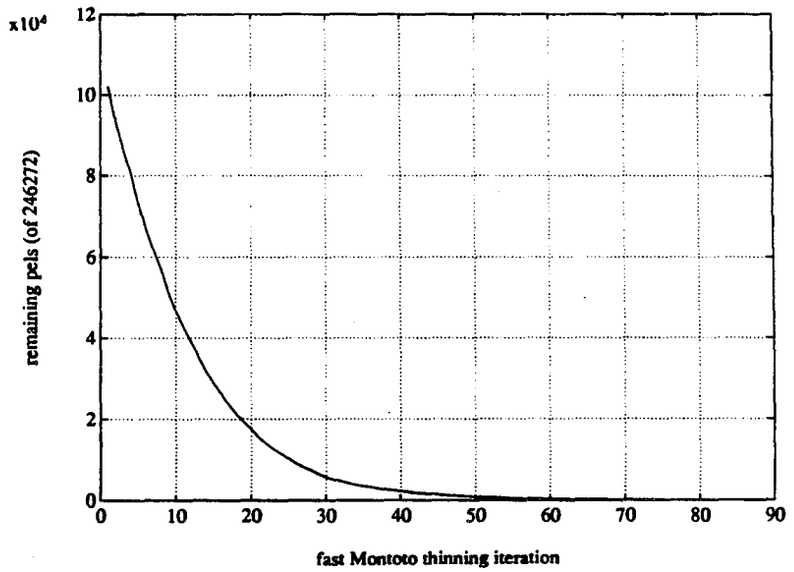


Figure 3.25: Active area history plot for remaining area at each iteration, recorded during the thinning computation that produced Figure 3.19. A rougher boundary will ablate more rapidly and make this curve approach zero more rapidly. The plot starts at iteration 1, and that value is equal to the optical porosity, 41.40%. Only at iteration 84 does the ablation finish. Width in pels is equal to half the iteration number, and at 40X this corresponds to a maximum pore width of 250 microns.

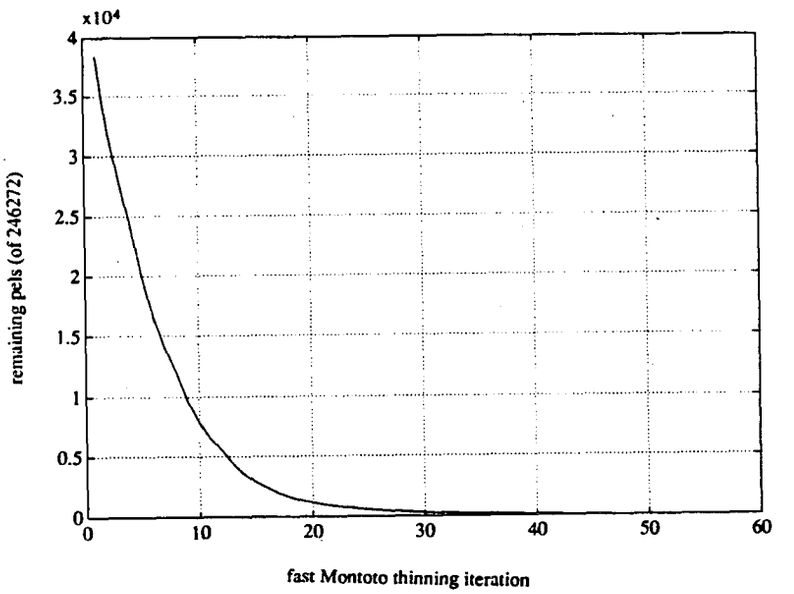


Figure 3.26: Active area history plot for remaining area at each FMT iteration, recorded during the computation that produced Figure 3.20. The narrowness of the pores increases their boundary area, increasing the rate of ablation. Optical porosity starts at 15.63%. The final iteration if 50, for a maximum pore width of 150 microns.

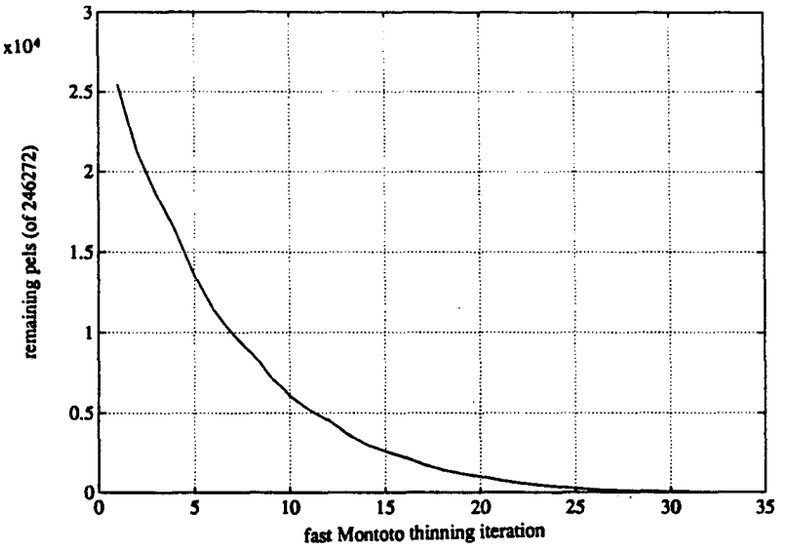


Figure 3.27: Active area history plot for FMT operations that yielded Figure 3.21. This plot is less skewed to lower values than were Figure 3.25 or Figure 3.26, emphasizing pore smoothness and isolation. Optical porosity starts at 10.34%, and FMT iterations continue to 34, for a maximum pore size of 100 microns.

optical porosities of 41.40%, 15.63%, and 10.34%[6] were reduced to zero iteratively by FMT's application[7]. Though the plots in Figures 3.25, 3.26, and 3.27

The steep early part of those active area plots is due to narrow interconnecting features as well as a rough boundary. While not completely independent from the data shown in Figures 3.22, 3.23, and 3.24, these plots are not simply an integration of thinned width values, because they do not measure area saved as thinned arcs. Depending on sample topology and roughness, large features in the image can reduce to very simple thinned results. Active area plots are guaranteed to be nonincreasing. Histograms of thinned results may be multimodal, although they are often nonincreasing if smoothed sufficiently.

The directional cycle used in the FMT algorithm removes outermost pels from pores along the sides facing north, east, west, then south, cycling every four iterations. All the histograms of thinned images I have seen are rough with a period of about four iterations. This is expressing, in what I believe is a complicated way,

[6]The systematic error in optical porosity can be greatly reduced through the use of a more sophisticated classification from gray level to binary image

[7]Even if a pel is saved as a medial axis arc, it is no longer part of the active area measured here.

some feature of the image's anisotropy. In other words, if the histogram has large peaks on the first, fifth, ninth, etc., iterations, then more pels join the medial line set during north ablation than other directions. While it seems logical that similar numbers should join during south ablation, this is not always so. If FMT becomes a more widely-used approach to rock image analysis, this question should be studied: what aspects of image anisotropy are affecting this roughness in FMT result histograms?

The disparity (by a factor of two) between bulk and optical porosities is due to the image segmentation selected for this work. Bulk porosity was measured in a helium porosimeter. Optical porosity was simply the fraction of image area identified as pore space after I segmented the video signal into a 0-or-1 image. My approach, not the helium porosimeter, produced this error—but I segmented these images to minimize the effects of my threshold selection on the resulting FMT network. In doing so, I gave up trying to get accurate optical porosity at the same time. A gray level threshold of 240 detected all visible pore features, even where only a small fraction of the thin section's thickness was blue epoxy. To match bulk porosity and at the same time model all detectable pore features, successive relaxation [45, pp 152–184] might be used, or FMT should be extended to become a gray level thinning technique. This is computationally more expensive, but the recent work in mathematical morphology [28, 44, 15] suggests that this is on its way.

My work has not made an attempt to relate FMT-transformed images to rock permeability. This remains as a challenge for others, who I hope will be aided in their work by Fast Montoto Thinning. Thinned images are still image data, difficult to relate to permeability without a feature extraction code, such as Cederberg-Sobel Coding (CSC), discussed in Chapter 2.

Chapter 4

Reasons To Apply FMT

In Chapter 4, I will start by introducing a list of the applications I believe would benefit from the use of the image tools that I am presenting. In Chapter 5 I will describe some of the terminology of mathematical morphology that describes the thinning I am presenting. Perhaps because the field of mathematical morphology is expanding and still developing, there is not as well developed a tradition of mathematical description and symbolism as there is for classical mathematics. And so I present a menu of selections which have been used in the literature of morphological image processing that describe this work—as it is distinct from the linear approaches in two dimensional signal processing, like convolutions, Fourier transform, and other linear filters.

After describing how these mathematical concepts are discussed in the literature, in Chapter 6 I turn to the *numerical* considerations which surround the implementation of these morphological concepts in imaging systems. I suggest that the classical operators one finds in programming languages such as FORTRAN are better replaced with other concepts and coding paradigms which better describe the different types of computing machines used in rapid nonlinear processing of image data at this time. These machines have names like parallel computing devices, raster engines, cellular array machines, and the like. Many are specifically built to extract numerical descriptions of images from the raw data of the image itself, as my work is directed. Such machines may be called vision engines. Other machines

used to take graphical object descriptions and render them onto a high resolution screen are called graphics engines and their task is complementary to vision engines, in a sense working the other way around.

Leaving the actual descriptions of the algorithm and the explicit numerical procedure involved in defining fast Montoto thinning for appendices, in Chapter 7 I discuss some of the consequences and steps involved in verifying the results of what this transformation does to digital images and how one can understand what the transformed results mean, hopefully in an application-independent way.

One can expect from thinning estimates of image object width. These have been called "pore throat size distributions." [18] But throat size distributions have been derived from pore boundaries alone. A more important result is that of tracing arcs of these distributions through the image—attacking the problem of spatial connectivity in two dimensions. Thinning provides an estimate of the object area remaining at various levels in a sequence of morphological erosion. This is used to evaluate disconnections in pore elements during cycles of erosion and dilation. From the results of a thinning operation on an image, one can hope to gain information that is needed for estimating the types of connections between pores that exist in the bulk rock outside of the observed plane of the thin section. This problem is arguably intractable, and I have not worked on it. Rather, the tools I have built provide symbolic descriptions of pores. This symbolic approach to image data may help others to tackle the concerns of estimating pores beyond the image plane.

I also discuss the types of analyses one could expect to make based on simple histogram summings of thinned results as well as another class of analyses that are based on the vectorization of curves—curves which represent either the boundaries of objects in the images or the lines that remain after thinning. Curve vectorization complements the thinning transformation and is a means by which many numerical evaluations of the image may be derived for a wide variety of rock property applications. I present a technique for this feature extraction that takes thinning from

the realm of a morphological curiosity to a practical tool.

I discuss conclusions in relation to those that have been stated earlier paper describing a version of this transformation by Bel-Lan and Montoto as well as putting their results into a context that may be more familiar to earth scientists already involved in image analysis, particularly those who are already making use of texture analysis when studying pores. I mention some of the areas that, at this time, appear that they would benefit from an application of this transformation, particularly when combined with its companion curve vectorizing algorithm.

4.1 Quantifying Image Connectivity

The original motivation for my work in image analysis of rocks was the estimation of permeability from thin section images. In reviewing the literature I felt that there could be an approach which was concerned with generalizing analysis methods rather than making them more rock formation-specific. This property that I am characterizing in the binary image is best called connectivity, or perhaps flow connectivity. It converts a raster expression of heterogeneity to a circuit model, at once saving aperture, position, and topology as related lists describing the image's contents.

4.1.1 Percolation Threshold Recognition

In images of pore space at thin section scale, or for larger scales, permeability channels in reservoirs, semantics change somewhat in descriptions of the fluid flow processes. Although it is relatively straightforward whether a two dimensional percolation threshold has been passed, characterizing the quality of that flow connection in an image is more challenging. At the same time, *the quality of observable flow connections is of fundamental importance to the study of flow properties.* And so one of the classical approaches to this problem has been to identify the position of medial lines or "median channels" within an image, reducing the wide passage-

ways of the pores in the image to their corresponding medial axes, then determining whether this line, rather than the pores themselves, extend from one side of the image to another, permitting percolation. In fact, the number of pels that occupy these lines are many fewer in number in almost any thinned image than is the original, relatively large two dimensional array that contains the original image.

With fewer pels involved describing the flow path in the transformed result, if the necessary information remains, then any analysis of these flow paths is greatly simplified because it can work with much more succinct data. It is the paradigm of computer vision work that one must reduce the information to make it useful: from hundreds of thousands of image pels, to a line drawing via thinning, to a symbolic structure via feature extraction, finally to a decision, yes or no. For rock fluid flow problems, this is exactly why I have worked on FMT, to reduce automatically our readily available digital image data to thinned representations that contain enough information for numerical modeling. With this reduction, modeling enjoys a major advantage in numerical efficiency. Further, if some encoding of the width or throat size of the image is made at the time of conversion from pore blob to line, it is possible to characterize multiply connected systems and go beyond a recognition of percolation thresholds, to attack the problem of feature connectivity—how well are two image features connected?

4.1.2 Mapping Networks Images

A companion problem, an extension of the first one, is the preparation of numerical maps which allow one to apply the results of network modeling to data that is, in its original form, intractably detailed. Again, for many of the same reasons, a thinning transformation operating on the original image array can work from the original to yield an identically connected network of one-dimensional "pipes." These pipes are automatically computed capillary tubes of varying width as were used in Fatt's 1956 permeability model [21]. While that model was concerned with time-varying

flow, in the context of his of flow network description, FMT allows one to reduce digital image data to a form that facilitates a numerical encoding of that image's equivalent capillary network.

This numerical coding permits one to make a deterministic measure of the image's flow properties in the context of that model. While this model remains linked only statistically with to the bulk properties of the original rock, the connection is a numerically comprehensive one. The image representation model is linked through a greater number of data points to the original rock than are analytical models calibrated by physical tests that measure an effective property implicitly—like the scalar results of fluid permeability or acoustic velocity—and so the style of effective property averaging is not implicitly constrained by the experiment. Together, the wealth of data in a thin section image and its fundamental incompleteness in sampling the rock volume present the investigator with a two-edged sword. I hope that my tools may provide other investigators with a surer and safer handle at the base of that blade.

4.1.3 Fluid Flow Property Measurements

With a digitized thin section image one has many data points linking the sample with the original rock. But how well is this data linked to the original rock property? Thin sections provide us with samples of the spatial distributions of rock properties. A link between the image and the bulk model should contain spatially distributed rock property information. The thinning approach allows one to characterize the image heterogeneity despite the fact that the spatial information in a given volume of image data may be rather sparse. It allows one to extract those few small details from the large image array. Vectorization or raster coding of the thinned image can convert it to a numerical network model. Why is this of any use? True, I am suggesting that one use image data to run a flow simulation rather than make a bulk measurement of permeability. The difference is that symbolic description of pore

heterogeneity can create a proxy system; an irregular network that is as complex as its topology, not its gridding. If these systems can be solved faster through the reduction of their size by symbolic description, then it may in fact be far easier to run flow simulations than at present. In particular, for situations where only fragments of a sample are available, the image analysis approach may be possible where bulk measurements are not.

Given a numerical model based on these extracted details, and applying boundary conditions that are consistent with the scale of the rock property being estimated from the image, a scalar measure of the connectivity may be evaluated so that variations in this property, namely the connectivity of the pore space and its changes with direction may be studied. The approach of FMT is neutral: no boundary conditions are assumed in its reduction of pores to a symbolic network. Rather, all possible b.c. are made easier to solve for through reduction of data bulk, from raster image to network description. In studies of fluid permeability at many scales, such as for reservoir description, a thinning transformation facilitates an automatic derivation of network models proceeding in an image-dependent way that permits comparative study of images prepared by many means at many scales in a uniform, objectified way.

Network models at varying scales can be linked together with existing techniques like the renormalization approach of Madden [36], which formally relates fracture flow properties in smaller scale models to their effects in incorporated into larger, more global models of flow properties including those for which no direct image is available. The tools I present serve a similar goal, with the difference that my technique is presently limited to two dimensions versus Madden's three.

For image studies aiming at evaluating fluid transport properties, fast Montoto thinning and its associated vectorization provide an image analysis tool to simplify voluminous image data into more tractable circuit representations. With thinned results listed by a curve vectorization algorithm, one which identifies individual ob-

jects and shapes in the image, then a numerical network model may be automatically constructed from an image of pore space as viewed in thin section, for example.

4.2 Thinning's Efficient Summaries

Since the early eighties in the U.S. [17, 19, 46] and since the seventies elsewhere [37, 47], image connectivity analysis has been applied to geological problems. Certain analytical pathways have become well-traveled. Many estimates of permeability derived from thin section have been based on multiple regression of textural image attributes (like the shape measures mentioned above) against measured bulk permeability. In other words, certain object textural properties, like pore size, aspect ratio, boundary roughness index and others derived from images have been tabulated. And in separate experiments, the fluid permeability of bulk rock samples has been measured when thin sections are cut from near the ends of plugs used for the bulk measurement [47, p.238].

Cluster analysis is used with data from those collections of thin section images statistics from a formation where permeability varies and is used to identify those textural measures that are variable between sections of different permeability. Then multivariate regression is used to find coefficients for the various image statistics when they are linearly combined to fit the measured permeability trend in a least-squares sense. Those statistics which do not change much between sections from differing permeabilities are not very useful in estimating permeabilities from image attributes. Those which do vary can be used with coefficients from the regression to predict permeability from the thin section image attributes alone. Since most thin section images' pore space indicator functions do not have very good connections between pores, because of their two-dimensional sampling of three-dimensional space, the statistics which characterize these isolated, oddly shaped regions have been used together with the elementary morphological operations of erosion and dilation and operational experience with these attributes to show that certain statistics are well correlated with the permeability from one formation to another. Certain properties that are measured from the results of erosion and dilation of the original indicator are also useful.

The question naturally arises, is the thinning transformation presented here just another textural attribute, something to be added to the list, now numbering about 300 [18] of known pore space textural attributes, or is it something different? If it is different, can it provide by itself all the information which is now used in these regressions, or at least those that are commonly found to be useful in a variety of formations, or must it be measured in addition to standard texture in images? In other words, does Montoto thinning allow users of rock image analysis to build on known experience? Does the thinning transformation offer anything to rock image analysis that, say, calculation of co-variograms would not offer?

My conjecture is that, when thinning (FMT) is paired with feature extraction (CSC,) yes it does. In fact, the thinning transformation is a unified approach to the calculation of those morphological properties that have consistently been found most useful in the characterization of images for fluid flow properties. Also, the Cederberg-Sobel curve vectorization allows measurement of standard texture statistics that are not directly provided in the result of thinning.

4.2.1 Pore Throat Size Dispositions

In the late seventies and eighties, the use of distributions of pore throat sizes, computed from morphologically transformed digital rock images, has been a part of both image property regression for bulk permeability [17, 52, 53] and the computation of an effective pore throat size [14]. These distributions are available from FMT output, but so is something more valuable: the connections among these pore throats, sampled in thin section. Distinction exists, however, between what is used as the state-of-the-art in measuring these distributions and what is the result of thinning transformations. The state-of-the-art involves eroding the pores seen in the image and then counting the number of pores which remain in the image, broken by that particular thinning cycle. When a pore is broken or split, presumably the erosion process, which strips outermost pels from the objects, has caused an isthmus be-

tween two larger pores to break. The number of such breaks or disconnections is tabulated for every successive cycle of erosion. Histograms of these pore throat sizes inferred from the cumulative thickness of erosion that had occurred at that cycle are used to characterize this important part of the image's texture.

The thinning transformation result, however, takes this throat size distribution and goes further. It measures not only the number of throats which are disconnected at a given iteration, but also the position within the image of those throats which are disconnected at that and every other iteration. At the same time, the area of the feature that remains at each iteration is tabulated as part of the thinning's output. All the information of the number of objects remaining in a particular erosion cycle may be derived directly from segmentations of the transformed result, while retaining within the transformed result that information needed to compute the centroid, or center of mass of those objects. For many elongated shapes, the thinned result alone can be used to calculate their aspect ratio and orientation as well.

Since it appears that geological image analysis has progressed for many years without finding the need for determining the spatial disposition of pore throats, what good could this new measure possibly do? Well, the most straightforward use of the thinning information that I see is its utility in computing a network model of the pore space, derived from the image, to dramatically reduce the cost of recalculation after changing boundary conditions. The thinned result, when histogrammed, provides pore throat size distributions. Extracted listings from CSC define throat size distributions throughout the image. Smaller throat sizes in the left side of an image and larger throats in the right side could be detected from FMT image listings.

Further, the orientation of the throat sizes, at various sizes, can be measured from the tangents to curves in thinned results. Working only from the thinned result, it is possible to get some information on the objects' aspect ratios, although this

information is not as detailed as either the original image, or a boundary coding. It is, however, easier to compute. Details such as true maximum and minimum diameters of objects, and orientation may be more accurately computed from a boundary coding of the original indicator function. Chain codes are recordings of the way one steps from one point to the next along the boundary of an object, or along some line in a gridded image. The Cederberg-Sobel compiler algorithm performs this coding for either thinned or blob images.

4.2.2 Thickening's Additional Connectivity Measurements

Thinning may be used in some rock material images to evaluate attributes that relate to the frame or matrix properties which may have a higher level of connectivity from one part of the image to another. Similarly, thinning can be used to study disconnected but closely packed objects by performing a pre-dilation or morphological closing of gaps less than a certain scale in the image, like all gaps less than 20μ. With this pre-processing, the original image of disconnected pores is first dilated and then thinned. This can allow one to approach in a computationally inexpensive way an estimate of connections which may occur just outside the observed thin section plane. This pre-closing of the pore space is a global operation, homogeneously expanding all pores in the image, optionally in an anisotropic way. This approach may be modified depending on the pore space anisotropy to close or dilate in ways that may minimize the number of artifacts introduced into the image. When the pre-dilation is followed by thinning, as opposed to the normal procedure of following it by erosion, then those pores which have coalesced, joining one another across narrow straits in the original image, are not broken when thinned and the bridge between them remains connected in the final thinned result.

This consequence of pre-dilation, when applied to images that are to be thinned, enables one to study the connectivity of a related image, scaled in a connectivity sense, which may be significantly more connected than the original, but determin-

istically related to it. Thus the simple morphological procedure of dilation can improve the degree of connectedness in the image. Although this violates the regular model [47, p.144], pre-dilation or thickening of the image before thinning improves the processed result in two ways. First, and of greatest concern to network modelers is that the connectivity can be improved through means of an inexpensive global computation. Secondly, not so obviously, is that the surfaces, or boundaries of the original objects are smoothed by the dilation. As such, fewer artifacts are produced as when, during the thinning process, a single pel protruding from an otherwise smooth boundary can case an arm or spur in the final skeleton. And so, pre-dilation results in network models which are both better connected and simpler, lacking some of these spurs and having "main tracks" that run further than those that often result from thinning of an indicator function directly.

The more cycles of pre-thickening applied to an image before thinning, the more objects will tend to coalesce into larger, interconnected objects. In this way, an indicator function image can be evaluated for such concepts as near percolation, near connectivity, or potentially, an estimation of the influence of the near neighborhood on the connectivity of pores in a thin section image.

4.2.3 Boundary Measurement With CSC

To reiterate, by itself, thinning provides image analysts with a unified approach to the computation of image connectivity properties which are presently discussed in terms of pore throat size distributions. Not only does the pore throat size information remain, but also where in the image and how it connects to other pores in the image is also available, all from one process. The analyst also gains an additional benefit of measuring the spatial disposition of various sizes of pore throats. But together with curve vectorization, thinning provides not only all the common textural attributes but also new types of image information which is of particular relevance to the evaluation of flow properties at many different scales.

The aspect ratio of many classes of pore elements is available through both boundary coding (where a feature extraction program lists the perimeter of all imaged pores—as is done by the CSC algorithm given in Chapter 1) and derivations from thinned results. This, together with pore orientation, permits study of the anisotropy of imaged rock properties. A count of pores or pore elements and their size range can be studied as presently practiced in petrographic image analysis. By the approach of boundary coding the original objects, as described in this paper, thinning as described in this paper, then re-applying the boundary coding to the thinned result, I suggest it is possible to derive all currently useful textural statistics, as well as a basis for an automatically determined circuit model representation of the original binary image.

In addition to streamlining the thinning algorithm of Bel-Lan and Montoto, and completing the algorithm of Cederberg into an unambiguous form, the important contribution of this work is that the two algorithms have been unified into a working collaborative pair, through the Sobel neighbor coding scheme. My conjecture that all of Petrographic Image Analysis' image processing work[1] can be expressed in a unified way through Montoto thinning and Cederberg-Sobel compilation has led to my realization, after implementing these tools, that their uses in the earth sciences are far more general than thin section analysis.

[1] As distinct from the unmixing, principal components analysis, and multivariate regression aspects of Robert Ehrlich's work, which is not addressed by my work, but which could make use of my results.

Chapter 5

Morphological Concepts and Terminology

Many of the concepts just described are based on some knowledge of a subset of digital image processing, or digital picture processing as many practical studies are called here in the U.S., that is known as morphology. Morphological processing, the processing of digital images for properties that are dependent on their objects' shape and changes of shape, is a branch of digital image processing that may have had its basis in geological image processing in the sixties.

Jean Serra [47, p.*v*] reports that he and Georges Matheron were engaged in 1964 to study the problem of mathematically analyzing images of rock pore space to estimate their permeability. In the U.S. the approach to image processing has been described as more one of *picture* processing and many of the applications for which image processing techniques have been developed in the U.S. have been relevant to problems of remote sensing and possibly procedures related to automatic target identification and tracking in weapons systems. Perhaps those of you reading this paper have a background in seismic data processing or digital image enhancement.

Perhaps you have read statements of image processing or seismic data processing algorithms and have experience with the types of notation that are used to describe processing sequences that are then translated into lines of program code. As I have read the literature of morphological image processing in the past three years, I have

found that some descriptions of the processing sequences have used terminology and techniques which differ from those more classical descriptions of linear filters such as two-dimensional Fourier transforms, convolution filtering, and the several algorithms for seismic data migration [9, 10, 12, 13]. For these types of image processing, descriptions in terms of integral calculus, possibly with integration over a number of independent variables, provides an adequate introduction to the concepts of the process involved. In morphological processing, however, the fundamental steps from which more complicated algorithms are constructed is not so familiar, because morphology is based on set theory, which is combined in non-linear ways that are quite alien to the concepts we are taught in classical mathematics. And so, a notation has been developed by those using morphological processing for the longest time with an appreciation for the distinctions between morphology and the rest of two dimensional signal processing, a notation that describes the fundamental operators of morphology, presenting a shorthand that can be used for analytical approaches to morphological problems.

However, not everyone accepts this new, symbolic description of morphological processing, and many morphological algorithms have been published without resorting to the more compact, symbolic notation that is presented in Serra's **Image Analysis and Mathematical Morphology** [47]. Morphological processing may be described using the notation of set theory, simply defining certain sets and including others in expanding definitions of those sets, such as certain features in an image belonging to a set or not. Other algorithms are described less compactly in prose. While this is a base level of description which certainly can convey the nature of how one implements a particular algorithm, the more verbose description of the algorithm does not often help when analyzing a processing sequence to see whether some part is redundant.

One of the three main approaches to morphology which I have found useful are those which rely on the more familiar set theory representation of segments within an

image, that is to say groups of pels within a digital image, then describing collections of these sets and relating these sets to one another to form morphological algorithms. Another approach eschews the use of set theory whenever possible, perhaps because it is unfamiliar to many people, relying instead on more extended prose descriptions of processing sequences in text. Finally, a comprehensive symbolic language, as it might be called, has been developed in the past twenty years by workers at the *École des Mines de Paris* Fontainebleau center for geostatistics and mathematical morphology [47, 37]. Their symbolism provides unique symbols to represent each of the basic morphological operators as well as a symbolic grammar to group these into sequences that represent processing sequences in digital or continuous domains. The latter approach sacrifices familiarity to gain efficiency in the specific task of describing morphological image processing.

5.1 Set Theory, Text, Symbolics

Perhaps it is because the history of mathematical morphology is measured in a score of years, rather than in hundreds, that the symbolic descriptions used by researchers communicating results from morphological processing are not truly standardized. While one could say that a standard set of operators has been proposed [28, 44, 47, 15], either authors choose to avoid the new notation for the sake of clarity in the context of their own specific communication, or they prefer not to use it for their own reasons. Up-to-date notation aside, it is still possible to say many things about morphological processing without using a standard notation.

I present here concepts that do not make use of strictly familiar mathematical operators, both in the fast Montoto thinning algorithm and the related Cederberg-Sobel compiler. The concepts they are based upon are morphological ones to a greater or lesser extent. In this section, I hope to explain parts of morphological processing that may be unfamiliar to those readers who nonetheless have had experience with processing of two dimensional arrays of numbers, such as seismic data.

I also hope to offer some meaningful description of morphology for those who have not had seismic data processing experience. I think the differences between morphological processing and conventional two dimensional linear filtering are great enough that there is a need for a different type of symbolic representation of the processing sequences used.

Perhaps because of the relatively short history behind morphological processing of digital data, less than twenty-five years at this time, there does not seem to be an agreement among those publishing morphological work as to whether a new symbolism is needed or desirable. As I have read the literature, I see three differing approaches to the explanation of morphological processing sequences and concepts that are used by various authors. Some make use of the symbolism of set theory. Others combine this theory sparingly with what are mostly text descriptions of procedures to be followed in a morphological process. Finally, a group has developed a special symbolism specifically addressing the needs of morphological processing, but based on Hermann Minkowski's classical work in topology from 1903, and convex set work from 1897 [15, p.404].

I associate each of the different types of morphological description with an authors who I feel represent that approach well in their writing. I also present summary descriptions of each presentational style in this section, and follow this with sections devoted exclusively to each style.

5.1.1 Set Theory Symbols Definitions

The approach to morphological algorithm description which uses set theory requires knowledge of only a few simple symbolic operators which are already familiar to some people. The expression $\{x : \star\}$ simply describes the set of points x satisfying property $\star$. By itself, this symbol is sufficient to describe image classification conditions where, for example, in the picture to be segmented into feature and not-feature, a certain range of intensity values would characterize features of interest, and the

conditions of that segmentation would replace $\star$ in a description of segmented set x. The symbols $\{\emptyset\}$ and $\Re^n$ describe the empty set, with no members including even zero-valued ones, and the Euclidean metric space of dimension n, where, for instance, Cartesian coordinates are used to describe positions within a *continuous* space. The symbol $\mathcal{Z}^n$ describes lattice space where positions have integer coordinates in dimension n. A point on this grid is a picture element or pel (pixel). This may also be called a *module* structure [47, pp 166–168]

A few symbolic shorthand expressions used in set theory are those for *there exists, for all, implies,* and *if and only if.* These are $\exists$, $\forall$, $\Rightarrow$, and $\Longleftrightarrow$.

The expression $\{x \in X\}$ means that x is a member of, or better *belongs in* X. Of course, $\{x \notin X\}$ means it doesn't. The not condition is also sometimes symbolized with $\{x \ni X\}$ but I prefer the $\notin$ symbol's clarity. The pair of relations $\{X \subset Y\}$ and $\{Z \supset B\}$ describe the set conditions were X is contained as a subset of Y and set Z contains B. This expression is usually used to imply that the set which contains one or the other completely contains it, as in a superset. The expression X^c is used often with binary sets, masks of zones in classified images, where the *complement*, or every part of the image which is not a member of X, is symbolized by X^c. The symbol $\check{X}$ is used to describe a transpose, or rotation by 180° of some set. Often this is used in manipulations of *structuring elements*, which are simply neighborhoods about some point $\{o\}$ called its *origin*, used as a process description when that point is moved about all areas of the image, as in convolution. The transpose is the two dimensional equivalent of a time reversal applied to a kernel in one dimensional convolution. The structuring element is morphological processing's equivalent of the kernel in linear signal processing. $\hat{X}$ is used to describe the reflection of X about some axis in an image.

The collection of binary set operators are probably familiar to readers who learned the "new math." The union of sets X and Y is written $\{X \cup Y\}$. Their intersection is written $\{X \cap Y\}$. Their difference, or that which is in set X and not

x, X	image element (point, pel) x, set of elements X
$\{x : \star\}$	The set of points x satisfying $\star$
$\{\emptyset\}$, $\Re^n$, $\mathcal{Z}^n$	the empty set, Euclidean space, lattice space
$\exists x$, $\forall x$	there exists, for all
$\Rightarrow$, $\Longleftrightarrow$	implies that, if and only if
$x \in X$, $x \notin X$	x belongs to X, x is not in X
X^c, $\check{X}$, $\hat{X}$	complement of X, transpose of X, reflection of X
$X \cup Y$, $X \cap Y$	the union of X and Y, their intersection
$X \setminus Y$	the difference of X and Y, that in X but not Y
$X \wedge Y$, $X \vee Y$	logical AND of X and Y, logical OR
$\{o\}$	The origin pel or point of a structuring element

Figure 5.1: basic logic and set theory symbols

in set Y is written $\{X/Y\}$ or sometimes with a backslash as $\{X\setminus Y\}$ to distinguish it from division of numbers. For set relations, the boolean operations AND and OR, symbolized by $\wedge$ and $\vee$ can be used to combine test results. Defining a set Y with pels between brightness levels 25 and 50, or equal to 255, which is to say members of input image X such that elements x are greater than 25 *and* less than 50 *or* are equal to 255 can be symbolized by this:

$$Y = [\{x : x > 25\} \wedge \{x : x < 50\}] \vee \{x : x = 255\}$$

These basic set theory operators are sufficient, together with a descriptive text, to define most any morphological processing sequence. The symbols presented in this section and that I will continue to use summarized in the following figure.

5.1.2 Text, Templates, And Sets

While the set theory symbols just described are adequate to specify morphological processing sequences, they are not ideally suited to the task because certain morphological sequences, some not describable succinctly with these symbols alone, are used repetitively in many processing sequences. A reasonable compromise in description of certain morphological process features like structuring elements, has

been to use these symbols and then link them together with a text description of how they are to be combined. Usually, morphological processing depends on one or more structuring elements to perform its task in the way that a convolution sequence in linear filtering uses a kernel of weights in a neighborhood to specify coefficients for a spatial linear combination.

These structuring elements are often expressed in graphical form as *templates* which, defining certain sets, operate on the original image using set statements. This type of approach has been used by several authors whose work I have used including [2, 11]. Using only the basic symbols, the text makes their meaning clearer even if the symbols' relations are not obvious. The templates look like:

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & \times & \times \end{matrix}$$

This template, taken from Cederberg [11], tests conditions in a morphological filtering process. To begin with, consider the image to be operated on as a binary one. Then, since this structuring element is centered, or has its origin $\{o\}$ at the central element of the 3×3 neighborhood, the central $\{1\}$ means only those elements in the image that are one-valued will be tested, zero elements are left zero. The morphological test implied by the template above requires that all three neighbors in the row above the tested points must be zero. That is represented by the three zeros. Further, the neighbor to the left of the tested element must be zero, while that to the right must be one. Finally, the lower-left neighbor, on the following row, must be one, and the other two to the right of it we don't care about, which is what the ×'s represent.

What does this mean? The morphological test performed by running this structuring element all around the image is called a *hit-or-miss transformation* and this particular structuring element would find those pels that are the first elements of an object that a raster scan touches when advancing left to right, top to bottom in an image. By including the "don't care" conditions in the two lowest right members

of the structuring element, this test would work for a single-pel wide line (both ×'s would be zero), a filled-in object (both would be ones), or a line joining an object (one or the other would be one-valued). So of the logical tests applied to the eight nearest neighbors, the two don't care conditions mean that out of all possible neighbor conditions, four (all combinations of ones and zeros in the ×'s) are defined by this structuring element.

In fact, this is a useful processing test as well as a useful example. It demonstrates the difference between nonlinear image filtering, with its linked series of neighbor tests possibly containing logical combinations of tests, and linear image filtering, where all neighbors specified are always summed in linear combination, though possibly with adaptive weights. Linear image processing is called two dimensional signal processing, and nonlinear image processing is mathematical morphology in two dimensions.

A compact way of transforming a binary image into a multivalued or gray level one that represents this neighbor data is called a *neighbor coding*, after Sobel [48], and also called a *local configuration* in the nonspecific sense used by Serra [47]. In it, the boolean results of neighbor tests are placed in a bitwise sense into pels of a 256 gray level image. In that way, the gray level value represents a mapping of neighbors from spatial to gray level values in the range [0 − −255], or eight bit image depth when including the information from tests on the eight nearest neighbors. Directional information is encoded by this in bit slices of a gray level image. With this mapping, any given boolean test on an image can be neighbor coded into an 8 bit deep image of the same size. All possible arrangements of nearest neighbors can be represented in this transformed image, and several boolean tests can be transformed in parallel in some machines. I will discuss this more later, but consider that the structuring element template above can control a hit-or-miss transformation applied to neighbor-code transformed data by simply selecting those four values out of 256 that correspond to that structuring element's local configuration.

As an alternative to the template description of structuring elements, one may use the set notation together with coordinate subscripts to describe structuring elements in a more classical fashion. For a raster scanned (left to right and top to bottom) image, A, image elements $a_{i,j}$ have right hand neighbors $a_{i+1,j}$, and lower neighbors $a_{i,j+1}$. This type of notation is used in text describing convolutional approaches to image processing [12, 9, 40]. Compared to the templates, these descriptions, however appropriate when listing convolution coefficients, take far too much space to describe the set of logical tests implied in most structuring elements. For convolution, such descriptions can be easily translated into FORTRAN statements of array operations *mutatis mutandis* to the direction of scanning versus its storage in a file. However, these array index notations have fewer constructive uses when describing nonlinear image processes, particularly without benefit of a morphologically oriented programming language—such is the case for many who implement morphology in parallel processing machines. And so, the templates seem a good compromise between intelligibility and conciseness.

The somewhat conservative approach of using a minimal amount of esoteric symbolism together with a slightly more verbose descriptive text has been a reasonable way of describing morphological algorithms. Really, I find the approach quite adequate for describing a single algorithm if that is the extent of the paper. When analyzing morphological processing sequences, however, the more verbose nature of the text description together with the cumbersome symbolism of simple set theory combine to impede one's understanding of more complicated morphological processing. For these problems a special symbolism has been developed.

5.1.3 Fontainebleau's Morphological Symbolism

This symbolism, compact and specifically created for the needs of morphological analysis, has been popularized by Serra and Matheron of the Fontainebleau school. They have given credit for the basis to earlier work by Minkowski by way of Had-

wiger. [47, p.43] The same way that a two dimensional convolution is performed by making a weighted average of neighbors for every point in an array, replacing that element with its weighted sum, one can conceptualize the convolution process as an application of a weighting function after a particular translation of that function, translation by moving the weighting mask around the original image.

A program which performs image convolution would systematically move or translate this weighting function through every pel, often row by row in the same way that a TV scanner moves. For any specific translation from the origin to a coordinate $(x, y) = \mathbf{h}$, one may use the expression $X_{\mathbf{h}}$ for the translate of the image or structuring element set X by the vector $\mathbf{h}$. An example of usage from [47] is $X_{\mathbf{h}} = \{x : x - h \in X\}$ This would also serve to describe the operation of translating an entire image in a direction $\mathbf{h}$, overlaying it with the original untranslated version of the image as a step in two point correlation function calculation. To actually perform correlation following the translation step, one checks to see whether an image element is set to $\{1\}$ in both versions of the image. Then, if $Y = X_{\mathbf{h}}$, nonzero correlations will satisfy $X \cap Y \neq \emptyset$, to describe a scalar result. This statement that set X "hits" set Y is symbolized as $X \Uparrow Y$ implying $X \cap Y \neq \emptyset$.

A generic morphological set or image transformation is symbolized with $\Psi(X)$, also the transform of set X with respect to set transformation Ψ. The symbol of set union can also be used in combination with indices to yield the union of members of a sequence, or limits of translation with respect to structuring elements. This type of description, such as $\bigcup_{b \in B} X_b$ symbolically represents the union of translates of set X by all vectors $\mathbf{b}$ that are in set B. This is the symbolic description of a procedure in Doyen [14]. One of the more common morphological operators that uses sets meeting conditions when translating a structuring element over an image is that of *dilation* when, for all points where the structuring element touches an object pel (set=1) in the original image, the transformed image contains an object pel.

For dilation with structuring elements that contain the origin and have some size, a 3×3 box for example, objects grow in size. Objects can be duplicated when the structuring element in a dilation has two points only, which are some distance apart. This dilation operation is one of the most fundamental morphological operators. It is symbolized with the operator of Minkowski addition, $\oplus$. As a practical detail, in the same way that kernels are transposed before two dimensional convolution, but not before correlations, so are structuring elements transposed implicitly in dilation. Thus the dilation of set X by structuring element B that does *not* transpose the structuring element before applying it around the image is written in the simple form $X \oplus \check{B}$, with the $\check{B}$ representing the transpose or 180° rotation of B, that makes the dilation behave like a correlation. Not every worker subscribes to this convention of implicitly transposing the structuring element before dilation. Sternberg, in the U.S.A., and Ronse and Heijmans in Belgium have chosen not to make this transpose implicit. But for workers on hexagonal grids, this implicit transpose is quite convenient.

Another basic morphological operation is that of erosion, where only those parts of the image objects which can fully contain the structuring element remain after the transformation. This is symbolized in the way of Minkowski subtraction. For example, set X eroded with structuring element B appears as $X \ominus \check{B}$. When the structuring elements are not symmetrical with respect to their transpose, the transpose step is important for geometric consistency, such that a triangular object dilated by an identical triangular structuring element yields a triangle and not a hexagon. This, as well as the suggestion that dilation should contain implicit transposition of the structuring element, is stated in [47, p. 44]. The symmetry induced on the relation between object and structuring element holds for dilation but not for erosion with this convention.

Often the complementary operations of morphological erosion and dilation are combined in sequence. In fact, when erosion or a number of erosions is followed by

dilation or an equal number of dilations, the sequence has been given the special name of *morphological opening.* In the case of dilation followed by a balanced amount of erosion, the operation is called *morphological closing.* Objects in a binary image that are closed may have their outside borders nearly unchanged, while holes within them may be filled in if smaller than the structuring element's size. Similarly, the object's outer border will be unchanged with closing if both it *and* the structuring element are convex [47, p.74]. The definition of size with respect to structuring elements is not trivial [1], but for a simple discussion, it is adequate to visualize the structuring elements here as digital realizations of circular disks from Euclidean space.

To return to the effects of morphological closing, if a hole within an object being closed can not contain the structuring element, then the dilation step will fill in the hole. Subsequent erosion, which will return the convex outer borders to their original position, will not recover such holes because no record remains of their presence in the dilated intermediate result. In this way, image objects which have holes can have them filled in while preserving the outer borders of shapes, provided the dilation keeps each object's outer border distinct from its neighbors. This is a typical example of nonlinear filtering—one part of an object is unchanged while another part changes, even topologically, elsewhere in the image. Morphological closing of a set X by structuring element B is symbolized by X^B.

Morphological opening, on the other hand, is a sequence of erosion followed by a matching sequence of dilation. In this procedure, objects that are smaller than a certain *size*, namely those which cannot contain the structuring element, will disappear in the intermediate image, eroded away, so to speak. Then, the subsequent dilation is unable to recreate those objects from $\{\emptyset\}$ and so opening becomes a technique for *sieving* sets of convex objects, or studying local connections between the larger portions of non-convex objects. With opening, particles smaller than a certain size,

[1] In particular, it involves the Hausdorff metric [47, p.74]

set by structuring element B, are removed from the collection of objects in an image. Given both convex objects and convex structuring elements, the remaining objects are unchanged. In any case, the convex hull remains. Morphological opening of set X by structuring element B is symbolized by X_B.

For non-convex objects in particular, opening will not recover well the original boundary, and at the limit of sieving, when the erosion step leaves only a single pel, the result of opening will be to represent the object by a copy of the structuring element used to dilate that single image element. A picturesque example of opening performed with a hexagonal structuring element is on the cover of paperback editions of the second printing of [47].

A very general morphological transform was defined by Serra in 1965, [47, p.39], it is called the *hit-or-miss transformation.* Essentially, morphological processes which involve erosion, dilation, or sequences thereof, can be described as special cases of the hit-or-miss transformation. The generality of the hit-or-miss transformation is drawn from its property of using a pair of disjoint structuring elements rather than just one. The development of this transformation was also drawn from the observation that an erosion of a set X by B is equivalent to the dilation of its complement, or symbolically:

$$(X \ominus \check{B}) = (X^c \oplus \check{B})^c \tag{5.1}$$

The result of the hit-or-miss transformation is the intersection between two erosion operations—one operating on the object X and the other on its complement X^c. The effects of the hit-or-miss transformation are controlled by some structuring element set T. To specify a meaningful hit-or-miss transformation, one that changes the object in an interesting way, the two members of the structuring element set T, T_1 and T_2, must be *disjoint*, which is to say that when writing a template to describe T, an exclusive-OR condition must hold between T_1 and T_2. Any space in the template occupied by an element of T_1 may not be occupied by an element of T_2.

The symbolic notation for the hit-or-miss transformation, in terms of the $\ominus$

operator already discussed, contains an implicit assumption that not only are structuring element set members T_1 and T_2 disjoint, but also T_1 contains the origin $\{o\}$ of the structuring element set and operates in terms of "hitting" the object, while T_2 necessarily does not contain the structuring element's origin and operates by hitting the object's *complement.* So T_1 structures a hit test, while T_2 structures a miss test.

In the hit-or-miss transformation these are combined with the intersection to yield the following definition [47] for the hit-or-miss transformation $\circledast$ of set X by structuring set T:

$$X \circledast T = (X \ominus \check{T}_1) \cap (X^c \ominus \check{T}_2) \tag{5.2}$$

One way of misinterpreting this expression would be to consider it equivalent to symbolically express the following morphological expression, using $\circledast$ rather than $\circledast$ to symbolize the difference that in fact exists, where T_1 and T_2 are not disjoint but both contain $\{o\}$, a small but significant difference from the conditions mentioned above. This creates a different transform that I will explore in some detail.

$$X \circledast T' = (X \ominus T_1)^c \cap (X \oplus T_2) \tag{5.3}$$

Instead of the set intersection, consider the exclusive-OR logical operator to be the relation. If the exclusive-OR condition is symbolized by $\triangle$, then it is defined by the following expression in terms of logical AND ($\wedge$) and logical OR ($\vee$), allowing for operator negation with a through-going slash /.

$$X \triangle Y = (X \not\wedge Y) \wedge (X \vee Y) \tag{5.4}$$

Now, consider the alternative definition of the hit-or-miss transformation symbolized with the five point $\circledast$ in equation 5.3. Albeit a less transcendentally useful one, it is illustrative and useful to a specific problem in morphological image processing. Namely, consider the expression of symmetry between an erosion (or dilation) of an object by a single, unchanging structuring *element*, rather than the disjoint set of complementary conditions that are implied whenever a hit-or-miss structuring

element set T is defined for the transformation symbolized with $\circledast$, Serra's hit-or-miss transform. Just complement the second term, and remove the transpose operators, leaving the definition of erosion or dilation used to implicitly transpose its structuring element. This makes a transformation, given a symbol reminiscent of signal processing's convolution, that runs its structuring elements over objects in the image in the transposed way, while the hit-or-miss transformation actually uses the transpose symbols over its structuring elements to perform, in a symbol consistent way, the morphological equivalent of cross-correlation. Thus the hit-or-miss transformation is symbolized with a $\circledast$ while the following expression is symbolized with a $\circledast$. Note the five and six point star distinction between the symbols.

Then, a morphological expression which determines the weighted (in the sense of width) and biased (in terms of the median's being placed inward or outward of the object) *boundary* of the object can be computed by:

$$X \circledast T = (X \ominus T_1) \Delta (X^c \ominus T_2)^c = (X \ominus T_1) \Delta (X \oplus T_2) = (X \ominus T_1)^c \cap (X \oplus T_2) \quad (5.5)$$

For example, a boundary could be determined within a region one pel outside the object's boundary, and three pels within. This is a crude digital equivalent of monolayer adsorption within pores. Assuming a 1:1 (non-stretching) affine mapping of the physical pore's shape into digital scanning of that space, surface area within partially saturated pores could be studied morphologically.

The second half of the expression for $X \circledast T$, the dilation, finds regions outside the objects, and the complement of the object's erosion intersected with it defines a region within which the boundary is output. For this boundary-defining use of $\circledast$, both T_1 and T_2 may contain the origin. When the structuring elements are not symmetrical, this operator can find boundaries of objects facing certain directions. In that way, it can perform the same function that is performed by a subset of hit-or-miss transformations defined for the thinning of set X.

A directional erosion behaves mathematically like sunlight melting snow while

illuminating from a particular direction. For conciseness, I choose to call directional erosion *ablation*, to recall the physical reality that is mimicked by certain structuring element sets whether used in erosion, hit-or-miss, or boundary transformations.

For however much the previous example's words stated a morphological relation, the following symbolic description may make more sense, even if the nomenclature is not fully endowed with meaning. And so to consolidate these symbols in a displayed fashion, one can say:

$$X \circledast T = (X \ominus T_1) \,\Delta\, (X^c \ominus T_2)^c \tag{5.6}$$

This I would read as "the boundary transformation of set X by structuring set T is the exclusive-OR of the erosion of X by structuring member T_1 and the complement of X-complement eroded by structuring member T_2. I have described this result qualitatively as that set which contains the border of all objects within certain spatial, perhaps unsymmetrical regions about the border of objects. More simply:

$$X \circledast T = (X \ominus T_1)^c \cap (X \oplus T_2) \tag{5.7}$$

Following this, a particular boundary transformation defined by a structuring set T weighted to contain more of an object within its border and little outside can be used as part of a *thinning* transformation—one which reduces object thickness and does not disturb object connectedness. This thinning could be described as the difference between the original image and this boundary set. Despite the requirement that a particular structuring set T is meaningful within only certain subsets of all possible structuring elements, thinning be described iteratively as the set difference between X and this boundary region found by $X \circledast T$ or $X \circledast T$ as:

$$X \bigcirc T = X \setminus X \circledast T \qquad \text{or} \tag{5.8}$$

$$X \bigcirc T = X \setminus X \circledast T \tag{5.9}$$

And in an analogous way, the *thickening* of an object, once one has a concise symbolism for the hit-or-miss or boundary transformations, can be described sym-

bolically as

$$X \odot T = X \cup X \circledast T \tag{5.10}$$

Consider as useful those structuring sets T that are weighted more towards the regions outside the border of object, for thinning and thickening. For most single pass processing sequences such as erosion or dilation alone, one or the other of T_1 or T_2 would be zero, in a hit-or-miss transform, which is to say not $\{\emptyset\}$ but $\{o\}$, the set that which does nothing in either erosion or dilation.

Because these boundary transform and set difference operations are run iteratively, one might wish to summarize a sequence of the thickening and thinning operations in the same way that sequences of the erosion and dilation pair are summarized with the opening and closing operators. This requires a more symmetrical structuring set, in the sense of T_1 and T_2 in both operations having comparable diameters. The terms $X \bigcirc \{T^i\}$ are used to describe *sequential thinning* operations that imply more a geometric sequence of operations than do morphological openings. Sequential thinning implies the following:

$$X \bigcirc \{T^i\} = [\ldots[(X \bigcirc T^1) \bigcirc T^2] \bigcirc T^3 \quad \ldots \quad] \tag{5.11}$$

A final symbolic process description is introduced here by way of example, and that is *conditional processing.* The conditional processing operator acts on the set to its left, applying the conditions to its right, with the semicolon symbol, ';'. That is, it operates on a set specifically in the sense of a post-processing filtering. The semicolon symbol is followed by a conditional set, which may be the transformed result of the set to the left of the semicolon. For instance $\{X\ ;\ Y\}$ where $Y = \Psi(X)$ describes a conditional processing sequence that computes some transform of set X then relates Y to X with an intersection, exclusive-OR, or other combination of the sets before proceeding. The conditional operator is quite useful as a nonlinear control on sequential morphological processes. With it, many classes of morphological operations can be expressed symbolically in a way that bridges the gap between the Euclidean space of continuous theory and the approach by which such an operation

is actually computed in digital space. Also, the conditional operator symbolizes a typical construct often included in algorithms from workers involved with *nonlinear* image processing, where a morphological process that has no analogy in linear filtering can not be described within a mathematical expression, but can be linked into an algorithm in the form of a subroutine call.

To view what type of transformation might be meaningful in Y's position, consider the hit-or-miss or boundary transformations. If one were using the conditional operator with, say, sequential thinning, then *conditional sequential thinning* could be represented symbolically, and one need only write:

$$X \bigcirc \{T^i\}; Y \tag{5.12}$$

If conditional set Y were the result of a transform that identified those elements of X that are part of, say, an eight-way connected line, or a single-pel wide feature connected to adjacent elements in any of the eight principal directions in a rectangular gridding, then the conditional processing step may cause the overall transformation to save all elements meeting that condition, elements that would otherwise be eliminated in the next step of sequential thinning. If conditional step Y can never modify the set to the left of the semicolon, then its presence in the processing sequence is redundant.

5.1.4 Morphological Presentations Review

I presented this section while intending to introduce the reader to what I find to be the range of descriptive techniques used in discussions of mathematical morphology and related nonlinear image processing. This range of description may be quite broad. In fact, if the sole purpose of this paper were to describe the fast Montoto thinning transformation, then at one limit of the symbolic description process, the most verbose one, a single morphological transform may be expanded to a small tome. At the other extreme, one can combine the symbolic elements just now presented, the *Fontainebleau* symbolism for mathematical morphology, from Serra [47],

to express the structure of the fast Montoto thinning algorithm in the following *conditional sequential thinning* sequence:

$$X \bigcirc \{T^i\}; Y_i \;=\; [\ldots[(X \bigcirc T^1; Y_1) \bigcirc T^2; Y_2] \bigcirc T^3; Y_3 \quad \ldots \quad] \tag{5.13}$$

To describe the same thing! Structuring set T^i yields is an inside-weighted boundary transformation, a realization of a hit-or-miss transformations that is used to sequentially thin a set X. Conditional structuring set Y_i identifies those members of the thinned image set, at that iteration, that are part of single pel wide objects in the image, and records them together with a measure of width implied by that iteration (hence the subscripts on conditional sets Y_i). With some elements recorded, all those under consideration are removed from the active set, and then thinning proceeds once again. Since thinning by a nonzero structuring element will eventually erode or ablating the entire original set X, the set of indices of those members preserved in the processing sequence at index i are the result from conditional sequential thinning.

Fast Montoto thinning, which I describe in this paper, may be more exactly expressed with:

$$\mathcal{X}_i \;=\; \bigcup_{i=0}^{\text{maxwidth}} \Psi^i(X_i) \qquad \text{where} \qquad \Psi^i(X_i) \;=\; X \bigcirc \{T^i\}; Y_i \qquad \text{recursively} \tag{5.14}$$

Where $Y_i \;=\; X_i \setminus \{mlc\Psi^i(X_i)\}^c$, and mlc is a medial line check, to identify that a neighborhood is part of a one-pel wide feature. Surely this description of the fast Montoto transform is among the most compact expressions for it possible. In detail, though, it lacks explicit statements of the structuring element templates used in T, and the medial line test templates used in Y. These are given in appendices. The following section returns to more detailed descriptions of the technique used to describe morphology by Montoto and by Cederberg. In doing so, it reviews Bel-Lan and Montoto's work. The section after that reviews Rosenfeld's technique as an example of the approach to morphological processing found in the U.S.A. I return

to the Fontainebleau school's symbolism in the section after that, and finally discuss my hybridization, both of algorithm and presentation technique in the final section of this chapter.

5.2 Bel-Lan And Montoto's Thinning

As a first example of the uses of these morphological concepts, I will describe the approach used by Bel-Lan and Montoto [2] when describing the thinning transformation which was a starting point for the work. I am presenting a revised version of their algorithm here. Their paper uses set theory descriptions sparingly and concentrates most of its communication in the description in the text.

5.2.1 Point Neighborhood Descriptions

When Bel-Lan and Montoto described their conditional sequential thinning algorithm in 1981 [2], the division of steps in the algorithm that they presented were different from the steps one might imagine from the symbolic summary given in the morphological symbols of the previous section of this paper. In fact, this shows that the authors adopted a more pragmatic approach to the description of their algorithm, perhaps because they actually implemented a thinning algorithm and tested it, rather than just describing it analytically. But some aspects of any morphological description seem to be unavoidable. Namely, the definition of structuring elements seems to be a common need for any morphological algorithm statement. And so the first definition in their presentation is that of the neighborhood of a given image element. Using basic set theory [2, p.39] , the set $V_{i,j}$ equals an image element of matrix A such that element $a_{p,q}$ is a member of

$$V_{i,j} = \{a_{i,j} \in A : b((i,j)(p,q)) = \max(|\, i-p \,|\, ,\, |\, j-q \,|) = 1\} \qquad (5.15)$$

This is a way of symbolically describing the 3×3 neighborhood centered about every image element, with the proviso that there is a one pel wide border surrounding the processed image. That type of zero-padding prevents edge-based computational artifacts.

It is also important to have a set that defines the object part of an image, $S_N = \{a_{i,j} \in A : a_{i,j} = N\}$ describes that part of the image that is considered

the object at a particular iteration, where N = *iteration number* in the sequential thinning process. The interior of the object, within S_N, is simply those pels which are not edges of S_N. This condition defining the interior set can be made hard, by requiring that all eight nearest neighbors are members of the object as well, or may be softened by requiring that fewer than eight, but a reasonably large number of neighbors, be members of the object set. The hard interior condition can be described by $I_N = \{a_{i,j} \in S_N : V_{i,j} \subset S_N\}$. Then, having defined those two sets, the boundary elements are simply that part of the object S_N which is not in the interior I_N. This has been described symbolically [2, p.38] by

$$B_N = \{a_{i,j} \in S_N : \exists\, a_{p,q} \in V_{i,j}\,;\, a_{p,q} \notin S_N\} \tag{5.16}$$

With this statement, there is no dependence on the previous definition of the interior of the object I_N, although in a practical sense, it is desirable to have the object partitioned disjointly into either interior or boundary. In other words, it is desirable to have the union of the interior and the boundary be identical with the object so that:

$$\forall\, S_N, \qquad I_N \cup B_N \;\equiv\; S_N \tag{5.17}$$

By relying on just a subset of available morphological symbols, Bel-Lan and Montoto used text descriptions to describe the more esoteric parts of their transformation. Defining set transformation $\Psi^i(X_i)$ for their transformation, they stated that, for every iteration, elements for which $\{a_{i,j} < N\}$ remain valued at their previous value $a_{i,j}$. In other words, despite the fact that the input to their process is a binary image or indicator function, throughout its sequential thinning the transformation works with and ultimately produces a gray level image result where elements in the final image may have values as high as their iteration number.

For those image elements where $\{a_{i,j}\} \in S_N$, the active image elements, further tests are performed. These N-valued image elements are either members of the interior I_N and are incremented to $N+1$, ready to be an object element in the next iteration, or else they are members of the border set identified by the boundary

transformation for that iteration of sequential thinning and are tested even further. The boundary elements are tested for membership in the medial line set, those elements part of a thin line (that would be eliminated by the next iteration) connected in an eight-way sense to each other or to the lower-valued elements where $\{a_{i,j} < N\}$.

As I mentioned in the previous section, this medial line is mainly a set extending through the border of the object, and is at most one pel wide. With this second condition, those pels in the image which are N-valued at the Nth iteration and are border pels are operated on with a medial line identifier conditional processing step. This makes Bel-Lan and Montoto's a conditional sequential thinning algorithm. Again, if the element is in the interior of the object and is not part of the object identified with the boundary transform, it is incremented to $N + 1$ so that it will be identified for use as the active object in the subsequent iteration of thinning. These objects, which are part of the next iteration's feature, are too far within the body of an object to be eliminated at that particular iteration. While thinning could progress faster, no doubt, if the entire border of the object were identified and either preserved or deleted for each iteration, things running faster would not necessarily run better. In doing so, objects of width 2 pels would disappear, going from a non line-width object status to $\emptyset$ in a single iteration! Since this might disconnect interesting parts of the object for no better reason than that they were some multiple of two pels wide, the transformed result is more valuable, better preserving *connectivity* when border pels facing only certain directions are deleted all at one time. The quality of the connected line representation that results from such a transformation is substantially better, and the reduction in speed is accepted in the interest of more topologically accurate results.

The structuring elements which are used at various iterations in the sequential thinning process are not all the same, but are drawn cyclically from a *family* of structuring elements. For the rectangular grid, they are usually drawn from a family

of four which identifies boundary elements in each of the cardinal directions (North, South, East, and West). This directional boundary identification process, when used in sequential thinning to remove pels, I call ablation. This task can be performed computationally by checking the object status of a single neighbor in a particular direction. With an exclusive-OR relation between neighbors in the ablation direction and opposite to it, a boolean combination of two tests may define those members of the object which are in the "border" set.

Without mentioning it, Bel-Lan and Montoto described the hit-or-miss transformation with their border defining relations. I find that the two AND conditions specified there [2, p.39] should be relaxed to just the outside one of the two AND tests, otherwise objects that are already thinned, but not saved by the previous iteration can not be identified in a streamlining of their algorithm. For instance, in a gridded coordinate system where the y axis increases downward like most raster systems, a pel is on the North facing border of an object at the Nth iteration when $a_{i,j-1} = 0 \wedge a_{i,j} = N$. A condition specified in [2] requires also that $a_{i,j} = N \wedge a_{i,j+1} \neq 0$ Since it is possible for a pel to be a member of a single pel wide line, even going into an iteration, and should therefore be considered part of the boundary set, the hit-or-miss transformation (or boundary transformation) structuring element set that is defined by *only* the $a_{i,j-1} = 0$ condition above is actually sufficient to identify those image elements that are N valued themselves and have a North neighbor which is zero, $\notin S_N$. This is the first modification of Bel-Lan and Montoto's algorithm I present, among the other changes I have found reason to make.

5.2.2 Template Tests

Given that certain elements in the image have been identified as boundary pels, [2] specified a set of structuring elements that identify boundary pels that are parts of eight-way connected thin lines in the sequentially transformed image. Together with

v	0	×		0	v	0		×	0	v		×	0	0
0	1	×		0	1	0		×	1	0		×	1	v
×	×	×		×	×	×		×	×	×		×	0	0

×	×	×		×	×	×		×	×	×		0	0	×
×	1	0		0	1	0		0	1	×		v	1	×
×	0	v		0	v	0		v	0	×		0	0	×

Figure 5.2: Templates from Bel-Lan and Montoto's medial line conditioning set, where 1 means object member, v means nonzero neighbor, 0 means zero neighbor, and ×means don't care. As an neighbor code filter, each template represents 2^n of possible arrangements, where n is the number of ×elements in each template.

cyclic ablation, these pels form a sequentially growing set of *medial line* elements that are part of the transform's important outputs. Their set [2, Fig. 1] was fairly complete for those pels that were identified as border elements in *four-directional* ablation, from the cardinal directions.

Of course, it is possible to specify border elements in a rectangularly gridded system from among eight directions. When this is done, elements in the medial line become four-way connected [45, p. 240]. This requires an expansion of the medial line test template set of [2, Fig. 1]. And so, at every iteration, by operating on the elements identified in the border set, those pels which meet the conditions of the structuring elements given for the medial line test, implemented as a hit-or-miss transformation, are recorded or preserved from further ablation. Those which are not part of the medial lines but are merely borders of a fat object facing a certain direction are zeroed, removing them from both the active object and medial line sets. It is possible to perform the four directions of border identification and medial line checking in successive passes for every iteration. This border check, running four times together with the medial line check before incrementing N, is called a *scan*. This multiple scanning is used to improve the algorithm's resistance to boundary roughness [2, p.40–41] when run in sequential processing environments,

rather than parallel. It is even possible to run each direction's pass twice, for a total of eight scans per iteration and immunity from the "single pel protrusion" problem. These projections from an object's boundary often cause extra arcs to be formed during thinning, and Bel-Lan and Montoto's algorithm is particularly immune to this problem. These are the algorithms specified in [2]. With multiple scans per iteration, however, objects may be thinned two pel widths with each scan, and it is necessary to examine, at every scan in their algorithm, a 1×4 neighborhood about each pel, to either assign a width of $2N$ for iteration N to that element, or in certain local configurations, only $2N-1$ to the element.

While this eight-scan step may have advantages in classically coded implementations of the original algorithm, my present implementation of the algorithm, in *microcode* for a parallel processing device, has simplified this approach to have one scan only per iteration of the algorithm. While this approach would have introduced some redundancy in classically coded versions of the algorithm, a processing symmetry arises when all morphological processes in the algorithm's sequence require only that information contained in the 3×3 neighborhood, or *local configuration* of each element in the image. Keeping track of local configurations of only eight neighbors makes an algorithm far more attractive to implement on an array processor. In doing this, the revised algorithm improves the overall speed with which the transform is computed, since the array processor is much faster than the host machine. In detail, the width in a one scan per iteration revision of the Bel-Lan and Montoto algorithm is simply calculated from the iteration number by rounding the iteration number N to an integer the value $(N+1)/2$.

5.2.3 Bel-Lan And Montoto's Thinning-2

As a last detail in their specification, Bel-Lan and Montoto described a modification of the interior set definition, I_N that softens the condition first given and allows just one of the nearest neighbors to have a value other than N. That is, if only a single

one of the nearest neighbors is not a member of the object, the pel is not on a very exposed border, and it is removed from the border set B_N, incremented to value $N+1$ and thereby saved for consideration in a subsequent iteration. This step, creating the *thinning-2 transform* [2, pp 42–45], with its additional conditional step in the algorithm sequence, reduces the rate at which the algorithm thins objects, but improves the quality of the thinned result—just like thinning by ablation rather than isotropic erosion slowed the algorithm's rate of object reduction while improving the accuracy of the thinned result with respect to a medial axis representation identification of object topology and shape, in other words, its heterogeneity. Ironically, this sort of fine tuning of morphological algorithms mimics the need, common in linear electronic circuit design, to reduce a device's gain factor to improve the linearity of its output [30, p.68].

The addition of the thinning-2 check involves a modification to the boundary element detection criteria mentioned above. In the original description, the modification is that the boundary point specification is modified to include only those pels with less than 7 of the 8 neighbors as members of the active object set S_N for that particular iteration N. In symbolic terms,

$$B'_N = \{a_{i,j} \in B_N : \exists\, a_{p,q} \in V_{i,j};\ 0 \leq \Sigma\{V_{i,j} \in S_N\} \leq 6\}. \tag{5.18}$$

There is a technique that can be implemented on an array processor to identify these elements that are excluded from the boundary set at no additional cost. Since these transformations, in their more straightforward form that I use, depend only on the nearest eight neighbors of each image element, they are more efficiently implemented on the array processor. Its speed is such that the less efficient single scan per iteration that I use instead of the multiple scans, to avoid testing of elements outside of the 3×3 neighborhood, is more than compensated for by avoiding any additional computation time when implementing the "thinning-2 transform" instead of the first algorithm given in [2]. These and other modifications that I have made to the published thinning transformation in [2] are discussed later. I have intended the

preceding section to be an introduction to the style of presenting a morphological algorithm description used by Bel-Lan and Montoto as much as an introduction to the fast Montoto transform that I am presenting in this paper.

5.3 Rosenfeld's Hybrid Presentation

Of the three examples of morphological concept presentation that I summarize in this section, each proponent whose work I discuss comes from a different country. Bel-Lan and Montoto [2] publish from Spain. This section, describing Rosenfeld and Kak [45, pp 130–133], discusses work which comes from the U.S. The section which follows describes more of the Fontainebleau symbolism from Serra [47] in France. In [45], Rosenfeld and Kak have used a hybrid text and algorithm statement approach which works well for the survey nature of their book, **Digital Picture Processing.** The term *picture processing* appears to be favored in the U.S. as describing those types of image transformations which are often applied to pictures of natural systems.

Although [45] discusses a wide variety of image processing algorithms, described in a qualitatively sensible way, the example of this section follows a raster tracking algorithm specification, from their second volume. The reason I have selected this particular algorithm to describe an an example of this approach to morphological discussion is because it is a necessary companion to the thinning transform I present here. Since I must also code this line following algorithm in order to apply fast Montoto thinning to relevant geological problems, I have a strong interest in understanding how these line tracking codes work in considerable detail. Another reason I use it for an example is that it is a necessary companion to thinning transformations as well as an important part of a wider class of very useful machine vision algorithms that are encompassingly called *raster to vector conversion* codes.

It may be argued, strictly speaking, that raster tracking is not a morphological problem. For instance, it is not mentioned by Ronse and Heijmans in their surveys of morphology [28, 44], nor is it mentioned by Serra in either of his volumes on Mathematical Morphology [47, 15]. But my work is a synthesis, for the benefit of earth science imaging in general, of morphological thinning with the picture processing "feature extraction" problem. Because I know all too well that morphology

alone is an incomplete tool without feature extraction, I demonstrate Rosenfeld and Kak's treatment of a non-morphological example. They, too describe thinning and Montoto-like Medial Axis Transforms. Historically, Bel-Lan and Montoto's paper was the first that clearly made the medial axis transform accessible to me by giving details, and so I have based my thinning on their specification. I believe I have not lost any generality through this, since my fast Montoto thinning algorithm can in fact be adapted to any medial axis transform requiring only eight nearest neighbor information.

5.3.1 Raster Tracking Described In Text

In Rosenfeld's hybrid approach, algorithms are specified with a few set theory expressions and an outline of a processing sequence described with text for every point in the outline, as in the following, based on [45, pp 132–133]

(0) For the two dimensional array or raster image which is the input to this algorithm, consider a structuring element set E which identifies the endpoints of lines.

(1) In the first line of the image, accept all points meeting this criterion. In a database, begin a list for every such point that will be tracked in subsequent lines.

(2) On other than the first row:

1. For each curve that is tracked, apply a curve tracking criterion to an acceptance region which adjoins every point identified in step **(1)**, to follow lines already identified. The curve tracking structuring set may be called M for midpoints.
2. Also apply the structuring set E to every other element in the row, starting new tracking lists in the database where necessary, and end those where single curves terminate.

3. Using a structuring subset $M_Y \in M$ that identifies points where curves branch (Y's) or join, add new lists to the database when branches form; terminate two lists, with labels pointing to each other's endpoints in the database, whenever two curves coalesce or merge.

(3) When the bottom row of the image is scanned, the tracking has finished.

That is the raster tracking algorithm. The real advantage to the approach of Rosenfeld is that readers learning about the algorithm may readily make sense of it, gaining a good intuitive understanding of, for example, how raster vectorization works. In other sections of [45], algorithms are presented in more detail with examples of how the process affects certain very simple images. That variation provides a more specific understanding of how such algorithms work.

5.3.2 Nagging Implementation Details

Unfortunately for those interested in actually implementing such an algorithm, many important details are left out of that presentation. These important hints make the implementation efficient or even possible. Perhaps because Rosenfeld has amassed annual compendia of picture processing paper titles, each year's bibliography containing up to 1500 papers, he values the approach which provides an essential view of an approach. Hundreds of original papers are referenced in his book's "Bibliographical Notes" section. In fact sometimes it seems like hundreds of *his* own papers are cited there. But then even some original papers leave parts of the algorithm ambiguous, whether for conciseness or to protect trade secrets. For example, an appreciation of the value of the concept of structuring elements would lead most authors to specify explicitly when and how to use the structuring elements that are implicit in their algorithms. When this is neglected, sometimes long passages of text are used to describe what could be conveyed with a simple graphical tool. More recent papers from around the world have been doing this. Some of the references in [45] are more classical works, ones that seem to have been chosen sometimes not

so much for their clarity of expression, but to demonstrate that certain processing concepts originated here in the U.S.A. long ago.

Often the graphical representation of structuring elements will describe many important details unambiguously—details that must be worked out in order to know how to draw the graphical description of the structuring set. I have found that these types of specifics, when missing from an algorithm's specification, require a special effort on the part of readers to resolve the choices that are left ambiguous in many text descriptions. From a pedagogical point of view, the examples of Rosenfeld and Kak [45] are very useful as a survey, allowing one to browse through the mathematical approaches to picture processing and look up details in original papers when they must be known. Rosenfeld's text algorithms help form an intuitive framework for many processing approaches. Unfortunately for me, they have seldom provided answers to the difficulties which must be resolved when implementing a code. Rather, they often provide only a nagging awareness that some numerical technique exists, one which will require much deeper study of the original paper, sometimes even a review of *its* references, before the algorithm can be coded.

5.3.3 Sequential Vs. Parallel Symmetries

My criticism of these intuitive presentations is incomplete. I do not intend to condemn an approach that simplifies an algorithm to enhance a reader's understanding. As the computational technology changes, the details which must be known for a given implementation sometimes change as well. A specification of an algorithm that is extremely detailed may become bound to a specific technology, and thereby dated. I believe that the set descriptions of Bel-Lan and Montoto's [2] boundary tests are tied to those specifications that are meant to be implemented in a classical, structured programming language. These types of specifications, while containing all the necessary details for their implementation in C or PL/1, may also contain an implicit choice of a processing sequence which is felt or known by the authors to

optimize computation in a particular processing model.

The text descriptions of Rosenfeld illustrate the way that a medium of communication can influence the ideas it presents. Neatly ordering process steps in a linear sequence is fine for sequential processing. I argue that a graphical extension to this is essential to express parallel opportunities in an algorithm.

In this paper I present a thinning transformation that is specifically adapted to the paradigm of *raster engine programming.* In other words, without an array processor, my thinning algorithms won't be directly implementable. The reason for this is that few people are patient enough to wait for a general purpose machine to perform thinning, and use some alternative scheme [17]. To try and compensate this trendiness, I also present a raster coding routine—based on an improved version of Cederberg's [11] and Sobel's [48] ideas—that is for general purpose machines of 16 bit or larger processor size, particularly IBM PC-AT and MS-DOS technology and up.

Raster engines are among the latest in a series of computational hardware advances. Far from being the final word, their popularity may only last until the early 1990's before they are replaced by some other type of parallel processor that is hard to imagine, much less to program. But for now, raster engines offer significant advances in speed with incremental advances in technology. They are distinctly different from the classical architectures in terms of programming approach. These machines can be almost arbitrarily configured within a specific flowchart-like arrangement that is adjusted by the machine's hardware configuration. In this way, the paradigm of the raster engine involves choosing among countless reconfigurations of a processor by the program, while classical languages are based on the sequential application of a small set of operators provided by the processor.

This means that the raster engine is very well suited to a graphical expression of its configuration. This configuration is synonymous with a particular step in an algorithm implemented on the engine. Processed results are passed from one step

to another in cache memory or as files.

5.3.4 Idea Consolidation In Prose

To avoid the pitfall of having the entire thinning algorithm which I am presenting very closely tied to the processing technology that I have chosen for its implementation, I will try to present both a symbolic description which may retain its relevance for a broader audience or a longer time, as well as the specific details of the implementation that I have coded and the programming technique which I have developed to make raster engine coding more systematic.

And so I hope that this paper will represent some learning from Rosenfeld [45] in that: prose expended on an algorithm is useful in helping a reader's philosophical condensation, or intuitive understanding of the process, more than it is an aid to others concerned with implementing the algorithm. That requires a separate section. I locate several of these in the back of this paper.

In this pedagogical sense, text descriptions of algorithms are important to familiarize readers with general concepts before diving into the details, and to outline the structure which sometimes becomes clear only after the algorithm is running on one's own system. Another benefit that is apparent after studying [45] is that templates are useful hybrids between text and morphological symbolism for the neighbor descriptions or structuring element specifications common to most morphological processing. These templates, however, belong not with the text but with the specifics of an algorithm—for there, they are indispensable. From my reading of Rosenfeld, together with my effort to implement algorithms on the raster engine from Recognition Technology [42], I feel that an important addition to most text descriptions of algorithms would be to include the author's consideration of whether points in the processing sequence necessarily follow one another or, whenever possible, they could be implemented in parallel.

Vision raster engines provide some hardware support for morphological con-

cepts like conditional processing (the ';' operator). Can an algorithm be simplified through use of this tool? Some simplifications of processing steps can be made to run using transforms like neighbor coding for a *bit-slice* representation of the local configuration, together with common features of array processors such as *look-up tables.* By using a common vocabulary of these more modern architectural features, the gap between scholarly communication of nonlinear processing algorithms and their installation in image analysis systems can be lessened. Thus, in the tradition of Rosenfeld [45], Pratt [40], and Bracewell [10], image processing can be made more accessible without a permeability barrier between the mathematics and the technology by which it is realized.

5.4 Fontainebleau Morphology Symbolism

By comparison with the conservative use of set theory and text description found in Bel-Lan and Montoto [2] and the structured text pedagogy of Rosenfeld and Kak [45], Serra's work [47], in defining a symbolic language for mathematical morphology, is certainly the most ambitious and elegant approach to dealing with the problems of communicating morphological concepts. The unfamiliar symbols that Serra uses may provoke resentment by those involved in image processing, particularly here in the U.S. where the Fontainebleau symbolism is barely accepted. Resentment may arise in those who feel unwilling to learn the meaning of new symbols, and who are satisfied with their mathematical background as it is at this time. Still, for nonlinear filtering work, there are some virtues of the Fontainebleau symbolism to consider.

With its succinctness in representing a complex morphological processing sequence, Serra's symbolism provides one with a vehicle to rigorously describe alternatives in morphological processing. And despite the unfamiliar appearance of some symbols, the processes of mathematical morphology can be made less foreign when seen in the familiar light of convolution filtering of images. After spending several months familiarizing myself with this symbolism, I believe that it is a means of symbolic expression which nonetheless may help standardize the presentation of morphological concepts—many of which are cumbersome to express any other way. The work of Sternberg [50] and Ronse and Heijmans [28, 44] are more recent examples of support for a standard, particularly now that mathematical morphology is branching out into grayscale imagery.

5.4.1 Description Of Alternatives

The symbolic morphology of [47] introduces new symbols to compactly represent the basic operators found in morphological image processing. In Section 5, I described some of the Fontainebleau symbols that are relevant to discussions of thinning trans-

formations. Here I present some more detailed manipulations of the basic processes and some of the basic symmetries in the Fontainebleau symbolic grammar.

One fundamental symmetry in morphological processing is that opening of an object's complement in an image is the same as the complement of the object's closing. This is symbolically represented in the following expression from Serra [47, p.51] for a structuring element B operating on set X as $(X^c)_B = (X^B)^c$. One can specify a proof of this equivalence given the definitions of opening and closing, and a definition of dilation in terms of erosion. The definitions of opening and closing, [47, eqns II-20,II-21] are:

$$X_B = (X \ominus \check{B}) \oplus B \tag{5.19}$$

$$X^B = (X \oplus \check{B}) \ominus B \tag{5.20}$$

While the definition of dilation in terms of erosion, from [47, eqn II-5]:

$$X^c \oplus \check{B} = (X \ominus \check{B})^c \tag{5.21}$$

From this, one may specify a proof of the first relation as

$$(X^B)^c = [(X \oplus \check{B}) \ominus B]^c = (X \oplus \check{B})^c \oplus B = (X^c \ominus \check{B}) \oplus B = (X^c)_B \tag{5.22}$$

Using the symbolic expressions in Section 5, the expression to the right of the first '=' substitutes the definition of morphological opening into a bracketed complement operation. The next expression to the right applies the definition of dilation once to bring the complement operator inside to the expression in parentheses, changing the outer erosion into a dilation. The next expression does this once more to complement the set X itself, changing the inner dilation into an erosion. That forms the definition of morphological closing and completes the proof.

In both opening and closing, the symmetry of using the transpose of the structuring element in the first operation and not in the second is because of the desire, among other reasons, to preserve convex boundaries in regular objects. Since a bounded (finite) structuring element can be completely overlain by itself only at

one point $\{o\}$, its origin, we wish to exploit this condition to balance the effects of bounded structuring elements when applied in succession. For asymmetrical structuring elements S, then, an erosion of themselves by themselves does not reduce them to $\{o\}$, because the erosion operator transposes the structuring element before applying it over the image. However, if we erode the structuring element by the transpose of itself, for example, even asymmetrical structuring elements will reduce to $\{o\}$, and the opening and closing will reflect image properties in light of the structuring elements and not properties of the image mixed with artifacts of the structuring elements. To summarize, the transpose of the structuring element is used in the first step of opening and closing because, for all bounded structuring elements, since [47, p.48]:

$$S \ominus \check{S} = \{o\} \tag{5.23}$$

The importance of the transpose is greater for hexagonal gridding than for rectangular, where elementary structuring elements are more often symmetrical. As mentioned earlier, recent authors [50, 28, 44] do not use the implicit transpose in their definitions of erosion or dilation. This is sensible for all who work in rectangular lattices.

Another advantage of the Fontainebleau symbolism is that it allows symbolic manipulation of the morphological expressions to gain insight to to the processing beyond that provided by the specification of a particular algorithm. In doing so, it avoids the perils of becoming bogged down in the details of an algorithm without seeing the possibility for alternative schemes. Five lines of symbolic expressions can show relationships among processing alternatives with more rigor than five paragraphs of prose.

The first two approaches to morphological description that I have reviewed have been pragmatic descriptions of image processing and neither approach has been so ambitious as to create a new language specifically for communication of mathematical morphology. Serra and his colleagues at the Fontainebleau school have

developed a no-holds-barred creative approach to constructing a new language for to write about mathematical morphology. Whatever their motivations have been, whether to convey more efficiently the concepts which they have developed, or to fortify a bastion from which they can launch accusations of plagiarism onto the field of picture processing or onto workers in related fields, does not matter so much.

In [47, p.408], Serra presents a simile of digital morphology used as a language. Through the structuring elements, those templates that are accepted as graphical descriptions whether used for set theory relations or actual arrays of elements, one scribes the letters of the morphological alphabet. One uses commonly accepted sequences of these morphological operations making use of these "letters" to form the words. Complete processing sequences that extract a desired result from a digital images are the sentences, some of which are grammatically correct and others less so. If only to avoid the encumbrance of using a particular language, English or French, for instance, to describe an algorithm and thereby making it less accessible to foreign workers in image analysis, this symbolism allows one to discuss morphological processing in a useful, and more internationally accessible way.

Part of the fundamental advantage of the Fontainebleau symbol set is an ability to express the same operation more than one way. Sometimes it is helpful to prove two allegedly different processing sequences say the same thing by transforming one into the other using accepted logical constructs, and thereby demonstrate certain algorithmic property rights, or remonstrate them sometimes. This example is from [47, p.390]:

$$X \circledast T \;=\; (X \ominus \check{T}_1) \setminus (X \oplus \check{T}_2) \;=\; (X \ominus \check{T}_1) \cap (X^c \ominus \check{T}_2) \qquad T_1, T_2 \in \mathcal{K} \quad (5.24)$$

Where $\mathcal{K}$ represents the family of compact sets. Practically speaking, this requires that the sets be bounded—an unbounded image would result if, say one were to attempt imaging the silhouette of an elephant in the jungle as a composite of images of its biological cells. The equation displayed above describes two equivalent ways of viewing the hit-or-miss transformation with a structuring set $T = \{T_1, T_2\}$. For

an accepted small set of these structuring element letters, one may use the general expression of a hit-or-miss transformation above for thinning, thickening, or erosion. The T_1 element describes part of a template which tests for neighbors being members of the feature set. The T_2 part of the structuring element describes other parts of the template, disjoint from the first, which test for neighbors being outside of the feature set. Generally, the origin of the structuring element $\{o\} \in T$ is always part of T_1. Viewing this another way, because $\forall T, \{o\} \in T_1 \wedge \{o\} \notin T_2$, those portions of the image that are the object are operated on, and not the complements of the object. With a correct "spelling" of structuring elements, then, it is possible to use the hit-or-miss transformation to define segments of the borders of image objects which bound the object in a particular direction. By selectively differencing these selected parts from the objects, conditioned by not erasing those parts that belong in the medial line set, an iteration of conditional sequential thinning is performed.

Thickening by the identical structuring element T will simply identify those same border points. Unless some allowance is made in T, such as letting $\{o\} \in T_2$, it is hard to use the hit-or-miss transformation in the way of selective dilation, since thickening is the union of the object with these $\circledast$-transformed borders. With a different structuring element, the hit-or-miss transform will attach an extra thickness of border points from particular directions, a *rhiming* operation. I call this anisotropic attachment of pels a rhiming operator because it connotes the forming of rime ice, particularly in the physical situation of blowing wind forming ice on a preferred side of objects.

Rhiming is the symmetric operator for ablation, where a rough ice surface, as may be found in nature, ablates by melting due to solar radiation, inclined some angle above the horizon, illuminating projections from the rough surface with parallel incidence and with an inclination of the radiation such that, in comparison to the inclination of the surface's roughness, very few prominences shadow neighboring prominences. This part of the analogy is important when using the hit-or-miss

transformation for ablation, since holes within objects appear to be dilating on the opposite side from which the object appear to be eroding—objects ablate in the sense of prominences on a glacier's surface, and not in the way that a meteorite ablates in our atmosphere from the outside only.

With a complementary set $T' = \{T_2, T_1\}$, where $T = \{T_1, T_2\}$ is used to structure $\circledast$ during thinning, or a small number of cycles in sequential thinning, then I suggest it is correct to state

$$\text{Given} \qquad X \bigcirc T \;=\; X \setminus X \circledast T \qquad \text{and} \qquad X \odot T \;=\; X \cup X \circledast T \tag{5.25}$$

Then abation-rhiming cycles can be defined as single words in the way that erosion-dilation cycles relate to opening and closing. These may be considered fairly reversible operations on sets $X \in \mathcal{K}$ which are also *convex*.

$$X^{\odot T} \;=\; (X \odot T') \bigcirc T \qquad X_{\circ_T} \;=\; (X \bigcirc T') \odot T \tag{5.26}$$

What these operators should be called is not as important as the morphological concepts they concisely represent. They are really not much different from opening and closing, given the right complexity of structuring element sets. When extended to conditional sequential thickening and thinning, however, differences may arise between these definitions and opening and closing, and they define new tools for automatic representation of heterogeneity. No difference will be found, however between them and definitions of opening and closing that are based on conditional sequential erosion and conditional sequential dilation. Despite the important differences between thickening-thinning and dilation-erosion, no morphological process could be explained with these symbolic operators $X^{\odot T}$ and $X_{\circ_T}$ that could not also be explained in more complex terms of X^T and X_T *mutatis mutandis* to the structuring sets and the conditional processing tests. The proof would simply involve a substitution of the definitions. For convex sets, then, thickening could reverse the process of thinning, even incrementally.

Digital thinning often progresses by ablation, or erosion from among several

directions in sequence. It is often accepted that mere rotations of a structuring element are part of routine approaches to sequential thinning. A given structuring element that progresses through a sequence in, say, four cycles is not given a special subscript in the expression of conditional sequential thinning [47, p.394]. To reiterate symbolically:

$$X \bigcirc \{T^i\}; Y = [\ldots[(X \bigcirc T^1 \cup Y) \bigcirc T^2 \cup Y] \bigcirc T^3 \cup Y \quad \ldots \ ]$$

$$\text{where} \qquad \{X \bigcirc T \cup Y\} \equiv \{(X \bigcirc T) \cup Y\}$$

$$\text{and likewise} \qquad X \odot \{T^i\}; Y = [\ldots[(X \odot T^1 \cap Y) \odot T^2 \cap Y] \odot T^3 \cap Y \quad \ldots \ ]$$

$$\text{and} \qquad X \ominus \{T^i\}; Y = [\ldots[(X \ominus T^1 \cup Y) \ominus T^2 \cup Y] \ominus T^3 \cup Y \quad \ldots \ ]$$

$$\text{and} \qquad X \oplus \{T^i\}; Y = [\ldots[(X \oplus T^1 \cap Y) \oplus T^2 \cap Y] \oplus T^3 \cap \quad \ldots \ ]$$

In these ways, a morphological word is created by the combination of relations such as $X \bigcirc T$ to describe thinning where T represents a particular structuring element set. The structuring set is defined by those parts it tests to be object members, T_1, and those other parts it tests to be object complement members, T_2, in a particular template, together with all basic rotations of that template, thereby becoming included as the structuring element set part of a word. Perhaps it is that structuring elements are the *vowels* of morphological words, since in mathematical morphology, as in many languages, words are more useful when they contain vowels.

A useful, if somewhat more subtle description of a morphological process is that of the *conditional bisector.* This operator yields a result that could be computed from a skeleton of the original image that is endowed with values of distance from that skeletal member to the border of the original object, more correctly called a *quench function* value, in tribute to the term given it in 1965 by Calabi, [47, p. 377]. Sometimes one is interested in the *rate* at which the quench function, defined along some skeletal structure and zero elsewhere, changes value. In $\mathcal{R}^2$, this could be thought of as the one dimensional derivative of the quench function,

in curvilinear coordinates along the backbone of some object's skeleton. In $\mathcal{Z}^2$, the digital conditional bisector makes it possible to calculate this measure without resorting to a study of a *run-length* encoding of the thinned line containing the quench function values. This bisector is a global operation that provides meaningful results as might be obtained through raster to vector conversion but, by virtue of its global nature, can be implemented in parallel over the entire image.

The description of the conditional bisector S^θ that follows draws expressions from [47, p. 390, 411–412]

$$S^\theta(X) = \bigcup_{n\geq 0}[(X \ominus nH) \setminus (X \ominus nH)_H \oplus (\theta - 1)H] \tag{5.27}$$

The expression is defined for $\theta \in \{\mathcal{I} \geq 1\}$, the set of integers greater than zero. The structuring element H is used to represent the six nearest neighbors of the origin in a hexagonal grid. The rectangular grid equivalent of H is the nearest eight neighbors. Mnemonically, this competes with the interpretation that H stands for *H*exagonal, or another culturally-dependent view that H is the eighth letter of the English alphabet. For instance, the structuring element H is a closed set. That is, $H \in \mathcal{F}$, where $\mathcal{F}$ is the symbol for the family of closed sets, mnemonically sensible more when one remembers it like a *fermée* door than a closed door.

The set difference groups operations in a broader sense than the erosion operators. Thus, in the relation given in relation 5.27, the quantity in brackets is the set difference between $(X \ominus nH)$, and $(X \ominus nH)_H$ dilated an amount corresponding to a radial distance θ. The digital conditional bisector is more commonly used with small values of θ. A specific example of S^2 is simply [47, p.411]:

$$S^2(X) = \bigcup_{n\geq 0}[(X \ominus nH) \setminus (X \ominus nH)_H \oplus \{H\}] \tag{5.28}$$

Digitally, this is the set difference between an erosion of objects in X by some number of steps, the nth iteration in a sequential or conditional sequential erosion or ablation, and the same result opened by H *and* dilated by $\theta - 1H$. The opening will eliminate objects in the intermediate, sequentially ablated result, that are less

in size than the elementary neighborhood H. In other words, if the erosion step is stripping off outermost pels at every iteration, then one more cycle of such an action would eliminate the object completely! The general conditional bisector uses the set difference to find elements such that when opened and dilated $\theta - 1$ times, can not contain the unopened eroded result. Then the difference between these two competing processes is given by $S^{\theta}(X)$.

To those readers who have an intuitive sense of how erosions should work, it may appear contradictory that one is taking the set difference between an erosion, and an erosion followed by an opening that is then dilated. The opening can be used in thinning to test those parts of objects that would be eliminated by the next iteration of erosion. The conditional bisector test above,S^2, is used to estimate whether some part of an image would, in the next erosion, erode further, often along a narrow, angular part of an object, in one iteration, than is possible to restore using $\theta - 1$ cycles of dilation at that point in the sequence, where $\theta = 2$. Plus, the conditional bisector can identify those areas of an image that are the centers of convex particles overlapping in some portion of an image [47, p.413]. That is, the operation above, which could be implemented in parallel because it is a global image operation, is able to identify those parts of the image that will, in a thinned result, have runs of equal quench function value for some length θ in curvilinear coordinates along the arcs of nonzero quench function values in a thinned image.

But because this conditional bisector does not tie itself to operations on the thinned result that are derived from raster to vector encoding, it can be useful in other applications as an intermediate processing step in what is desired to be an array processor or parallel algorithm. For an imaging problem that might involve the reconstruction of overlapping convex regions, an example might be an observation of the surface of a sand and clay mixture which has been pressed at some significant confining pressure, say 200 bar, to bring the grains in contact and extrude wet clay through the pore space [29], then dried and broken, viewed on a petrographic

microscope under diffuse lighting. The conditional bisector can be used to identify convex objects like grains which are overlapping, marking centroids of convex objects implied by the object boundary with a mark wherever the opening's effects can not be recovered by dilation with $\{\theta H\}$. For instance,

$$S^3(X) = \bigcup_{n=0}^{\text{maxwidth}} (X \ominus nH) \setminus [(X \ominus nH)_H \oplus 2H] \tag{5.29}$$

is an expression which would define objects in the center of features that are elongated, of size such that an inscribed circle of radius n would just fit inside the boundary, and sufficiently angular so that three elements in a row would have quench function values n at that point along the thinned result. In particular, two convex disks overlapping to make an image object would produce two isolated spots within the object following application of conditional bisector S^1. The conditional bisector, in addition to identifying run-length regions in the quench functions of thinned results by a global operation in parallel on the original objects, is, in a manner of speaking, a morphological tool for shooting holes in digital objects. Depending on their convexity, objects differenced with the conditional bisector of themselves will have one or more holes specified by this set difference. In this way, the conditional bisector provides digital access to an object's *convexity number* [47, p.143], a number which is distinct from, and more continuous under morphological changes of the object than the object's *connectivity number*.

In these sequential morphological operations, ones which are at the level of morphological words, if the structuring elements or letters are of the most elementary size, erosion steps will be removing one pel wide borders, or the ablation steps will be removing one pel wide border segments. The algorithm will run more rapidly, of course, if the structuring elements are simply larger, removing two or more pels from the outside or some outside direction of the object with each iteration. This reduces the fineness of details that remain in the transformed results, but it does so without changing the structure of the morphological words. By varying the structuring elements alone, it is possible to find tradeoffs between speed and accuracy

that control the algorithm's action.

Another one of the important uses of mathematical morphology is in the identification of points, even in thinned networks, with special connectivity properties. In a thinned result, where nonzero values of the quench function exist only along thin lines in the image, a characterization of the network requires one to identify the positions of *nodes*, that is, points where three or four lines, in a rectangularly gridded image, merge at a single point. A structuring set E used in the following, adapted from [47, p.412]:

$$Y = \bigcup_{j=0}^{7} [X \circledast F(j)] \cup [X \circledast F'(j)] \cup [X \circledast F''(j)] \tag{5.30}$$

could identify all the three and four-way intersections of lines in a thinned result with the following templates for the "four-way street" structuring set $\{F, F'\}$:

0	1	0	1	0	0	0	1	0
1	1	1	0	1	1	1	1	1
0	1	0	1	0	0	0	0	0
F(0)			**F'(0)**			**F''(0)**		

The structuring set $\{F, F', F''\}$ accommodates cases of near neighbor local configurations where three $\{F', F''\}$ or four $\{F\}$ lines come together to define a node, this is also described as a map of the local configuration of the object [47, p.396][48]. In fact, when operating on nearest neighbor mapped, or neighbor coded data, $NC(X)$, the sequential union over index j does *not* necessarily imply a series of sequential operations. In fact, all twenty four states implied by the union of eight-rotations of the three structuring letters can be consolidated into a look-up table filter that can be applied in parallel on $NC(X)$ by a raster engine. This important consideration is not mentioned by Serra or Rosenfeld, but it is a central concept in by application of fast Montoto thinning.

While investigators in the field of mathematical morphology may revel in or revile the new symbolism from the Fontainebleau school, some simply describe to their international colleagues image transformations in terms of basic set theory.

Questions will surely be raised as to why *we* in the earth sciences, having successfully described tremendously complicated seismic processing sequences, from all manner of seismic migration to three dimensional Fourier transformations, seldom having resorted to special-purpose symbolism for our processing, should now have to familiarize ourselves with the gamut of Fontainebleau symbols that appear as complicated at first sight as once did integral calculus equations. Why should we bother, and how is it different to process an image morphologically than to process one linearly, with the tools of two dimensional signal processing, or the tools of time series analysis adapted to two dimensions?

To these questions I can reply in the context of my background. First, I am not a member of the Fontainebleau school, and my experience has led me from refraction seismic data analysis and gravity surveys through studies of reflection seismic processing to traditional linear image processing and adaptive filtering. Now I have become involved in the problems of extracting meaning from the image, as opposed to processing it and leaving the graphical display of the result to be interpreted by a *seismic stratigrapher.* I have found subtleties and significantly inhomogeneous types of needs in the problem of feature recognition within images which make the classical models, operators, and even programming languages less useful with respect to these needs than the nonlinear operators possible in morphological filtering, however intractable their analytical expression may be.

The literature of machine vision, for example the journal *Computer Vision, Graphics, and Image Processing* does not overlap much with journals like *Geophysics.* I believe it is a beautiful thing to see the symmetry inherent in two dimensional Fourier analysis and its inverse. But as one who analyzes geological imagery I am often faced with the need to study decidedly nonperiodic objects in images. I feel the symmetry of the approach taken by the fast Fourier transform is likewise beautiful. However, even the simplest morphological operations involve something different, at the most fundamental level, than any linear filtering of time series anal-

ysis or its two dimensional adaptations. But for all the fundamental differences in the concepts that underlie the operation, there are some similarities between the implementations of these algorithms and those of two dimensional signal processing.

When I read a work like Serra's [47], presenting a terminology, or symbolic language seemingly all its own, and a language which is relevant to the work I am involved in, I feel regretful, not quite remorseful, to have not learned what were the fundamental differences between morphology and the signal processing approaches to image analysis that I have been taught at the graduate level in the mid 1980's. I wish I had been exposed, as a geophysicist, to morphological image processing earlier in my studies. I think it would have been appropriate to study mathematical morphology immediately after a second year of calculus. It would have been a much more pleasant introduction to a field that seems to have a broad range of application in geophysical data processing, yet seems to be relatively underexploited, left to those fanatics involved with "artificial intelligence" when it appears now to have important applications to seismic data just as in petrographic data. Morphology seems, from Serra's presentation, to be very thoroughly developed by large number of workers in Europe, and yet at the same time, from the literature in North America, to be not too widely understood by many in geophysical research.

Part of the problem with the Fontainebleau symbolism for mathematical morphology is that it tries so hard to represent morphological operations in a way so independent that it avoids all contact with the familiar, overlapping areas between two dimensional signal processing and morphological image processing. Because of this, I believe, most geophysical researchers with experience in seismic data analysis and reduction are disenfranchised, feeling that there is no bridge between their work and the work of the Fontainebleau school as presented by Serra [47]. Even more, the world of morphological activity is seldom seen to dirty itself with the (boring?) details of feature extraction, which is a crucial step in bringing morphological tools into application. This has historically been left up to the providers of image analysis

hardware.

5.4.2 A Convolution-like Process

In this regard, then, mathematical morphology needs to be seen in the familiar light of convolution processing of data—seismic data, for instance. When one does a two dimensional convolution on an array of numbers, any given element in the output or result array is the sum of a weighted combination of two or more neighboring elements in the input array, including the original element in its own neighborhood. For instance, a Laplacian edge-detecting convolution performs a very simple operation of taking the next element to the North, multiplying it by -1, the next element to the East, multiplying it by -1, to the South and to the West as well, and the center, the element that is to be replaced in the output, is multiplied by 4, then all five of these are added together and used instead of the center element in the output array. It is fairly tedious going through a large array, accessing all this data. In this case, for *every* element written to the output image, five data accesses, five multiplications, and an accumulation of their sum are required. Not only is this descriptive of convolution, this particular convolution *kernel* or set of coefficients is one which performs a *finite difference* operation on the input array. Variations of this very same convolution procedure, the Laplacian operator, are the basic units of a finite difference seismic data migration process. A description of this process can be done using classical terminology for elements $a_{i,j}$ of a two dimensional array A as:

$$\nabla^2[A(i,j)] \doteq 4a_{i,j} - a_{i,j-1} - a_{i-1,j} - a_{i,j+1} - a_{i+1,j} \tag{5.31}$$

To eliminate some redundancy of the classical expression, one might summarize the convolution process with a *kernel diagram*, this is simply a small matrix of the weights used at each position relative to the origin, that element which is replaced by the linear combination of neighbors in the convolved result. This is just like the templates of structuring elements used in morphological work, except the

members of a convolution kernel are often signed numbers, the coefficients of the linear combination that is the convolution—while structuring elements are often symbols describing conditions imposed on logical tests, of either an object or of its complement.

In template or kernel form, then the Lapacian convolution can be shown as:

$$\begin{array}{ccc} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{array}$$

This kernel describes the Lapacian difference operator for a 3×3 neighborhood. In fact, at this level, the basic morphological letters or structuring elements may be represented in a very similar way.

It became apparent some time ago that these types of convolution are well suited to solution by special purpose computing hardware which accesses the data more or less simultaneously throughout the input array, repetitively performing global operations on all elements of an array at high speed. With special parallel arrays of multipliers and summers, these *array processors* could perform the coefficient multiplication more or less simultaneously, summing results in an accumulator that contained the output of the convolution for one array element.

Similarly, the local configuration or neighborhood templates, can be accumulated into an output image in such a way that the structuring elements of mathematical morphology can be applied in parallel, by similar types of array processors, much like the way array processors compute certain convolution kernel. This is not just a coincidence. The significant difference between morphological and convolutional processing at this juncture is that instead of taking arithmetic operations such as a linear combination of neighboring elements, most morphological processes perform *logical* operations which relate one or several neighbor elements and put the result of these tests into the output array. For example, if we allow a 1 to symbolize a nonzero element of the array, 0 to symbolize a zero-valued element, and an × to symbolize not caring, that is to symbolize the same thing that a 0 did in a convolution kernel—

there only nonzero coefficients contribute to the linear combination, here only non-$\times$ members of the template impose a logical test. Then, a structuring element such as:

$$\begin{array}{ccc} 0 & 0 & 0 \\ \times & 1 & \times \\ 1 & 1 & 1 \end{array}$$

$$\mathcal{L}$$

Would have the following more classical statement implied by it:

$$\mathcal{L}[A(i,j)] = \begin{cases} 1 & \forall a_{i-1,j-1} = 0 \wedge a_{i,j-1} = 0 \wedge a_{i+1,j-1} = 0 \wedge a_{i,j} > 0 \\ & \wedge a_{i-1,j+1} > 0 \wedge a_{i,j+1} > 0 \wedge a_{i+1,j+1} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.32}$$

In this way, the need for forming more concise descriptions of basic processes in mathematical morphology becomes apparent. What is more, a series of structuring elements specified as local configurations of values $\{1, 0, \times\}$, given in template form, can express in a succinct way a fairly lengthy set of linked logical tests.

The templates, such as the one described in the previous paragraph, are not always required to be strict tests. Sometimes it is desirable to use the templates for counting purposes, using not the strict AND combination of all seven tests above, but requiring instead that, say, five or more of the tests be satisfied. Some hit-or-miss approach to this could be specified as well, as that all three of the $= 0$ conditions must be met, but only two or more of the $= 1$ conditions then need be satisfied. This type of constraint leads to the general morphological process known as *region growing*. With it, constraints can be specified for a structuring element containing, say, five 1's and some $\times$'s such that elements with more than, two of the conditions satisfied can remain as part of the object, and those with four or more conditions met can be added to the object if they are not already in it.

Region growing structuring elements defined with the two inequalities on the number of tests satisfied, specify a global image operation that can perform some erosion and some dilation both in a single pass. This is not possible with a fixed structuring element in a hit-or-miss transformation. With this soft type of con-

straint, a set of templates are given for each of possible conditions. But again, the use of many templates to describe a particular morphological test does not neccesarily require that a series of tests be performed, one after the other in the actual algorithm. Rather, with a parallel processing machine and data which is mapped into a local configuration space, or nearest neighbor mapped $NC(X)$, a raster array can be tested for a series of templates in a single pass through a look-up table. These type of considerations are important when implementing any type of process, morphological or linear, on an array processor. Ranges of tests, or soft conditions that are expressed in region growing, here, imply that in general, erosion or dilation can be described as special cases of a region growing process [45]. Likewise, for hard conditions, erosion and dilation can be described as special cases of the hit-or-miss transformation. Since thinning has been described above as the difference between the original set and the hit-or-miss transformation of that set, these details, and any new ways of expanding the class of tests performed by the hit-or-miss transform and its variations can become important in applications of morphological thinning.

The local configuration expression, or neighbor coding, is a type of image transformation that is a hybrid between linear and morphological processes. It is a fundamental tool in array processor implementations of morphological processes. In many situations, it is possible to perform a hit-or-miss transformation or any other morphological processes more rapidly on NC transformed data than on the original in the same way that convolutional types of signal processing can be performed more rapidly by multiplication on the Fourier transform of the original array. For morphology, the parallel technique is that morphological transforms based on local configurations, those with compact templates, can be expressed as a look-up table configuration operating on $NC(X)$ where otherwise they must result from serial operations on X. In practice, the local configuration of a binary image is much easier to describe than the two dimensional Fourier transformation. What Sobel [48] calls

the neighbor coding, $NC(A)$ is:

$$NC[A(i,j)] = \sum_{n=0}^{7} 2^n V_n \quad \text{or} \tag{5.33}$$

$$\begin{aligned} NC[A(i,j)] = \sum & 2^0 a_{i+1,j} + 2^1 a_{i+1,j-1} + 2^2 a_{i,j-1} \\ & + 2^3 a_{i-1,j-1} + 2^4 a_{i-1,j} + 2^5 a_{i-1,j+1} \\ & + 2^6 a_{i,j+1} + 2^7 a_{i+1,j+1} \end{aligned}$$

Where the orientation of the spatial to numerical relation is given by the template for V_n as follows:

$$\begin{array}{ccc} 3 & 2 & 1 \\ 4 & \times & 0 \\ 5 & 6 & 7 \end{array}$$

V_n

In this way, the logical information resulting from some binary result from a test of the eight neighbors in the rectangular grid's local configuration are summarized in an eight bit gray level transformed image with each direction's neighbor information residing in a particular bit plane. For this, bit-slice computers, which have the features of array processors particularly suited to raster image processing, are very well suited to fast computation of $NC(A)$ on large arrays. If two logical tests are required, as in the two parts of the structuring set T of the hit-or-miss transform, one testing object, T_1, and the other testing its complement, T_2, or templates from the $V_n > 0$ and $V_n = 0$ tests mentioned above, then the pair of transformed results $\{NC(A_1), NC(A_2)\}$ can be derived from a binary or gray level set X if $A_1 = \{a : x = 0\}$ and $A_2 = \{a' : x > 0\}$.

5.4.3 Obfuscation That May Help

In keeping with the Fontainebleau symbolic description of mathematical morphology, the hybrid neighbor coding described above is too linear in its approach to

deserve a morphological symbol, though I might try to make one for consistency. Let the $NC(A)$ refer to the original set X in relation to conditional test A with the symbol in $NC(A) = X \# A$ where testing set $A = \{A_1, A_2\}$ and the subscripted A's refer to the combination of one equality and one inequality test, to set up the neighbor coding which summarizes all data needed by the hit-or-miss transformation in a parallel implementation. This arrangement uses as input one gray level image and produces as output two gray level images that are bit slice local configurations or neighbor codes. While this is not a pure morphological transformation, it is a useful hybrid between linear and morphological transforms that helps one to implement nearly any morphological expression as a simple look up table or pair of tables. Neighbor coding is also useful to those readers familiar with array processing techniques and who are interested in making use of morphological algorithms in a familiar context.

So what does the Fontainebleau symbolism offer? I think it is more than simply symbolic obfuscation. It helps one to relate morphological image to more familiar processing concepts. Two dimensional signal processing can make do with expressions from calculus in the continuous domain and FORTRAN-like statements in the gridded one, but workers have used shorthand notation, such as $\circledast$ and $\oplus$ for cross correlation and convolution [10]. Morphological processing needs similarly compact ways of describing its basic operations. The Fontainebleau symbolism provides this. The symbols permit one to manipulate the procedures to gain insight and understanding, to explore changes in complex processing sequences that identify possible shortcuts, and provide one with an internationally understandable way to describe morphological processing at all levels, from structuring element to processing sequence. On the other hand, while symbolic expressions purport to convey a pure and rigorous sense of the concepts being symbolized, an unfamiliar symbolism exacts a steep toll on the reader.

Morphological symbolism is not familiar to those with accepted scientific or

engineering backgrounds in mathematics, most earth scientists in particular. So while the Fontainebleau symbolism provides a numerically flavored intuition, as for an engineer about to make use of an image analyzer, the symbolism itself will not guarantee that one can understand the processing sequence in sufficient detail so as to be able to implement it as a code. Likewise, in gaining a freedom from adaptation to a particular implementation, procedure, or language, the symbolism may leave unrepresented some important insights into how one actually goes about performing these operations. The best way to symbolize both morphological theory and practice remains to be seen. Certainly, the Fontainebleau symbolism fills a need at one extreme in the description of morphological processes and concepts. There is always an abundance of words to fill the other extreme. How to describe mathematical morphology in a concise yet understandable way that includes information not only on the continuous-space concept being mimicked but also how to rearrange the sequence and implement it in a parallel fashion on discretized space as well—this is the middle way in morphological communication.

5.5 Discussions Of Morphology

The three preceding sections have each presented different approaches to the description of mathematical morphology and morphological processing concepts. The first was to describe an algorithm in terms of the simplest set theory equations or relations. The second was to describe processing sequences in structured lines of text. The third was to use the Fontainebleau symbol set to describe all fundamental morphological operations uniquely and succinctly, also creating a need to define permitted uses of combinations of these symbols.

In this section I present six brief examples of how these various techniques may apply to aspects of image processing that are relevant to the concerns of thinning and that surround my approach to rock property estimation from images. The first example describes in text some of the differences between the approach I am developing and the work of the University of South Carolina Petrographic Image Analysis group. The second example describes, more with Fontainebleau symbols, the difference between what has been popularized in petrographic image analysis as erosion and what is ablation. The third section describes the difference, again with Fontainebleau symbols, the difference between thinning and skeletonization, also between thinning and the skeletonization to zones of influence , or skiz [47, p. 385–386, 411]. The fourth section describes, with combined text and symbols, one of the important differences between petrographic image analysis and the wider set of morphological tools when characterizing pore space images for connectivity properties. The fifth section describes in text and symbols how "invertible" is thinning, and presents some symbolic description for how one can recover convex hulls of original pores from thinned results. This section presents a proposition in text that combinations of these concepts, specifically boundary encoding of objects followed by thinning and curve encoding of thinned results, can form a better analytical procedure by which to characterize pore space images than the techniques that have been published describing petrographic image analysis—for a variety of reasons.

5.5.1 FMT Is Not PIA Repackaged

Differences between mathematical morphology and petrographic image analysis have motivated my approach to pore space characterization by means of thinning rather than successive erosion and boundary coding. In my search of literature in 1986, I found a number of papers on petrographic image analysis [3, 32, 20, 16, 17, 19, 18, 26, 25, 33, 52, 54] describing in some detail the Univ. of South Carolina group's approach to characterizing the texture of pore space images, using the concepts of erosion and dilation which were new to me at that time.

Later in my search I found a paper by Bel-Lan and Montoto [2] that described a thinning transformation. Their transformation skeletonized objects while preserving a quench function as well. They applied this approach to micro-fractography, studying fractures in Parisian granite statues. Since the fractures in their study were quantified by connectivity with other fractures visible in the image, their approach seemed to be a more straightforward way to approach the connectivity of pores than PIA's regression of textural attributes measured with petrographic image is against bulk permeability. One of the objections which seemed more relevant to PIA was expressed by Alec Desbarats at Stanford, namely that the textural approach used in PIA could not distinguish between a natural image of pores and one that was simulated in a texturally conditioned maximum entropy realization. In the stochastic simulation, the pores are randomly jumbled throughout the image and their connectivity is destroyed. But since they have the same types of shapes as the natural pores whose textural statistics conditioned them, they look the same.

When I realized that PIA's numerical approach to connectivity could be fooled this way, a way that should profoundly affect the simulated pores' permeability, I began to pursue the morphological thinning approach to connectivity measurement. Since network model studies of electrical analogs to pore space [55] acknowledge that series of interconnected pore chambers and throats are needed to describe the fluid flow properties of many rock pore systems, a maximum entropy juxtaposition of pore

elements in a model of fluid flow should disrupt the permeability that is estimated from that model—even if that model is an image. If the permeability of the system is controlled by the smallest diameter throats between pores, and those throats are well represented in the thin section image, then this argument is weakened. However, the point remains that there are morphological transformations developed specifically for the evaluation of object connectivity in an image and these have not been mentioned as components in the published PIA approach to permeability estimates from thin section data.

At this time it appears to me that there are two advantages to be realized from a use of thinning as a connectivity criterion in morphology. They are first, that the image analysis technique used on thin sections would also have relevance to studies at broader scales, of wider ranges of flow problems in the earth sciences than only the thin section scale, namely reservoir models. This is because with thinning, connectivity may be calculated deterministically from the image, rather than inferred from a combination of textural statistics. Secondly, as a numerical constraint on the physical likelihood for existence of pore spaces which are realizations of a stochastic simulation process, thinning can be calculated to improve the similarity of simulated pore spaces to natural ones. This can be used by conditional simulations as a deterministic measure of connectivity. By concentrating on a more universal measure of flow connectivity, I work on providing a tool that is relevant beyond petrographic images, to all scales of geological information.

5.5.2 Erosion Vs. Ablation

When performing erosion, perhaps the simplest of morphological processes, it is common to use a convex structuring element, symmetrical about its origin $\{o\}$. Thus, for instance, a 3×3 square, or a structuring element that requires all eight nearest neighbors in a rectangular gridded system to be members of the object, can be used to strip off the outermost layer of pels from all sides of all objects. Holes

within objects would also be surrounded by outermost layers in this definition. In this way, a single fixed structuring element is all that is needed for even sequential erosion where, at every iteration, objects will be reduced in width and height by two pels. In thinning for *flow connectivity* applications, it is usually desirable to retain a *quench function* that records variations in width of objects. The quench function values should be able to record width to single-pel size precision, to the accuracy of the maximal block they represent [45, p.211] The structuring sets used in the hit-or-miss transformations at every iteration of sequential thinning are often oriented to perform single pel width erosion from only certain directions, a directionally selective erosion I call *ablation.* Ablation, if repeated always from the same direction, may not preserve the *homotopy* or topology-invariant properties of the object in the thinned result. To thin homotopically as well as leave quench function values along more nearly *medial axis* lines of the object, ablation is performed from a variety of directions. In this way thinning can compute the medial axis position and quench function in a single morphological operation of conditional sequential thinning. My first explicit introduction to this concept was through Bel-Lan and Montoto [2] and I have never seen mention of this approach in PIA literature, despite the fact that is the logical way to approach any calculation of pore *throat size distributions* in an image using morphological tools.

As an exercise, consider that thinning by repeated ablation from one side would shift the thinned results to a residue along object edges opposite the ablation's "incident radiation" direction. The thinned lines would form an *umbra* around holes in the manner of [47, p.426,Fig.XXI.1], which is drawn there to illustrate the class of objects that can be studied in gray tone morphology. The quench function values would still vary, but would measure the *projected* thickness of the object for rays arriving from the direction of ablation. In this and other ways, thinning is a morphological tool that provides a far richer set of means by which to analyze pores than does repeatedly eroding and boundary coding them.

Sequential thinning can be described as producing a quench function Y from objects in a binary image X in the following symbolic way:

$$Y = X \bigcirc \{L^i\} \qquad \text{where } i \text{ is used to step through ablation directions} \qquad (5.34)$$

The sequence $\{L^i\}$ may also step through members of the structuring set that alternate between ablation in the cardinal directions and the diagonal directions. In detail, this would produce strange medial lines that were 4-connected in some places and 8-connected in others, disconnecting the medial line at the point where the connectivity rules change [45, p.239]. Since the rectangular grid is not as symmetrical with rotation as the hexagonal, choices exist in defining diagonal ablation structuring set members [47, p.419]. However the choice is made, in sequential thinning the members of the structuring set are each a letter in the morphological word that defines the thinning. If the structuring set is not changed except by rotation, the superscript notation is dropped and $Y = X \bigcirc \{L\}$ suffices to describe the sequential thinning, with a changing ablation direction from step to step in the sequence. The same could be done with erosion, but less selectively if it is isotropic erosion, or the very same way if it is directional.

5.5.3 Thinning Vs. The Skiz

The difference between thinning and skeletonization can be more than simply that of one retaining a quench function value and the other not. The skeleton is described very elegantly in [47, p.375] for Euclidean space $\mathcal{R}^2$. Intuitively, it makes great sense to have the skeleton represent the origin of the maximally inscribed circle within a closed object that has a curved boundary. But in lattice space, the maximally inscribed circle has the two problems of being always rough and being able to exist at only a few discrete sizes. Yet it is still possible to approach digital skeletons when the problem is redefined in terms of conditional sequential thinning [47, pp 389–390] and maximally inscribed blocks [45, p. 210]. Namely, conditional tests that save those features that define an object's homotopy and would be eliminated in another

cycle of thinning can define sequential homotopic thinning. This has been described as:

$$s_n(X) = (X \ominus nH) \setminus (X \ominus nH)_H \qquad \text{with the quench function} S(X) \quad (5.35)$$

$$S(X) = \bigcup_{n}^{\text{maxwidth}} s_n(X) \quad (5.36)$$

But this simple thinning approach is known to leave branches or "bones" as Serra calls them leading out to corners of squares, for instance. The culprit, so to speak, is that this approach uses the *conditional bisector* S^1 with its isotropic structuring element $\{H\}$, rather than conditional ablation with $\{L\}$ at every iteration n of the sequential thinning. But the expression is useful in the way it provides insight into the calculation of quench function values. If it is possible to avoid leaving these branches, while preserving the homotopy of the object, then a thinning algorithm could avoid producing certain common artifacts and save the trouble of a post-processing step known as *pruning* to remove superfluous branches.

While fast Montoto thinning, for the most part, avoids leaving these branches in its thinned result, they sometimes still form from single pel protrusions on an otherwise smooth object surface. The original algorithm of Bel-Lan and Montoto solved this problem with a nine scan per iteration approach, on a general purpose machine. FMT does not use the 1×4 neighborhood information of the nine scan algorithm, however, and is a modification of their five scan per iteration approach.

The pruning procedure removes these ends by, of all things, sequential erosion of the thinned result using structuring set $\{E\}$ that identifies the ends of thinned lines and nibbles away at them. In a skeleton where branches that, like railroad spurs, lead nowhere in the sense of network connectivity have been pruned down to nothing, one has produced a skeleton of influence zones or skiz. In a multiply connected object which is skiz'ed, the prunings have the specific goal of removing those spurs which project into closed loops [47, p.406]. If the term skiz evokes a sense of the term "schizo-", it can be viewed constructively in the sense of splitting or finding cleavage between objects in an image, and defining those areas of the

image that are closer to those objects than to any other object in the image, that is to say the skiz defines a cleaving of the image into objects' zones of influence. The digital skiz is discussed in [47, p.397].

If a structuring set $\{E\}$ identifies the ends of lines which terminate in the eight directions possible with rectangular gridding, then a skiz could be defined with:

$$Y(X) = (X \bigcirc \{L\}) \ominus \{E\} \tag{5.37}$$

So that an image X is first sequentially thinned with structuring set L and then pruned with E to eliminate the stray branches in the thinned result. Since the structuring set E will not attack the middle parts of thinned lines, it is possible, once only the main structure or multiply connected structure remains, to perform conditional dilation with the complement of E which only dilates the endpoints of lines. This could be expressed as $Sz(X) = Y \oplus \{E\}; X_0$ where the dilation of the pruned skeleton is conditional to matching those parts of the thinned set $X_0 = X \bigcirc \{L\}$. Since the stray branches will have been pruned back to the main part of the skeleton, the only endpoints that remain to be dilated with structuring set E are the endpoints on the main skeleton. Since ideally the main skeleton wouldn't have been pruned in the first place, this recovers that seemingly unavoidable step. This is the traditional approach to getting skiz $Sz(X)$.

It has been recognized by Doyen [14] that one of the more important features provided by Petrographic Image Analysis at the University of South Carolina, PIA, is that of pore throat size distributions which result from the *sieving* process, a textural separation of objects in an image into distributions of size defined by a particular metric. Sieving can also find pore throats of a particular size by identifying nonconvex objects that become broken by successive erosions. This is the pragmatic approach taken Full, et al. and by Ehrlich [25, 19] which has produced meaningful results for several years in north America.

When drawing from the wider set of morphological tools available in research from outside the U.S., it is possible to recover pore throat size distributions in a more

unified way by thinning the pore indicator image to a quench function image. This would be the case for elongated pores even if they did not connect to one another throughout the image. And so in this way the one dimensional size distribution histograms recovered from the image in PIA could be produced from a vectorization of the quench function image very readily. This is the result of applying Cederberg-Sobel compilation to FMT output. Further, many properties concerned with the spatial disposition of objects within the image could also be derived from the same quench function image. This type of information has been lacking in published PIA results, but is available using the tools provided in this report.

5.5.4 Thinning's Lost Information

What is lost when representing an object by its thinned result, the skeleton and quench function alone? If the thinning that has been performed leaves a homotopic result, one that is same-topologied as the original object, then the thinned result has preserved the connectivity of the original object. Despite this feature, the roughness of the object's original boundary may be lost. This is even more the case for objects reconstructed from the skiz and quench function than from a skeleton and quench function with every branch remaining. An inversion from the skeleton and quench function toward the original object is the union of dilations by many different sized structuring elements H^n where the size of H^n depends on the value of the quench function. The larger the size, the fatter the original object at that point. In digital practice, since thinning is computed and not a skeleton, so thickening is applied and not the inversion of skeletons. In practice, a sequential dilation from a specific direction, what I call rhiming, is implemented as a form of the hit-or-miss transformation, and used to dilate by what is effectively a larger square structuring element at later points in the sequence by compounding the effects of sequential rhiming. The exact structuring element is given in Figure 5.3.

Without becoming stuck on semantics, consider that successive dilations by a

simple structuring element will make objects bigger. In fact, there should be some way to do the work of, say, five successive dilations in a single pass. What is the relationship between this single compounded structuring element and the simple ones from which it was built? A simple answer has been found [47, eq.II-15,p.47]

$$(X \oplus B) \oplus B' = X \oplus (B \oplus B') \tag{5.38}$$

And so sequential thickening can synthesize recursively the largest structuring elements needed in recovering width from a quench function, by effectively dilating the structuring element over and over to produce the effects of a single dilation with a large structuring element. But what original details are lost in doing this? It is difficult to recover details along rough boundaries that are at larger distances from the quench function in the thinned image. This is because these larger distances are recovered through dilation by structuring elements that are the result of many sequential dilations, which is to say they are smoothed. This question is important from the standpoint of demonstrating how much information is contained in the thinned result's quench function, in practice, workers analyzing connectivity with thinning would not need to do this because the original set still exists, if only as a copy of the input to the thinning operation. But the question of how well one can recover the original from the thinned result is a fairly important one.

My answer uses a notion from morphology that is called the *convex hull* of an object. In continuous space, it is the Saran Wrap-like covering over a non-convex object. In digital space it can be calculated too, even iteratively. Without proof, what I claim is that quench functions may recover the original to within the differences between the original and a copy of itself that has been operated on by a sequential convex hull process for a small number of iterations [47, p.392]. In other words, it has been shown that a thickened quench function can recover the original object, with some filling-in of embayments or concavities in the object's border, that is, it can recover its convex hull [47, pp 400–401]. This would also smooth over rough borders—one can expect to recover a smoother representation of the object's

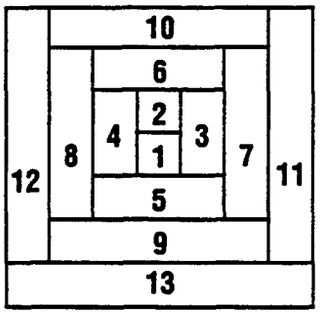


Figure 5.3: The structuring element used in fast Montoto thickening. A quench function value is dilated for all parts of the square less than or equal to its value. The size of one pel is shown in the center box labeled "1".

borders when recovering it by successive thickening of the quench function.

The companion thickening algorithm uses a square structuring element shown below.

However, from the standpoint of characterizing rock pores without a loss of information, the paragraph above suggests that thinning is not a substitute for all texture analysis, just some of it. In terms of summarizing a pore space image, however, one prefers not to keep the original image in raster form because of the size of the data. So raster to vector conversion or *boundary coding* should be used to supplement the thinning calculation, particularly when surface roughness, a good perimeter/area ratio, or the exact total pore area are needed for coefficients in a regression equation. Still, a reasonably close total pore area should be recoverable with thickening of the quench function. For very high aspect ratio pores, a weighted sum of quench function values can suffice, and Bel-Lan and Montoto [2, p.44] report errors in the neighborhood of 5% with their thinning-2 approach. Their thinning-1 and my fast Montoto transforms both manage to come within 20% of the original feature area for very low aspect ratio pore features in Fontainebleau sandstone. A correction of this magnitude may need to be made when using an automatically derived network model from thinned results.

A useful supplementary morphological procedure is called particle marking [47,

p.409] or "blow up" by Riepe and Steller [43]. This is the dilation of an object, often from a single point, conditioned by its original outline. With it, some feature derived from the thinned result, such as thresholding, could be used to start from, say, all quench function values = 22, and then dilate from those points to blow up these values, like a balloon, until they fill the original object again. In this way, particles marked by certain properties can be recovered exactly, using the original object as a mask in dilation. In this case, only objects with features of width corresponding to quench function values = 22 would be recovered. Semantically, if one looks at the single pel that expands to fill the object, the process looks like blow up, while from the global perspective it appears that certain objects are simply marked, identified by their presence in the particle-marked image. Clearly, one can find many possible uses for this technique. It allows one to recover not just the whole image, but only those objects meeting some test, even based on analysis of the thinned result, to be recovered exactly from thinning with all their surface roughness. This is a typical example of nonlinear processing's unique characteristics.

However dissimilar their names, the particle marking procedure generally lessens the need for an accurately invertible thinning procedure. Symbolically, this conditional dilation is described by the following expression that details calculation of the skiz Y from a thinned result X', based on [47, p.411]:

$$X' = X \bigcirc \{E^n\} \qquad \text{the thinning of } X \tag{5.39}$$

$$Y = X' \cup [[\bigcup_j (X' \odot E_j)] \oplus \{H\}; X \setminus X'] \tag{5.40}$$

In this algorithm, all dilated points are first identified by a structuring set E, a morphological letter that means endpoint-finder template, that is rotated through the eight directions j in a rectangular gridded system before the dilation is applied. In the first thinning equation, describing the set X', the superscript n is used to describe the number of rotation cycles through which the thinning proceeds. The rotations of the same structuring element from one direction to another are implicit in the notation. In the second equation, the subscript j is used simply to

step through the various directions, so that all endpoints in the image, facing all grid directions are identified before conditional dilation by $\oplus\{H\} \cap (X \setminus X')$. The conditioning ensures that the dilated points will honor that part of the image that was pruned away. Since the structuring element H will expand each endpoint in all eight directions in parallel, to keep the dilation thin, conditioning to a thinned set is applied.

Likewise, a procedure that both marks and isolates endpoints would use the skiz Y as input and reapply the endpoint structuring set E with a hit-or-miss transform, dilate these identified endpoints, then set difference these regions from the skiz. The isolated endpoints, with a border around each, can then be returned to the trimmed skiz with an intersection.

$$Y' = \bigcup_j (Y \circledast E_j) \qquad \text{marked endpoints } Y' \tag{5.41}$$

$$Z = y' \bigcup [Y \setminus (Y' \oplus H)] \qquad \text{marked and isolated endpoints } Z \tag{5.42}$$

The example above, while slightly more specific than general particle marking, is an example of line marking. It is a global operation, and so can be used to mark thinned lines in an image in a parallel implementation. In fact, together with the conditional bisector, it makes a pair of global or parallel-implementable operations that yield some of the results that can also be derived from raster to vector conversion, without the conversion step. The procedure above is specifically used in the calculation of a true skiz. It works whenever the longest "stray" branch has a length less than n, and so is pruned back to its nub. Structuring set E does no damage where these branches join the main arc, but does nibble down its ends. If the image is cropped in such a way that at its edges, the main branch appears to be much shorter than a stray one connected very close to the end that appears in the image, this algorithm has problems. The longer stray branch will become misidentified as the main one in the skiz. Most likely a comparison of relative quench function values can resolve this problem, presuming the main branch's values are larger than those in the stray tributary one wishes to prune.

5.5.5 Boundary And Arc Coding

Considering the value of the pore throat size distributions in image analysis, and the unified single-pass solution of porel size and spatial disposition by sequential thinning to a quench function, I suggest the following approach as an alternative to that used in PIA for rock pore image analysis for the purpose of permeability estimation from thin section images. Fast Montoto thinning is not wholesale replacement for PIA, but I believe it refines the process now used, since it merges connectivity attributes now unmesured with bounday measurement.

After segmenting the desired porespace from a multispectral set of gray level images such as red, green, and blue for visible light, boundary encode the objects in this original image. The CSC algorithm can be used for this step. Boundary encoding is an important part of both PIA and this suggested procedure. Perform sequential homotopic thinning such as fast Montoto thinning on the image. Then, use a similar type of boundary coding to identify line segments, closed loops, and each point's associated quench function value in the thinned result. This is exactly what I designed CSC to do.

The results of this procedure are two raster to vector databases and two raster images, and the thinned raster image. From the original object boundary coding, precise measurements of total object area, object count, mean object area, area/perimeter by object, true max/min diameters, and other textural information can be derived. A list of possible attributes was given by Freeman in 1974 [23], many recognized as useful have been listed by Ehrlich [17].

Computationally, there can be an important advantage in keeping the global operations grouped in a processing sequence. That is, by avoiding the local step, or boundary coding of objects at every iteration, a parallel machine may perform sequential thinning faster if it does not need to wait for a host computer to circumnavigate each of its image's objects after every iteration. If one uses only the basic erosion and dilation operations, however, there are few options besides boundary

coding to extract useful information from the image. Considering the full complement of morphological operators, sequential homotopic thinning may let the parallel machine do its thing best, and the host computer do its thing best, without an algorithm synchronizing the two machines between every morphological iteration.

The approach I suggest summarizes the line segments explicitly as components of a network representation. Then, once boundary conditions are specified, the line segments may be replaced by their effective values and a compact linear system, with a mass balance equation for every node, or pel where one, three or four segments meet, and a pressure drop equation for every closed loop, or ordered sequences of line segments, can numerically represent the flow network as a circuit.

Not only does the use of sequential homotopic thinning provide a unified way of approaching the calculation of pore throat size distributions, it also generalizes the applicability of the image analysis techniques that are applied now mainly to thin sections. By characterizing numerically the way that objects in a rock image are connected to one another, that is, *how* they are spatially disposed throughout the image, a connectivity analysis technique may find application not only in a *particular* rock formation, but in wider ranges of thin sections as well as border classes of fluid flow problems and structural problems in the earth sciences.

5.5.6 Reprise: Morphological Description

This section has been the delineation of three different approaches to the description and discussion of morphological processing. In the many situations where I have found it desirable to use the Fontainebleau symbolism to represent concepts more succinctly than was possible in words, hopefully the introduction to these symbols provided by this section has been sufficient to familiarize the reader with the symbols used in the rest of this paper. After troubling to learn some of the Fontainebleau symbolism, my view of image analysis has shifted from the paradigm of PIA in the USA to some appreciation of Serra, the sans-culotte.

Here I turn from the theoretical concerns to that complementary part of morphology, the numerical aspects. That is, given a theoretical knowledge that one is better off performing sequential homotopic thinning than if one successively interlaces erosion with boundary encoding—how in the world do you actually do this: $Y = (X \bigcirc \{L\}) \cup \{M\}$, where the sequential homotopic thinned result Y is the result of repeated rotated applications of structuring set $\{L\}$, which I adapt from Luis Montoto, conditioned to include only members of the medial line set $\{M\}$, again based on the transform presented by Bel-Lan and Montoto [2]. There are other sequential homotopic thinning transformations in the world, but in my search of the literature, [2] was the first one that not only made sense, but convinced me that there was a way of advancing beyond current petrographic image analysis.

Sadly, as I turn from one semantically sinuous section, I face another. For all the symbolism of morphological theory, parallel processing hardware has a vocabulary all its own. By dragging the reader from one extreme to the other—from the grand vistas of morphological theory to the myopia of microcode, I feel that I incur an obligation to ease the transition by explaining jargon as it escapes from the keyboard.

In conclusion, the image processes of mathematical morphology are the *nonlinear* companions of two dimensional filtering operators, those that have traditionally relied on linear combinations of neighboring array elements to produce convolution, cross-correlation, the Fourier transformation, seismic migration, and adaptive linear filtering, among other distinguished accomplishments. Although it may be a struggle to acquaint oneself with the terminology of nonlinear image processing, the rewards to those who succeed can include a perspective that incorporates the best of both approaches. Some interesting (and simple) results of morphological filtering can yield results that are weakly approximated by filtering with linear operators.

Chapter 6

Numerical Morphology Techniques

The results of this section are less global and analytical than the previous section. However, they are just as necessary if one is to accomplish any real morphological measurements. The references that I make in this section are less to the morphological processing literature, with the exception of Bel-Lan and Montoto [2], than to equipment manuals and software manuals from a vision engine manufacturer. I introduce some terminology that is accepted in the computer graphics community, but may be unfamiliar to many involved with geophysical computation. I then describe the use of a particular class of morphological concepts, those suited to a lattice domain. Some of the groundwork for these concepts was given in the previous section. I then relate some of those concepts to the graphics or raster array domain—where the array processing hardware meets the morphological relations and presents a need for the modification of certain concepts in mathematical morphology.

6.1 Some Processing Terminology

The elegant expression of mathematical morphology with the Fontainebleau symbolism, as with linear algebra expressions in signal processing, becomes more interesting, almost alive, when actually implemented on machines. In the case of linear algebra, the traditional programming languages have served the needs well.

In mathematical morphology, the logical descriptions and their combinations have created a need for a different way of computing the results, much less expressing that process to a machine. Because the large sizes of arrays needed in higher resolution imaging challenge a machine's throughput, or ability to access the array, process it, then write the processed result pel by pel, conventional general purpose computers are not ideally suited to the task of morphological processing of larger arrays. As a consequence, when one moves to a better-adapted processing architecture, one often loses the convenience of traditional programming languages. This loss is because not only the compilers but the structures of the languages themselves are tied to both the paradigm and the small instruction set of general purpose computers.

Some new types of special purpose computing hardware may be thought of as performing many computations at once, in parallel. Some processing algorithms can be adapted from their expressions in classical programming languages to these special machines, but not all of them. Parallel processing algorithms are not always amenable to expression in a traditional sequential flowchart. Thus, one of the problems of those who engineer image analysis systems is to find a hardware architecture that makes morphological computations more rapid.. While one can write manuals to describe the way that this hardware is used, the programming problem has not been solved in a standard way. In searching for a way to convey the morphological concepts while working with a particular set of computational hardware, I have reached a compromise that symbolizes the algorithms by coding them in a parallel flowchart with a fixed structure. The structure of the flowchart is defined by the architecture of the parallel processing hardware. The operators in effect at the nodes of this multiply connected flowchart change from step to step within an algorithm. And so my approach to the presentation of morphological algorithms has developed into one with a sequence of flowcharts, each sharing the same structure, but combining different operators specified at different points in the flowchart. Intermediate results are passed from one chart to the next through either

image memory caches or with direct connections from one processor to the next in a multiple-processor vision engine. I will describe these details at the end of the section. First, for completeness, I will define my vocabulary more systematically.

In computer graphics as in television, the term *raster* is used to describe an image that has been coded, at one time or another, into a one-dimensional sequence or *time series*. Despite that coding, raster data are familiar two dimensional arrays. It so happens that any bounded two dimensional array can be represented by raster. In fact, two dimensional arrays in a computer are constructs of high level programming languages. At some level (one many programmers would prefer to avoid) the array must be stored in memory. There, it resides in a monotonically increasing address space with only a single index, which is to say it can be viewed as a one dimensional sequence. An acknowledgement by programmers that two dimensional arrays do not really exist as such can be one of the stumbling points in a transition from programming in FORTRAN or BASIC to programming in C or Assembler. In the context of image processing, raster is synonymous with a two dimensional array representation of a one dimensional vector.

6.1.1 Raster Arrays

In higher resolution imaging a raster array has come to mean interchangeably a *big* array or a *very* long vector. In the television signal digitizing system that I have used in my dissertation research, the integers in this big array are each 8-bit quantizations of the TV signal voltage. Voltage in a TV signal corresponds more or less linearly to the intensity of imaged radiation focused on a photoreceptor like a camera's TV tube, whether vidicon, Nuvicon, or Ultricon for visible light, or PbS, lead sulfide, for visible and infrared light, or even a CCD, charge-coupled device—a solid state discrete photoreceptor array.

Eight bit quantization means that one can reconstruct the TV signal from one of 256 discrete voltage levels. The analog signal amplitude varies from ± 0.7V [30,

p.574]. Eight bit quantization means that this signal is recorded, at every discrete sample, to a precision of 5.47×10^{-3} V. That is not too bad, but how often are the samples taken? My system digitizes eight bit digital samples of the analog video (TV) signal 7.39×10^6 times each second. Before you gasp, let me say it works best in 1/30th second bursts, but it can be coerced into running for up to 7/30 s. This is because the digitizer, the analog to digital signal converter, runs at 1×10^7Hertz, or samples per second. Why the discrepancy between rate and result? The answer to this goes back to the nature of a raster image. The TV signal is always working, trying as hard as its NTSC (National Television System Committee) standard allows, to send images to the screen. But the video signal is inherently a time series, or one dimensional array $\mathcal{R}^1$. The coordinated mapping of this time series onto a minimally distorted two dimensional array $\mathcal{R}^2$ does not come cheaply. In fact, over 25% of the signal's information is spent keeping this mapping from $\mathcal{R}^1$ into $\mathcal{R}^2$ from getting out of control in the TV receiver. In my case, that receiver is the analog to digital converter. The raster signal that appears on the screen is composed of 480 lines in North America and Japan. At 10 Megahertz sampling, a digitized TV signal has 512 pels along each horizontal scan line. Since it digitizes the time series, a video digitizer can't help but digitize 480 scan lines per frame—this is implicit in our TV standard. So my system's TV digitizer works at 512×480 gridded resolution. Because the eight bit discretization fits one pel into one byte of data (eight bits exactly—what a coincidence!) one raster image contains 480×512 bytes of data. Well, actually the first line in a digital TV signal is only the last half of a line and the last line is only the first half of one, so since the total image is truly 480×512 we must digitize 481 scan lines to cover these two half lines and so each digitized TV signal is exactly 512×481 or 246,272 bytes. This is almost 1/4 megabyte, which would be 2^{18} exactly.

When the conversion is made to or from the one-dimensional form, arrays in a TV-based graphics system will contain information from a sinistrodextral descending

scan (left to right, top to bottom). For symmetry, this downward-directed y-axis convention is often used when addressing members of the two dimensional or image array. It is logical to adhere to this convention since as the speed of computers increases, approaching the speed of raster display devices, the display device needs the top part of image data first, and scan lines in order. Data storage conventions that respect this need will work more harmoniously with graphical display hardware.

Upward-stored scan data is only piecewise continuous with respect to the raster scan, and requires a conversion process seldom implemented in vision engine hardware. The raster coordinate system places the origin in the upper left corner of the screen, with x increasing to the right and y increasing downward. In practice, however, graphics programs and graphics adapter or display hardware often place the coordinate origin in the lower left corner the screen since that is more natural for plotting of data. (What are graphics systems for, after all, if not for graphs?)

When considering the relations between the image array in its more tangible, two dimensional incarnation, and its one dimensional alter ego in system memory or a data file, it is important to note that a simple 3×3 square $\{H\}$, the building block of morphological structuring elements, becomes split into three separate groups of three pels each, one group per scan line [11, p.227] [48, p.128]. Scanning top to bottom, the upper three pels in this 3×3 square will appear first, followed by the middle group of three starting exactly one raster line length of pels (512 in my case) later, and the bottom three one raster line length after that. In other words, each pel in a raster data file follows the one above it in the image by 512 pels or whatever the scan line length is. It also precedes the pel directly below it in the image by the same amount.

This important realization is often made by seismic data processing students who must learn to program convolution procedures. It is directly applicable to both linear and morphological image processing. In this way, the two dimensional convolution which mathematically weights and sums a 3×3 neighborhood over an

entire image can be expressed as an equivalent one dimensional processing sequence, often with an improvement in program speed and a lessening of machine memory allocation needs. In any case, a two dimensional image processing procedure that works with the machine's logical storage model, rather than the mathematician's conceptual one, can incur less computational overhead than a program that trusts to a high level language to shield it from the realities of mass data storage. In other words, it is less wasteful of a computer processor's cycles to understand a few things about how it actually does the computations, and work with that knowledge, than to construct an algorithm in a way that happens to make sense in the language being used, but actually depends on the language's compiler being able to transform that idea into a correct but very roundabout procedure in the machine's language.

6.1.2 Graphical Vectors

A type of graphical data other than raster exists. *Graphics vectors* are a type of shorthand image data that are complementary to raster data. They are compact, symbolic expressions for an object that could be plotted or *rendered* in a raster array. For example, consider this graphical text string:

```
DRAW circle (25,55)(17)
DRAW circle* (35,35)(5)
```

These are text strings, or arrays of printable characters, that describe two graphical objects. They use the verb `DRAW` to invoke some vector to raster conversion routine that will render some vector statements into a raster array where we can see them, on a graphics screen or a plotter output for instance. Then, the command `circle` tells this program to plot a circle on the raster grid. The circle is a concept in $\mathcal{R}^2$ that can only be realized in certain sizes in a gridded, raster image domain $\mathcal{Z}^2$. The `(25,55)(17)` arguments to the `circle` command tell it where to plot some reference point like the center of the circle in the graphics coordinate system, often with the origin at the lower left and y axis increasing upward. The size of the circle

must also be specified, so with the (17) everything is set. An outline of the circle is plotted on the gridded raster array. The second line of text uses a slightly different vector graphics command `circle*` that plots filled circles or dots. The center and size of this dot are also given as arguments to the `circle*` command.

These two text strings of 32 essential characters can be rendered by the DRAW function into about 65 explicit pels in the raster array, perhaps 52 for the outlined circle of diameter 17 and 13 more for the crudely filled circle of diameter 5. Even with these very simple examples, vector coding of the circle objects has provided a 2 : 1 data compression over the number of pels affected in the raster array. To actually store these renderings, one might need to save square *windows* or small pieces of the raster array in the form of mini-scans. For this, a 17×17 plus 5×5 storage would require 314 bytes, plus the coordinates of the windows in the image. So the compression ratio begins to approach 10 : 1 for our simple example. This example is not contrived. Vector representations are so much more compact that it is worth spending much computational effort on their rendering.

Simple vector graphic descriptions are exceedingly useful when rendering graphics. Rendering takes a symbolic vector description and converts it into corresponding elements of a raster array. More complicated graphical shapes may be produced by combinations of basic vector graphic descriptions.

The types of vector graphics commands or symbols just presented may be familiar to readers who have used computers to draw graphics on screens or plotters. What is done by the machine between one's specification of a vector graphic command and the realization of it on a raster device like a screen or a plotter is a system process called *vector to raster* conversion. Since the process often takes a Euclidean notion like a circle and converts it to a gridded or spatially discretized realization of that concept, in so doing it must perform some two trigonometric calculations for each point plotted. As a result, the plotting of analytically defined curves onto a gridded domain often uses calculations with floating point numbers, the machine

version of real numbers.

In the machine, floating point numbers are stored in variables that are distinguished from integers in that they may have decimal fractions, exponents, or both. Integers only have the options of being signed or strictly nonnegative, long or short. Here again, a distinction is made between a machine's understanding of a number and the view a user is given through their programming language. Storage and computational power is available in some increments of bytes, or eight bits. A text character or letter can be expressed with relatively few possibilities, upper or lower cases, numbers, punctuation marks, etc. These have all been summarized in only 127 possibilities for English. That way, they only require 2^7 or seven bits of machine storage. And the 256 possibilities in one byte of data are more than enough for characters.

Integer numbers come in different sizes on machines. Some are 16 bits, while others are 8 or 32 bits. Add them up for yourself, but the `short` or `int` integer type has 65,536 possibilities with its 16 bits, 2^{16}. If one uses these same 16 bits to describe signed numbers, then they can range between -32767 and 32768. The binary number system uses the symbol K often when describing large binary numbers. In this context, $2^{10} = 1024 = 1K$. That is to say that `short` or `int`size numbers range from $\pm$32K, and `unsigned short` size numbers range from 0–64K.

Signed or unsigned, the range is set by the two byte limit on storage. If one wants to save a 17-bit number, one must save 32 bits, since four bytes is simply the next larger size available on most machines. These `long` type integer variables can cover a large range of numbers, to say the least. An average human life could almost be quantized in seconds and still be indexed by a signed `long` integer variable.

The floating point types allow one to record fractional numbers with an exponent. When one resorts to using floating point numbers, as certainly every scientist has done, the computer, which can only store integers, uses a conversion technique that separates all floating point numbers, even a representation of π, into a

mantissa and an exponent—two integers. The mantissa is something of the form .31415926535897932384626 and the mantissa is the 1 of $\times 10^1$, or whatever it needs to be for a given number. Like integers, floating point numbers must fit into one of two *storage classes*, the 32 bit or 64 bit versions. The single-precision as it is sometimes called uses 32 bits, but might as well be called half-precision since some watch calculators do more accurate mathematics than provided by this variable type. These variables are called the `float` type. The 64 bit floating point numbers are often used in scientific computing and are called the `double` type, or double-precision floating point variable type.

Together, these five variable types dominate most scientific programming. The single byte size of `char` and `u_char` for character and unsigned character data are often used in text manipulation as well as raster graphics based on a 256 gray level or 256 color display device. The `u_char` variable type is preferred for graphics arrays, such as those defining an explicit screen display in 256 colors or gray levels. The two byte size of `short` or `int` variable types is used for arrays specifying images to graphics devices with 32K or 64K colors, that is, 32768 or 65536 colors. The reason why a display device would throw away one bit in the 16 bit storage scheme is simply because 15 is divisible by 3. That translates well into three independent measures of red, green, blue color intensity at 32 quantized levels each, where 32×32×32=32768, or 32K and not 64K colors. The 32 bit `float` type has been neglected in many programs because it induces large roundoff errors in long trains of calculations, but it is useful for quick and dirty floating point calculations. The four byte size of `long` (integer) and eight byte size of `double` (float) variable types are used for integer calculations that deal with values over 64K, or for most scientific floating point calculations. For example, to pinpoint a specific pel in an array of 512×481 pels, one must find a single pel out of 242,767. This requires a `long` integer variable to describe the pel's position in the raster array, since in reality pels are *not* stored from `(0,0)` to `(512,481)` but instead are located at points along a very long string

from offsets 0 to 246271 in the machine's memory.

6.1.3 Vector-to-raster Conversion

The value of vector to raster conversion or VRC is that it turns a tangible concept like `circle (25,55)(17)` into the exact 52 or so points in the array between 0 and 246271 that make a digital circle visible on the screen. In doing this, the rendering algorithm uses some trigonometry, which requires floating point calculations, then a clipping or rounding of those precise results into lattice points that are actually available on the raster gridding. In reality, this digital circle is only an approximation of a true circle. In practice, knowing that about 50 points are needed to describe a digital circle of radius 17 on a rectangular grid, a vector to raster rendering program could calculate the precise location of, say, 100 points analytically defined along the perimeter of the circle, fit every one of those results onto a gridded realization, and plot them. Some of these calculations would be redundant, but certainly no holes in the circle's perimeter would ever result from such a rendering routine. In this way, vector to raster conversions become floating point intensive.

To enhance the performance of computers that must perform this rendering task, a *floating point co-processor* is sometimes installed in the machine. This is a special purpose hardware device that makes these floating point calculations run much faster than can the original processor. How? Because general purpose computers are integer devices. They do not need to use floating point numbers to compile programs, store files, edit dissertations, or even typeset them. Machines don't need floating point numbers for their internal work, but people often do. So in the same way that general purpose machines (secretly) store floating point numbers as a pair of integers, they have ways of making floating point calculations using those pairs of integers as well. They just can't do both the mantissa and exponent calculations at once. But floating point co-processors can do that and more. Designed more for floating point than integer math, they are much faster than the general purpose

machines themselves for the `double` variable type.

Another way that floating point co-processors speed things up is that they have large look-up tables programmed with transcendental functions like $\sin(x)$. The look-up table is an array itself. The difference is, they are arrays programmed with functions. Pick a number, say `0.5235987`, which is about $\pi/6$. Give that number to a math or floating point co-processor's look-up table that is programmed with $\sin(x)$, and one finds at that corresponding address in the look-up table the value `0.5000000` waiting for further computation. In similar ways, math co-processors can rapidly compute problems that use any of the transcendental functions, $\ln(x)$, $\exp(x)$, $\arcsin(x)$, etc. The look-up is a very powerful feature of these special purpose processing devices. It may also be located upstream of a floating point multiplier unit, so that a trigonometric function's value can immediately be combined with some coefficient in the next clock cycle. A simulation of this same procedure by a general purpose machine can take ten times longer or more with double precision arithmetic.

In general, the finer the lattice that the vector expression is being rendered onto, the more points that must be resolved before drawing the gridded realization of the graphics vector. Often, a floating point scaling value is applied before rendering, or a rotation, or a coordinate system change, etc. All these manipulations are made with floating point arithmetic to maintain accuracy, then converted to the integer coordinates of the raster grid only as the last step. The more complicated the manipulation, the more floating point intensive is the rendering process. For very finely gridded systems such as the LaserWriter that is printing this paper, a vector to raster graphics engine converts expressions similar to `ellipse(3,2)(5,3)(pi 6 div rot)(fill)` into an ellipse, five inches over and three inches up, two inches high and three inches wide, rotated thirty degrees counterclockwise, filled in with black, into the appropriate raster pels that should be black when the page is printed. Strictly speaking, vector to raster conversion does not often make use of mathemat-

ical morphology, except when a "paint" program has been assigned a particular brush pattern, and that brush is used to *stroke* some path. For example, if a calligraphic brush has been defined and is used to draw an arc, then that arc is drawn by dilation of the curve by the brush. The brush is in fact defining a structuring element to be used in the dilation of the points along that arc. Of course, the dilation operation is usually deeply hidden beneath a user interface. Certainly, a knowledge of mathematical morphology is not a prerequisite for using paint programs.

6.1.4 Raster-to-vector Conversion

However familiar the vector to raster conversion process may be to users of computer graphics, the complementary procedure of *raster to vector* conversion or VRC is both less familiar and more difficult to get workable results from. Mathematical morphology is important in many ways to the solution of these problems, which are quite different from VRC. Given a large raster array, how does one identify the contents of that array and describe it in a symbolically succinct way? And while vector to raster conversion often is floating point intensive, in terms of trigonometric functions in the rendering of analytical shapes, the raster to vector conversion problem is integer math intensive. Raster to vector is also queue management intensive, in that when several objects in an image are being identified by their boundaries or paths in raster data which is arriving line by line, many objects must be tracked in parallel and their features sorted out in a dynamic data base. Each scan line's miniscule addition to the boundary or path of objects in the image must be attributed to the correct feature as the scan proceeds. Mathematical morphology is particularly well suited to the filtering of shapes found in raster images.

Some transformations exist for searching the image for known curves of various sizes and orientations. This is the Hough transform [40, 45, pp 121–126]. Other types of feature enhancement are available depending on which criteria are useful in identifying a desired feature in an image. Some of these enhance the part of

the image that contains a feature even more. Examples of these are Sobel edge detection [40, pp 487–491], and segmentation or object classification by successive relaxation [45, p.173–177]. Some algorithms will follow a contour, at a particular brightness in a gray level image, or around the boundary in a segmented or classified feature in the image.

When classifying pels in an image for gray level brightness or boundary properties in indicator function images, it is important to be able to identify widely varying numbers of objects in images when one is using an RVC technique to characterize an indicator random function. For an algorithm to be stable under a wide range of inputs, it must have a database that is flexible and expandable when tabulating object features in the image.

Overall, there are two different approaches to this common machine vision problem of raster to vector feature extraction or object classification. Their differences are based in how one accesses the raster image data. On one hand, *omnidirectional* line followers require storage in memory of the entire image or a working subset, then they take the first element of the object found, and proceed around the border of that object until returning to the original point or reaching the end of the feature. Because these followers may track the feature in any possible direction, they require access to the raster data in a random fashion.

Rapid random access of large raster image files has been available in general purpose computers, but there is more of a problem when accessing those arrays in raster engine cache memory. This cache is where parallel processing image results are computed. The solution for some is to copy the raster data into the *host* or general purpose computer that is controlling the raster engine and work with it there. There is some cost to this, but it is a good way to go if one already has an omnidirectional line follower there to use. Still, these followers use more and more memory to perform their task as image resolution increases, or else incur even more program overhead by *swapping* sub-images from mass storage to memory as

needed, possibly more than once if portions of several objects appear in a particular sub-image or *tile*. From an information standpoint, the omnidirectional follower algorithm depends on the image being completely known before the tracking process begins, so that the algorithm can have access to any part of the image it needs.

From my perspective another concern arises about the applicability of omnidirectional line followers to my task at hand: the raster to vector conversion of a multiply connected network of thinned lines. My concern is a topological one. Namely, can the omnidirectional line follower work more efficiently or at all on a less restrictive topological class of features? From a part of topology, doubtless very near its cornerstone, is the notion of the *open set* [47, p. 66].

Aside from fascinating mathematical details like the way that gridding is imposed on continuous space can produce sets that are at once both open and closed, consider this: For a collection of isolated pore elements $\{X\}$ in an image, all objects being bounded, $\{X \in \mathcal{F}\}$ is a closed set. Necessarily ignoring the boundary of Z, our gridded array, the complement of our closed set $\{X^c \in \mathcal{G}\}$ is an open set. A homotopic sequential thinning that results in $Y = (X^c \bigcirc \{L\}); M$ from an open set will likewise be open.

In practical terms, an omnidirectional line follower $R(X)$ can circumambulate each member of a closed set once and be done with it. Its algorithm can prevent it from bothering with the same boundary twice, at least walking in the same direction. When following $R(Y)$, however, it can still follow boundaries in a local region Z of the open set Y, if it always takes the left fork at all path intersections, and has prior knowledge of the position and direction of its last visit to that node or intersection. Then it can be made to vectorize the sampled class of open sets $Z \cap Y$ in a bounded image. In the limit, the omnidirectional line follower that accommodates vectorization of a multiply connected or circumfluent thinned network will run over each line segment twice. In doing so, it will identify every closed loop in the circumfluent network.

The alternative approach to raster to vector conversion is called *raster tracking.* As the name implies, it is designed to work from raster data, not just in the accessible imaged form, but in the *flat* time series or one dimensional form as well. From an information standpoint, the raster tracking approach does not require more than a sequential *stream* of data. The entire image need not be known from the outset, nor be stored in its entirety in memory, nor be accessed more than once, except for a three scan line-long buffer.

Compared to the omnidirectional approach, raster tracking is more database intensive. Because lines and boundaries appear a few pels at a time, aside from simple horizontal lines, each object is sliced up in such a way that large gaps appear between successive additions to its boundary, and a new problem arises. That is, the information on not just one, but every boundary appearing in a given horizontal scan line must be kept track of in parallel. When bits and pieces of boundaries arrive, they must be squirreled away in the proper queue of the vector database. Also, because the raster approach is limited to tracking lines in only four of the eight possible directions in a rectangularly gridded system, simple shapes in certain pathological directions will appear as large numbers of very short queues. Thus, additional work must be done on the queues in the raster scan database before storage in a boundary database.

The problem that must be solved by queue management is that of converting the four-directional raster code into the general eight directional code. To do this, some of the queues are read forward, the same way that they were written, then when these have joined another, the reading jumps to the last member of that queue, and reads its list *backwards.* In this way, the backwards-read queues, which connect with the forward-read ones in a specific way, can produce the other four line directions that were not obtainable in a causal or forward-only reading of the raster image. I present this detail just in case the choice between raster or omnidirectional line follower should appear too clear cut.

The raster to vector conversion approach is useful in situations where the image data is a particularly large array like a remote sensing image, or is located on tape or similar streaming access device, or is located in another machine like a raster engine with a limited *bandwidth* or limited speed connection to the host machine. In all these cases, it is better to transfer the data once, in sequence, and be done with it—random access is undesirably slow. In practice, an array processor like a raster engine may have much poorer performance in random access of its cached images than in sequential or streaming access to them, and a very fast cache memory for synchronous streaming to the array processor may actually be much slower than the host's memory when used for random access of image data by the host.

And so there is no clear-cut advantage to one RVC technique or the other, without considering the circumstances of the particular application in view of the available hardware. The advantage of the omnidirectional line follower is that vectorized features are extracted one at a time, and completely. Further, the true eight directional *chain* code or boundary code is extracted in one step. This process depends on having nominally the entire image loaded into memory. The advantages of the raster scan technique are that only a few scan lines are needed at any given time to proceed with the feature extraction, a given effort at vectorization samples the image in an unbiased way, and multiple scanning routines can be synchronized at the end of their scan lines for progress in a three dimensional array.

For images with many objects in them, the omnidirectional line follower may effectively make several runs through parts of the image, to encircle each object. While that may be a disadvantage in terms of the omnidirectional follower's execution time, the queue shuffling at the last stage of raster tracking can become a lengthy process if features run in certain directions, particularly lines running at about 30° from the horizontal [11, p.228] in rectangularly gridded systems. When extracting large numbers of multiply connected objects from an image, the raster tracking approach creates a rather challenging database management problem, be-

cause it is essentially following boundaries on many objects at once—as many objects as are found in a given scan line. With appropriate post-processing of the somewhat complicated database of raster scan queues, however, Cederberg [11] has shown how one may extract the equivalent boundary encoding, based on the queues, that would have resulted from an omnidirectional scan.

When implementing one technique of the other in a classical computer architecture, the omnidirectional line follower seems to have a preponderant set of advantages. However, when an image analysis system is equipped with special purpose hardware such as a raster engine, the raster scanning technique may be chosen because it simplifies communication needs between processors, as well as the fact that some of the information needed by boundary following algorithms can be computed rapidly, in parallel, by the raster engine itself. Again in this case, Sobel's neighbor codes can be the guiding map for a raster tracking program. This is the distinctive character of Cederberg-Sobel compilation.

This can be an advantage particularly when the image being analyzed is a result just computed by the raster engine, and thus already loaded into its cache memory. This information needed by the follower is just another computation of local configurations or neighbor coding, as discussed in Section 5.4.2 and Sobel's paper[48].

For an example of the omnidirectional line following technique, implemented on a fast general purpose machine (MicroVax II), one could observe the image analysis system of Everest Geotech of Houston, Texas. For an instance of the raster tracking implemented on a system with a raster engine, one could see the Tracor-Northern models 5700 or 8500.

6.1.5 Faster RVC and VRC

Given the increased resolution of graphics images that continue to make both the vector to raster and raster to vector conversion problems more challenging, various

approaches have been engineered to increase the rate at which these processes can run. The popular approach for speeding vector to raster conversion has been to attach a math co-processor or floating point accelerator to speed trigonometric and floating point arithmetic calculations when rendering vector descriptions onto a raster array—video screens or laser printers. These co-processors, as found on IBM PC's, Apple Macintosh II's, and Apple LaserWriters, seem to be well integrated into the processing architecture of general purpose computers that support most users of these machines.

The raster to vector conversion process, however, depends on improved total speed or throughput in integer calculations. In general, for both linear combinations of array elements, as in two-dimensional signal processing as well as the nonlinear computations of mathematical morphology, something more than just processor speed is needed. A machine's throughput is affected by the time spent reading data from its memory storage, making calculations, often repetitively over a large array, and writing calculated results to memory storage as well.

In the case of processing large arrays where the dimensions of the input and output arrays are matched, array processing architectures use fast memory, often called cache memory, to hold a handful of instructions or, in special purpose machines, the entire input and output arrays. Some cache memory is special because it can be accessed in two complementary ways. First, it can be loaded with sequential or raster data, or a padded input can be written to randomly, as when a blank image (all pels zeroed) has a circle written to it by a circle rendering routine that only changes needed pels by random access of the (zero-padded) array. Secondly, this load of cached data can be fed synchronously to an array processor, one which takes the same amount of time to perform its computations no matter what is programmed (though limited by what is programmable.) A separate cache memory can synchronously accept the processed results, which necessarily are emitted by the processor at exactly the rate that data enters it. Then, the random access con-

nections between the second cache memory and the host machine can be used to move the data back into the host, to its memory or mass storage or onto a screen.

The synchronous data channel or data bus, for all its speed inside the array processor, is not very useful as a feature in a general purpose machine. To run the host machine, edit files, execute command lines, and do what a host machine does, an asynchronous bus is indispensable. While slower than the synchronous channel, it is much more controllable in its behavior. The reason it is slower is that many parts of the machine communicate with one another through the same channel. As on a party line, each unit must wait until the others are through talking, and only one component, the main processor, gets to speak without being spoken to first. Time is lost in ringing up some machine component, like a disk drive, waiting for its reply, then issuing a command. But in this slower and more careful way, chaos in the data channel is avoided. Likewise, the response of the system to some random input, such as an operator's key-strokes, can be very adaptable. This type of machine response is needed to interact constructively with users.

The synchronous machine, on the other hand is much more single-minded. Once programmed, data is always streaming to the processor from someplace, and results are always available from the processor someplace else. There is a party line among the machine's components, but it is for control purposes and not used for internal data. In addition to this asynchronous or *control* channel, every component has an exclusive *synchronous* channel both to and from the processor. These may become disconnected from the processor if not needed or if the processor has run out of ports where it connects to synchronous channels. The asynchronous channel, although it is slower, is the way that the host machine can control the array processor. Over it, the host can specify how and where the synchronous channel connections are to be made. It can also specify exactly how the processor itself should be configured. But the synchronous interchange of the array processor is isolated from the asynchronous world of the host machine. In fact, it is double insulated in the sense that the

host machine connects through its system channel, to the array processor's control channel, and then only data which passes through the *dual-ported* cache memory, both synchronous and asynchronous in its access, is it accessible to the synchronous processor itself. Data is extracted the same way, perhaps from a different cache.

The processing units within the raster engine or array processor are mostly compatible with the synchronous channel, but are manipulated over the asynchronous control channel. To preserve the synchronization at the output of the array processor, this compatibility requires that any function programmed into the processor take the same amount of time, whether adding, multiplying, dividing, or filtering through a look-up table, then doing something to combine these results before emitting a result. It must take the same time for anything, whether multiplying two small numbers or the two largest ones it can accommodate. This avoids corruption of the data by delaying more "difficult" results with respect to easier ones.

Random access of data, or more precisely random relative offsets between caches are accommodated by delaying one with respect to the other. This process is given a two-dimensional feeling with the use of shift registers that offset the pan and scroll of some cached image, just like a movie image. Pan values move the image left or right and scroll values up or down. These offsets are useful if one cache is being combined with or accumulated into another cache that is not offset. Large differences in shift registers are usually dealt with by wrap around. Wrap around is a common approach to array manipulation used in geophysical data processing, see Claerbout [13, p. 77]. It simply means that when a part of the image slides off the end of the screen because of shifting, it reappears immediately on the opposite side of the image.

Division is an operator that is implemented in most array processors for floating point computation, but not all array processors for raster work. The vision engine I use, in particular, does not have a feature allowing division by arbitrary integers in a single data pass. It does, however, offer either division or a second stage of

multiplication by 2^n. This is accomplished with another common component of array processors, the barrel shifter. This unit takes an integer's bit values and shifts them either left or right, bringing zeros in to fill the space left behind, or it rotates them left or right, with bitwise wrap around. The shifting has the effect of multiplying or dividing by 2^n, where n is an integer.

The bitwise rotation has a more morphological effect. With it, a nearest neighbor mapping could be made to experience a local rotation of each of its configurations. This would not rotate the image as a whole, only each element's representation in an NNM transform. Aside from that, the bitwise rotation can be used to swap bytes, in the sense of allowing two parallel byte-wide data streams to exchange paths beyond that point in the processor. Bitwise shifting has an accepted symbolic shorthand, <<, and an expression showing how it is used can be written `128 = 8 << 4` meaning that 8, or 2^3, shifted left four bits or multiplied by 2^4, is the same as 128, or 2^7. A bitwise shifting expression for division is >>5, as in 32 is `22 = 704 >> 5`. [1]

6.1.6 Raster Engines Paradigms

And so vector to raster conversion is speeded by the addition of a math co-processor, which is well integrated into most host systems. The raster to vector conversion problem, especially when using morphological filtering steps, is speeded by a special type of array processor, one which offers both the linear combination features of more traditional floating point array processors such as those used in seismic data processing, together with logical and conditional processing elements which allow one, pel by pel, to select among various simultaneously computed and available results—this is the morphological version of an array processor. The vision engine is a raster to vector co-processor, so to speak.

The general name for these integer type of array processors, whether they are

[1] Colloquially, bitwise rotation does something else that bitwise shifting does not. The rotation operator is a humane alternative to shifting for those programmers expressing facetious concern over the fate of bits which are shifted off the end of integers when using the >> or << operators. When shifting, bits that fall off the end are collected in the proverbial bit bucket.

used for vector to raster or raster to vector conversion is a raster engine, where the rendering of vector descriptions onto a raster array would be called a graphics engine or rendering device, and the converse, a morphological processor would be called a vision engine. While these machines offer high performance for their respective tasks, their synchronous data channels together with some parallel processing attributes make it cumbersome to encode programs for them in classical programming languages.

For the graphics engines, the needs of very many users can be served with a common set of graphical instructions, conveniently accessed through traditional programming language or newer, fourth generation languages such as Adobe Systems' PostScript, where fourth generation refers to the use, in the language, of a combined data and instruction object *stack*. With the ability to manipulate either data or instruction sequences with equivalent tools, treating them more or less equally as objects, the repetitive sequences that are sometimes needed in graphics can be represented more concisely, without many of the trappings that classical languages have with the definition of subroutines or functions.

In the vision engines, however, the needs of various users are not so clearly defined, because rather than sharing a common need such as printing a rendering of text and graphics on a piece of paper—where a page description language such as Adobe's PostScript works well—different users are interested in having their machines serve different needs, to find different properties in images of very different objects. In other words, at this time, there is more of a need to program vision engines for particular applications than there is a need to program graphics engines, which have even been designed around the language, as in the case of the engine running the Adobe PostScript interpreter inside every LaserWriter printer, where there is already an acceptable language interface. From my perspective now, the needs of image analysis are so varied that no one product has come into being to satisfy everyone, so this question remains open.

For all of its elegance, the Fontainebleau symbolism does not yet help one encode a morphological process any more than a double integral equation helps one to encode a two-dimensional Fourier transform. At least, in the Fourier case, it is possible to take the concept and write the discretized expression of its algorithm in terms of a known programming language.

At this time in the evolution of image analysis, it is necessary to take expressions of mathematical morphology and turn them into configurations of a vision engine. Then, with the component interaction among parallel processes within the vision engine, as well as constraints on the use of processes implementable over synchronous data channels, described on Page 167, many types of processes can be arranged, and many solutions exist to ways of implementing a particular morphological expression. The benefits of having a well defined language become very apparent when dealing with parallel processing problems. In vision engines, one is faced with the compound problems of deciding whether an array process must cache its results, stream them to a waiting processor, or operate in parallel with another processor while partitioning the data, a piece to each processor.

6.2 Morphological Lattice Concepts

In this section, I will refer back to the Fontainebleau symbol set and attempt to bridge some of the differences between that symbolism for machine-independent morphological relations and the realities of implementing these descriptions on parallel array processing hardware which lacks a structured programming language of its own. My goal in this section is to describe how one makes use of gridded domain, or lattice morphological concepts.

6.2.1 Gridded Vs. Continuous

In his book, Jean Serra [47] often makes a distinction between the continuous and discretized image domains, $\mathcal{R}^2$ and $\mathcal{Z}^2$, when describing morphological processing. In some cases, the Fontainebleau symbolism is almost deceptively similar when describing the continuous and discrete versions of some processes such as the hit-or-miss transformation. In linear image processing, the difference is plain in that with the continuous domain, expressions often use $\iint f\,dx\,dy$, while the corresponding expression in the discretized domain would be expressed with $\sum_{\forall x}\sum_{\forall y} f(x,y)$, very noticeably different. In carrying morphological concepts from the continuous to the discrete domain however, many of the concepts that describe well a process, such as skeletonization by the positions of the origin of a maximally inscribed circle in $\mathcal{R}^2$, can not be applied so directly in discretized space. While the meaning of the digitally computed skeleton may be very similar to that of a continuous one, its computation is approached quite differently, by sequential ablation or erosion from a variety of directions. In this case and in others, the morphological processes that are implemented in discrete space $\mathcal{Z}^2$ are often less direct in their approach to the calculation of a given morphological property than their $\mathcal{R}^2$ counterparts.

In $\mathcal{Z}^2$, there are many reductions of otherwise unbounded morphological methods to finite numbers of steps. Most processed pictures represent knowledge of only a certain local area Z, that which is contained in the image, being able to compute

the quench functions of objects having a finite maximal width, etc.

Often this simplification in the discrete domain introduces some subtleties that do not exist in $\mathcal{R}^2$, such as the finite number of directions possible in a gridded system. It is not always easy to make use systematically of these simplifications. For instance, on a rectangular grid, is it better to perform ablation from the four cardinal directions, or from all eight possible? If the diagonals are included, since the centers of grid points are further away from each other in these directions, should these ablation directions be performed less often than the cardinal directions in a rotational sequence? What shape of compound structuring element is best to form as the aggregate of ablations, that which is the inscribed shape at larger quench function values?

The sequence of steps that is logical when describing a morphological process symbolically may not be required if a given step does not need to make use of a previous step's intermediate results, as in the absence of conditional processing. For this, it is important to consider whether a rearrangement of the sequence that is expressed symbolically affects the computed result, or whether a reduction of two sequential steps to a parallel pair, which would improve the algorithm's efficiency, should be considered. For example, conditional processing may make use of results computed from by-products of earlier calculations and the conditional processing set may be more efficiently computed at the same time as that part of a transform symbolized to the left of a semicolon. In that way, the conditional process may execute without delay as soon as intermediate results become available, if the conditioning set is ready then or before, computed by a second processor. I have used this approach in coding FMT.

It is difficult to evaluate these modifications of the processing sequence with the symbolic description of the morphological process alone. Indeed, without a machine on which to implement them, there is no need! But with a machine, be it special purpose or a general purpose one with a friendly compiler on hand, something else is

needed, and that something may depend on the actual array processor that is being used. For instance, if the vision engine being used has more than one processor, an additional consideration should be of how much processing can be performed without storing intermediate results in cache memory. That is, if an intermediate result does not need to be shifted relative to another, it is possible to stream the result of one processor directly into an input of another, and so on.

These considerations are central to the concept of real time image processing, where only a small delay is needed to compute morphological results, which are then displayed on a screen as fast as the data arrived, in real time. Also, given the possibility of multiple processors in a vision engine, should the resources available be used in a serial sequence for processing as described above, or should the process divide or partition an image into disjoint subsets and operate in a parallel simultaneous fashion? How can one systematically study the possibilities when deciding how to map a given morphological symbolic expression into an actual processor configuration for an algorithm? This is a rhetorical question, perhaps, but an important one for any approach to an optimal algorithm.

Since these vision engine array processors are morphologically oriented, how can one make use of conditional processing of an image, selecting one from among several intermediate results, simultaneously computed and available, with a pel by pel decision table operator? These sorts of considerations are common concerns in morphological image processing and present strange new features to be considered when programming, in contrast with floating point co-processors.

6.2.2 Nonlinear Operators

Quite importantly, one may ask, "Why bother dealing with these architectural details that are so far removed from rock properties of interest?" In response, I say that the understanding of morphological concepts is really only tested by one's ability to apply them to data of interest, or enable others to apply them, particularly

for automating rock property identification. Morphology's nonlinear operators are important tools when programming machines to do more human-like tasks. First, morphological processing is slow on general purpose machines. This does not prevent research from progressing but it limits the number of useful applications if the process is too slow. Secondly, to my knowledge, no morphological processing language has yet been developed. Approaches to new applications must pass through a stage of morphological processing research, or else make incremental improvements to an accepted state of the art. Thirdly, the most significant application I see for morphological processing is in the earth sciences as a whole. The way to succeed in this most challenging application is by making the concepts more accessible, *both* theoretically and technologically. By using special purpose hardware to compute results before user boredom sets in, together with a graphical workstation to easily and powerfully edit processed results for presentation, the application of mathematical morphology to the earth sciences succeeds when workers can use its results—independently.

I include here a description of a vision engine's architecture. Until earth scientists can purchase a general purpose morphological computer, whatever that may be, it will be necessary to use and perhaps program these vision engine array processors. As an important component of rock property measurements made from image data, these computational engines make the nonclassical operators of morphological processing accessible to the earth sciences in the way that superconducting magnetometers made a difference in paleomagnetic data quality over spinners. If one had only an early version of these magnetometers and had to frequently tune it to work with a range of samples, wouldn't some model of how the magnetometer works be an important prelude to a discussion of measurements affected by calibration corrections? I suggest that it would, and so in an analogous way I describe the parallel architecture of the processor I use.

6.2.3 Parallel Process Restructuring

Morphological processing work is done fastest on special purpose hardware, that may come in a variety of configurations depending on the manufacturer, but should share some functionality with most raster engines. An appreciation of these common features is, for the time being, a substitute for knowledge of a morphological processing language that does not yet exist. For example, in seismic data processing, where linear two dimensional filtering techniques are used almost exclusively, it is possible for geophysicists and seismic data processors to encode algorithms in FORTRAN, and to make use of standard subroutines such as fast Fourier transforms that run on array processors, without understanding how the algorithm was encoded on the array processor. This approach works for two reasons. First, the process calling the array processor subroutine is running on a general purpose computer. Secondly, it passes its results to and from the array processor through system calls in a high level language.

In morphological processing, where almost all the manipulation of data, short of analyses based on boundary coding, can take place inside the vision engine, even the simplest use of scripts, where one calls a series of preprogrammed morphological functions, requires a basic understanding of how the array processor is connected to its various cache memories or frame buffers. Simply varying the sequence of steps in a working script often provides more of a rude introduction to the intricacies of vision engine architecture than it does a useful computed result.

The Fontainebleau symbolism for mathematical morphology does a complete job of describing symbolically the possible combinations of morphological steps. Likewise, classical programming languages and calculus together with linear algebra describe the linear processing sequences that are used in, for example, seismic data processing. And FORTRAN allows one to convert these classical equations into workable code. This type of *formula* to *translating language* model worked very well as long as the architecture of the processors remained relatively constant, as

they had for many years in mainframe computing. With the introduction of very fast multiple-processor mainframes like the Cray X/MP around 1984, more speed in multiuser computing was gained, but the processor model, based on a moderately sized instruction set, was not so different that it challenged the possibilities of classical programming languages. Vision engines, with their synchronous channels and programmable architectures, *do* challenge this model.

Without the fortunate circumstance of a processor designed around a language[2], I have made headway in moving concepts expressed with the Fontainebleau symbolism into algorithms using two simple but nonclassical operators. These describe what are nearly generic features of vision engines. While the details of their use may vary from machine to machine, these basic features are likely to be present on any vision engine.

6.2.4 Neighbor Code Transform

One important concept in realizing a morphological process is to recognize that the structuring elements, often given in published algorithms in template form, are shorthand for what needs to be made into look-up table specifications. engines. In special types of convolution engines it may be possible to have the eight nearest neighbors available simultaneously for input to a look-up table.

A more generic approach that I have followed is to use a single array processor to put the result of a logical test on each of the eight near neighbors in a rectangular grid into one of eight bit slices available in the frame buffers. In this way, after the local configuration transform or NC has been performed in a series fashion, all that data that is needed for morphological processing with 3×3 structuring elements centered on their origins $\{o\}$ is available in parallel. The central bit, or status of each pel itself, comes from an indicator function overlay. That way, many templates, including those from the hit-or-miss transformation that require results from two tests, can be available on a single 16 bit wide synchronous channel.

[2]The Apple LaserWriter series and Adobe PostScript, for example

If a structuring element requires that two neighbors must be equal to the iteration number and five neighbors must be nonzero, and one doesn't matter, it is possible to take two NC-transformed versions of the image, one which was tested for those neighboring elements which were nonzero and the other tested for those neighbors being equal to the iteration number, place the NC transforms into two frame buffers, where they can be streamed together simultaneously, at a single input to a 18 bit pipeline processor. This is what I have done with the FMT algorithm.

With two 8 bit LUT's any 3×3 hit-or-miss transform for each pel can be computed from an NC transform in one clock cycle. Other parts of the processor like multipliers, adders, and barrel shifters can be set to no-operation status. All the tests implicit in any hit-or-miss transformation's 3×3 structuring element array are set up in the LUTs. Essentially, by combining the concepts of the need to transfer structuring elements into a LUT format when implemented in vision engines and the view of NC transforms of the original images as morphological summaries, it is possible to run several morphological tests simultaneously in the vision engine.

Likewise, within an iteration of some morphological processes, it is possible to make use of a neighbor coded image more than one time. If this is true, then it may be possible to take advantage of a change in the processing sequence from that which would be necessary in a classical language on a general purpose machine. In some circumstances it is possible to manipulate the Fontainebleau symbols that describe the process being coded to see whether a proposed change in the processing sequence defines a different morphological process or not.

Likewise, an important nonclassical concept built into some vision engines is that of conditional processing, wherein gray level data is overlain by indicator functions, a binary overlay or two, which are used to define which one of several results will be emitted by the processor for that pel in the output stream. Like the neighbor coding, this conditional processing sequence can not always be described symbolically, although there is a provision for its simplest cases in the Fontainebleau symbolism,

with the $;Y$ operator.

The considerations that should be made for restructuring algorithms in raster engines are straightforward and many have been mentioned, but I will summarize them here. The types of templates or structuring elements expressed in many different transformations should be adapted to fit into LUTs, and then operate on neighbor coded images which, once computed, may be used in several steps of a given morphological process' iteration. It is important to identify how far it is possible for a processing sequence to progress before it needs data feedback, or new intermediate processed result before it can proceed. For multiple processor vision engines, the processing sequence can stream data from one processor to the next, until it is limited by this need for feedback. When the multiple processors divide tasks into parallel simultaneous sequences, where processors perform the same computation at the same time on partitioned data, results must agree at nonoverlapping borders between the partitions of the original image. Since morphological processing often involves the use of neighbor information, this condition may be hard to meet without overlapping image data. Once data is sent into the vision engine for morphological processing, the longer it can stay there before host machine operations, the better.

6.2.5 Processor Architecture Maps

When one is traversing the region between the Fontainebleau symbolism and an implementation of an algorithm on a vision engine in microcode, the generic term for parallel-processor machine language, the vision engine map is an extremely useful construct. In it, the configuration of the vision engine can be expressed graphically. Since the hardware components that are interlinked in a particular way that can not be changed, their structure is plotted graphically as shown in Figure 6.1. The actual functions that the various components of the engine are set up to do for a given configuration are inserted as labels at each of the nodes in the engine map. Also, the

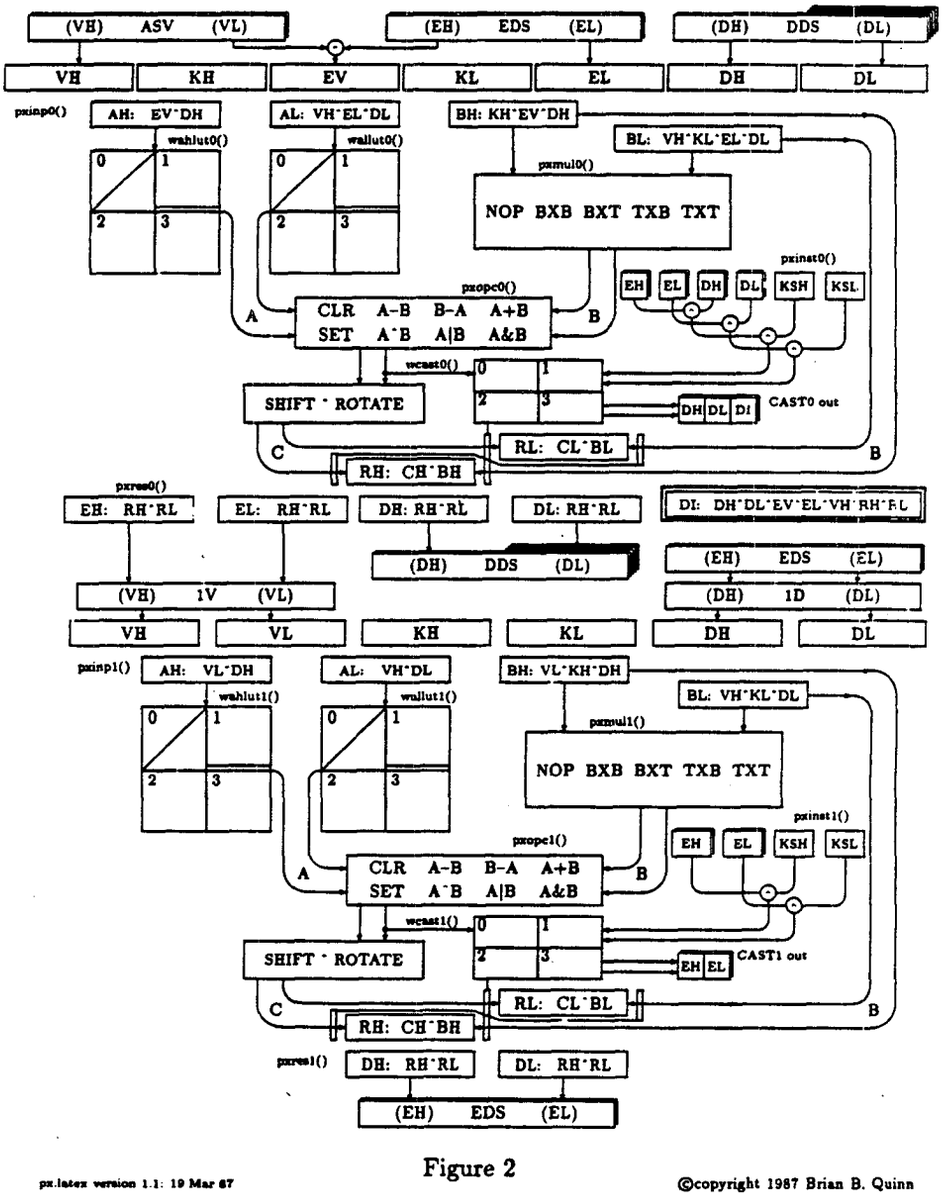


Figure 6.1: A Recognition Technology, Inc. PX-501 processor map

data caches are symbolized with their respective offsets. The logic interconnections of the engine, such as whether a multiplier section is set up for multiplying binary u_char or signed char variables, whether a barrel shifter is set up to shift or rotate a certain amount, and all other options are summarized. The LUT table functions can be plotted in small graph windows of the engine map, and all of this information can be consolidated onto a single page which graphically interprets the machine state from a particular microcode word, just as the processor would.

The diagram represents the processor at a particular step of a morphological algorithm. Any possible arrangement of these vision engine components determines an explicit processor setup code, that is, a particular microcode description of that configuration. Then, together with LUT functions, the microcode is implicit in any given vision engine map configuration, and vice versa. The engine map could be generated from the microcode. This microcode word is also equivalent to a sequence of function calls which, for instance, could change one aspect of the vision engine's setup at a time, such as setting the arithmetic logic unit to subtract two short integers rather than add them. Nothing more complicated than that. The microcode is a summary of a sequence of setup function calls for the array processor made by the host.

For the particular vision engine that I have used, a microcode word is 24 bytes long, principally because the engine has two array processors in it. This means that there are *very* many processor configurations available. Considering the multitude of functions that can be expressed in LUTs, there are even more possibilities. The extremely large number of possibilities that could be explained in microcode underlines the difference between vision engines and general purpose machines, ones which have only 200–300 possible instructions.

With only 200 or so commands understood by general purpose machines, more complicated processes are simply longer and longer serial sequences of simple instructions. This all works well for the general purpose machine because of its asyn-

chronous system channel. If it takes the machine 11 times longer to divide two numbers than to add them, fine. The rest of the system will wait for the main processor to figure the answer. This is how general purpose processors, which are truly integer calculators, can perform floating point math. It is just done in a roundabout way that burns up clock cycles.

Such a variability in response time depending on the complexity of a problem would wreak havoc on a synchronous channel system. So to work within the constraints of that higher-throughput environment, array processors, which must do everything at the same speed if they are to do them at all, are designed to be as adaptable as is possible. But maybe, just maybe, one of those 2^{192} possible configurations will do just what is needed, and replace a general purpose machine serial sequence that took 16 steps or more, up to 100 clock cycles. That is how array processors do their thing so fast by comparison.

To program a 24-byte microcode word, someone must figure out which possibility out of some 2^{192} choices for each step is the right one for the job. Nonlinear filtering algorithms may take intuition to implement. When it takes several sequential configurations of the raster engine to accomplish a task, it is hard to analyze what is the most direct solution from the full set of possibilities. It is a matter of programming experience to get from a configuration that just "feels" right to one that works as it should. And so a microcode description for a vision engine is less like a sequence of assembly language or the assembled output of a FORTRAN compiler than it is a plumbing diagram, where special numerical "line filters", the LUTs, work together with a collection of operators, one at each node in the multiply connected "plumbing" system of the vision engine. Hence, the alternate name used for some vision-oriented array processors, pipeline processors. The plumbing analogy carries with it the nature of synchronous data channels within the vision engine. The data enters the processor at exactly the rate at which it leaves it. Feedback may not occur within the synchronous processing sequence without caching the results in a

frame buffer, then recombining them with an original.

In all, there seems to be a fairly large gap between the morphological concepts presented in the Fontainebleau symbolism and the nuts and bolts of actually implementing morphological processes on a vision engine. But a small handful of concepts common to "generic" array processors, and a vision engine map for one's particular hardware can together help one find the way from symbolism to algorithm, to microcode, and then the processed image at last.

6.3 Look Up Tables

In carrying a concept from symbolic morphology to a processed result, some compromises need to be made on both fronts. First, the symbolic description of the algorithm may be made less symmetrical and more representative of the actual steps needed in a digital implementation, and also, the algorithm can specify the processing steps in terms of a few simple concepts. In this section, I hope to define a few *atomic* units, so to speak, of concepts to be built together when thinking about array processing rock images in vision engines.

Any given configuration of microcode with look-up table functions can define an engine map as an algorithm design tool. The map helps one manage the raster engine's memory and can plot the sequence of arithmetic operations that are performed in that configuration. The program that defines a morphological processing sequence is a series of engine maps, one leading to the next, connecting at points where data is either stored temporarily in cache memory or streamed over a synchronous channel from one processor to the next. In the absence of a compiled language for morphology in vision engines, some of the simplest codes that would be implemented in a compiled language can be very frustrating to fit into the environment of a vision engine. The engine map is useful for those working with vision engines, whether for instruction in its use or for programming. For those programming the vision engine, the map may clarify those options that are configurable when choosing among processing sequences. Also, there is psychological value in a graphic to ponder when faced with 2^{192} possible choices, to help clarify thoughts.

For users of pre-programmed functions on a raster engine, the graphical representation of the vision engine for each function used helps one understand how the functions will interact with one another when they are linked. For instance, a given function may leave a computed result in a second or third cache memory, instead of overwriting the original data, so that subsequent results can continue to access to the original source as well as the intermediate results of processing.

Without a graphical notion of where one's previous processing steps are leaving cached results, it is possible to become completely blocked and unable to use even a simple sequence of pre-programmed morphological functions. For these users, if the function being used is documented with all the sequences of engine maps it invokes, where the data is streaming from and to in each step, it could be possible for users at an intermediate level to write macro instructions defining a script or processing shell without repeatedly selecting the same sequence of basic operations.

The look-up table part of the engine map contains linear, nonlinear or comb functions. It could express transcendental function approximations or morphological structuring element or template filters acting on neighbor coded image data. For example, the linear ramp look-up table with a slope of 1 is the no-operation mode for the LUT. Whatever comes in leaves unchanged. If the LUT has been programmed with a linear function of slope 0, all input values result in a fixed value being output. This is most useful if some values are not the same as all the others, as in a linear LUT function which outputs 255, also written in hexadecimal with the `0x` prefix, `0xFF`, for all input values except those that are 0, and puts out `0x00` for zero. In this way, the LUT provides a nonlinear operator to flag nonzero values as it makes available a bit at every possible slice in an eight bit, or `u_char` variable type image.

The comb filter is a nonlinear type of LUT function that is simply a more varied adaptation of the slope 0 linear filter above. A comb filter is often used for image segmentation or classification, where certain values or ranges of values are used to identify a class of feature in an image based on its brightness value or the result of some intermediate computation, while other values not meeting these specifications are zeroed. Thus, with certain values being flagged and others zero, the LUT function has some high peaks and other zeros, giving it the appearance of a comb, possibly with a few teeth broken off.

Transcendental function approximations such as exp(), ln(), and trigonometric functions sin(), cos(), etc., can be expressed in discretized approximations as LUT

values. The exp() and ln() LUTs together with the arithmetic operations of addition and subtraction can approximate a global-divide of every pel in an image by some constant, or even by a pel by pel division by another image. The accuracy of this approximate operation can be improved using LUT pairs.

The template filters are special cases of comb filters which, when operating on neighbor coded image data, are spatial segmentation or morphological structuring element tests. That is, they can be used to identify elements in an image that are the endpoints of lines, testing all eight directions simultaneously. They can find sharp corners, or smooth edges facing certain directions, etc. Most interesting morphological algorithms can make some use of structuring elements implemented in the form of comb segmentations, programmed within a vision engine as LUT functions operating on neighbor coded versions of the input image.

While the design of a vision engine map is almost unlimited in its possibilities, limited only by the imagination of its designer, there are some basic properties of useful engine maps that would characterize them from among the wide range of possibilities. It is important to show in the map both the cache memory arrangement and synchronous channel topology as well as the specifics of the processor's configuration. The input to the processor is controlled by the settings of the pan and scroll registers on cache memories used for input, which change the position of an image. The engine map must be a *directed* graph showing clearly the data flow from cache through processor and back to cache again. It must be clear how the data is progressing from each cache, possibly through multiplexers selecting cache memories for input to the processor and through the various array processor components as well. With a graphical description of these in the map, the challenge of visualizing the processor's configuration is eased.

In an engine map that is drawn with a page description language it is possible not only to label the source and destination image caches but also to include small halftone plottings of the raster contents of cache memory in the map itself. This

is easy to do when using Adobe Systems' PostScript to describe the engine map. This can emphasize that a certain cache has a gray level image, another has its inverse, another a segmented version, and so forth. Not only can the cache memory be symbolized by a little box with a label on it, the label could be adjacent to a window which contains a small image of the raster contents of that cache memory for the configuration. When images are copied from one cache to another, this graphical approach could be very illustrative. Figure 6.2 shows what this would look like, using a microcode interpreter that writes native PostScript. Also, the LUT functions can be plotted as part of the engine map, making it apparent to the initiated whether a transcendental function, a linear ramp, or some comb segmentation has been loaded into the processor's LUTs at that point, as shown in Figure 6.2.

As another detail specifying a vision engine algorithm, the LUTs can be defined more than qualitatively by describing graphically and also in tabular form the relationship between the structuring element templates and the LUT values for each input, tabulated. Given some accepted convention of nearest neighbor mapping, such as the classical counterclockwise direction sequence used by Freeman in 1974 [23], a standard neighbor coding [48] convention would make tabulated LUT values more widely useful. With a convention, it is possible to explicitly tabulate the LUT values that are implied by any hit-or-miss transformation structuring set. With these tabulations of LUT values, together with the engine map, the configuration of the vision engine for that particular step in a processing sequence would be defined explicitly for a generic vision engine. I have presented both the FMT and CSC algorithms in this way.

As another detail in defining the implementation, the graphics display board often associated with a real-time vision engine usually has its own set of LUTs mapping a single gray level image streamed to it from the processor into 256 or 512 colors from a palette of 2^{24} or 16 mega-colors. This is not unlike a Macintosh II color screen, where the frame buffer is only 8 bits deep, and can only mark 256 different

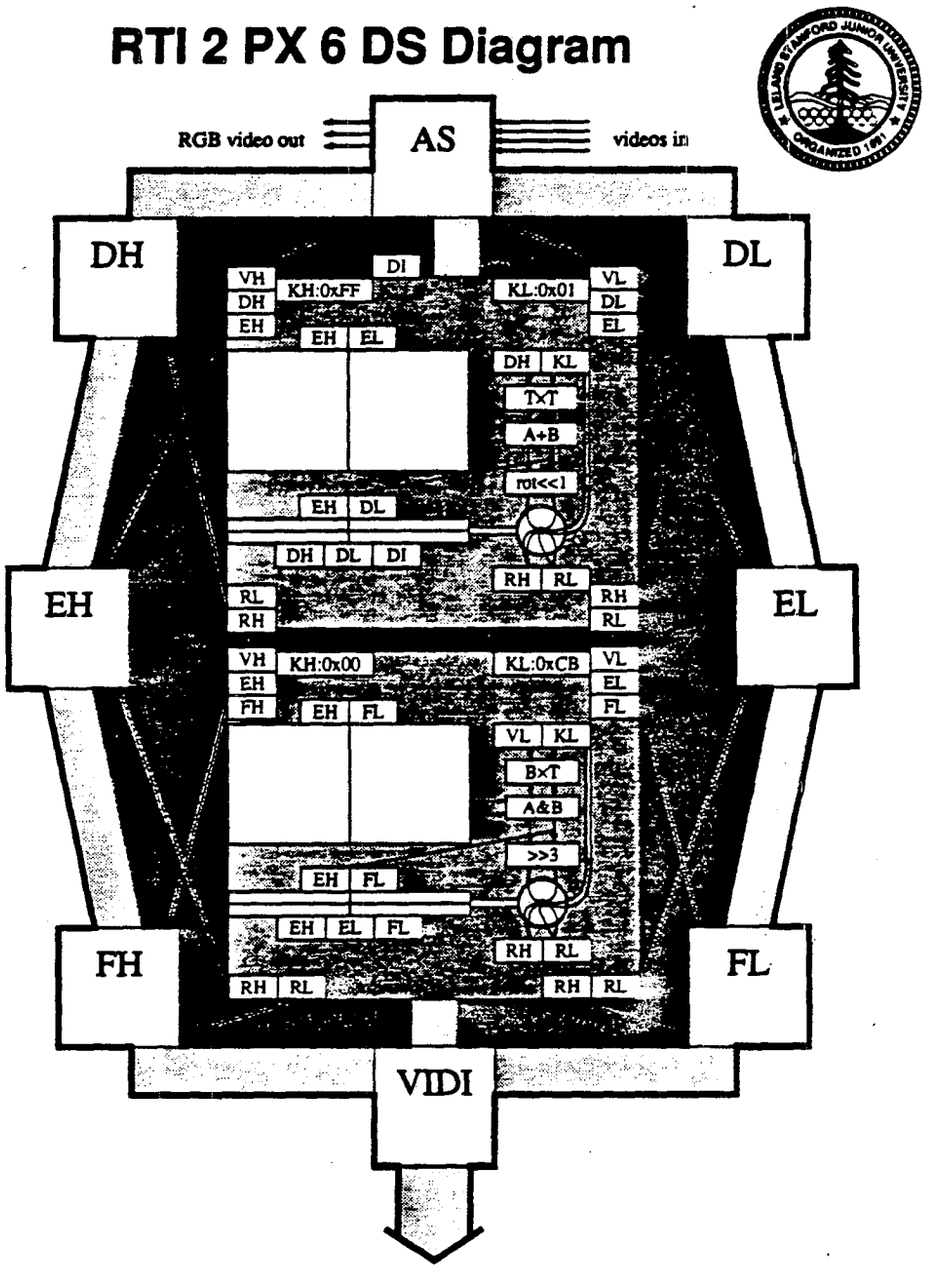


Figure 6.2: An PostScript engine map of the Recognition Technology PX-501 pipeline processor.

levels, but each of these can be mapped independently into one of 16 mega-colors because the screen display is the result of three independent color guns, each of which filters the frame buffer with its own LUT. Thus, 256^3 happens to be much bigger than some people suspect, 16,777,216 colors!

Since a programmable LUT is a digital device, the digital to analog converter board that drives a display monitor for the vision engines is the last chance to post-process a digital image. Many graphics boards[3] do it in grand style with three independent LUTs before the image is output to a screen. By defining the various colors in LUTs with these tabulations, the setup of the digital to analog converter, or analog subsystem, in the vision engine can be characterized. By plotting these LUT functions, and engine map can display the setup of the graphics board used in a function.

The way that conditional processing is used in the vision engine can be defined by its own version of the LUT. Since the LUT has come to mean an in-line type of filter that takes, say, eight bits in and maps it into eight bits out, often in one clock cycle, this function is not quite inclusive enough to describe conditional processing. Although it is defined with its own type of LUT, the precise name for the tabulation which controls conditional processing in a vision engine or pipeline processor is a CAST, for control and state table.

Conditional processing is controlled by indicator function overlays, more commonly called state planes or the state bits for a particular pel. Each pel has its own state bit(s), and their setting determines which of up perhaps four results gets put into that pel. What goes into the CAST as a function of the state bits is the *control* part of the table—it maps a particular state bit combination (1 of 4) into a particular result (1 of 4) to be emitted in the output stream for that pel. Sound restricted? Remember, any processor with 2^{219} programmable functions may have a few tricks up its sleeves.

[3]Number Nine, Targa, RTI, ITI, and others

The particular mapping of four inputs for the four state bit possibilities is also a function of the *data* value of that pel. Thus, the particular $\{4 \Rightarrow 4\}$ mapping that is used on any given pel may also vary depending on its 8-bit data value as well as its two state bits. This defines one of 1024 conditional processing control statements, a decision that is made by the vision engine independently for each pel in the image. And that's only the control half of the CAST story.

The state bits that are sent out from the processor, perhaps along with the data as *tokens* to second sequential processor in the pipeline, or just to be stored in the state bits of the caches, are set as a function of which of the 1024 conditional processing conditions were satisfied in the last processor to handle that pel. So the conditional processing result can be flagged, or a state bit left unchanged to pass along an original indicator function, or whatever is needed. This is the state half of the CAST story—a brief history of that pel's operations that can be passed along as a token to the next processing step.

The particular numbers I have used in the preceding paragraphs have been those I am personally familiar with in the Recognition Technology RTI-1210 two processor–four cache model I use with an IBM PC-AT host. More advanced, and expensive, is the Pixar model that has *four* processors, each working with 12 bit data, where three processors can work on each color independently at 2048×2048 resolution, and each color also has *four* state bit tokens of its own, that are collectively processed with CAST results in an independent pipeline processor. Well, at least I can say my system interfaces with televisions and videotape machines more easily...

To reiterate, CAST refers to those tables where conditional processing criteria are coded, as well as the output indicator function or binary overlays on the results that are used to communicate a message to the next step in a conditional processing sequence. This information passes through the processor on a separate channel from the data, although tokens pass through the processor, pel by pel, synchronously with the data and are always written in cache at the same time or emitted from one

processor's output channel to another's input in sync with the data.

I have converted structuring set templates into look-up table values that operate on neighbor coded image data, and conditional processing sets into control and state table logic. My motivation is that these are the efficient means of realizing morphological tools on a vision engine. These techniques make realizaions of morphology run fast enough to be useful as earth science applications. The LUT and CAST tabulations unambiguously define the nonlinear operator that is to be performed by the vision engine. They specify the exact function that must be loaded into the engine's LUTs and CAST to affect the desired morphological result.

6.4 Numerical Approach, Reflections

I have written this section to describe my approach to bridging the gap between formal presentations of mathematical morphology and the hardware product manuals that ignore the existence of such a formalism. Using practical steps to the best of my experience and ability, I have implemented these techniques on an image analysis system equipped with features of the most powerful vision engines, like multiple processors and token passing. I wanted to present the common vocabulary of computer graphics and machine vision work, including terms of vector to raster and raster to vector conversion that I hope are now clearer to the reader than before perusing this section. I have described some of the choices made by computer scientists and engineers when specifying and designing machines optimized to do one or the other of these graphical conversions. Then concentrating on the *vision* raster engine, the co-processor that helps raster to vector conversion, I have described the approach I developed for myself when realizing the morphological lattice algorithms as microcodable programs for my specific vision engine.

I described two concepts, the neighbor coding and look-up table representations of structuring elements, that together nonlinearly transform data in the manner of a convolution by multiplication in the Fourier domain of linear filtering. I expressed

the graphical approach I use when coding morphological operations on vision engines. Finally, I described the graphical approach that I used to document my work and to present the algorithms I developed in my search for morphological approaches to rock property estimates from rock imagery.

Chapter 7

FMT Discussion

In this chapter I describe properties of the fast Montoto thinning (FMT) algorithm and its results as I present them here. The first three are properties of the thinning operation itself. They include its estimation of pore throat width from FMT values, that are what remain after thinning, the estimation of pore area from only the quench function values, and the relation of this information to that available from sequential openings and object counting. The latter topic compares FMT to petrographic image analysis, from the University of South Carolina. I also consider the problem facing the use of the quench function values as network representations of the pore space from two dimensional data, namely, that pores in a thin section do not connect with one another. This leads to the question of how collinear segments of these quench function values can be made to connect in a manner statistically representative of the original rock. This problem I refer to as the channelization of collinear segments. I deal with in two ways, one of which operates on the array data, and the other operates on extracted results. These are obtained from Cederberg-Sobel compilation (CSC), discussed earlier.

Following these three sections dealing with properties of the Montoto thinned result, I present two other sections that discuss some of the types of analyses that appear at this time to be easily obtained from two uses of the thinned results. One deals with the types of analyses that could be obtained from the histogramming of different intermediate results. This is a type of analysis that is based on the

raster data itself, and by-products of the computational process for the fast Montoto transform. Then, I discuss a wider class of analyses, which are based on CSC vectorization of the quench function objects in a thinned results.

7.1 Throat Aperture Estimation

As with other types of sequential thinning algorithms, fast Montoto thinning produces a *quench* function, describing for a given member of the medial axis set the iteration at which it had joined the set. Fast Montoto thinning also presents an interpretive problem. The challenge arises from the fact that the medial axis is not actually computed from a maximally inscribed circle, but is instead the result of a sequential (iterative) thinning operation, using structuring elements that ablate from various directions. A permutation of the four directional sequence can not change the position of a thin arc by more than one pel, but can change the width value by up to two pels. The interpretation of these quench function values, in terms of the maximally inscribed polygons which they represent, should be considered of fundamental importance when interpreting the results of fast Montoto thinning. In fact, because they are the results of sequential ablations from just the cardinal directions, the quench function defined by the published algorithm [2] most accurately represents an inscribed *square*.

In Bel-Lan and Montoto's published specification, a test was described that involved the use of 1×4 regions to identify even- or odd-width areas in features, along with two scans of the object by the ablation function from *each* direction, for every iteration. The 1×4 neighborhood test described in [2] was performed at every second scan in this approach. As a result, the error in recovering the total area of objects from a simple sum of quench function values is about four times less with this eight scan per iteration approach than that I have implemented in the one scan each iteration of fast Montoto thinning. Also, sensitivity to single pel protrusions, as recorded in arcs, was greatly reduced.

I have avoided their exact implementation because I wished to use only morphological procedures that could be defined in terms of 3×3 neighborhoods around each image element. The 1×4 test to check for even or odd width exceeds this requirement. Also, the multiple scan approach represents a sequential rather than parallel approach to algorithm design, as the 8+1 sequence maximizes the work done each scan, at the expense of larger search neighborhoods for logical testing. This efficiency tradeoff between the cost of scanning and the cost of testing is different in vision engines. Since this is not strictly compatible with the information contained in the neighborhood that can be bitwise summarized in an 8 bit frame buffer, what I have implemented instead is a combination of the thinning, and thinning-2 algorithms given in [2], without the 1×4 neighborhood check, and with only one scan per iteration. I have thus avoided this test as well as the choice between 4 or 8 scans per iteration approach described in [2]. As a consequence, the intermediate result at every iteration is the result of a single scan.

While this is less efficient than the original algorithm for a general purpose computer, in comparison to the published approach, it is better suited to the capabilities of a vision engine. And so the quench function which results in the fast Montoto transform may have an uncertainty in width of ±1 pel because the sequence of ablations defined for the hit or miss transformations, in order to keep the thinned lines smoother, uses opposing directions cyclicly. That is ablation in the sequence of cardinal directions N,E,W,S,N...

In this way, long thin objects that are running E–W will have almost no pels ablated from them for two iterations while ablation "melts" from the E or W. To recover the dimension of the maximally inscribed square from the quench function, 1/2 of the quench function value is used, such that opposing directions map to different widths. This is accomplished most easily by using the NEWS sequence, rather than the NSEW sequence. With NEWS, all orientations of lines are equally likely to have a given integer width value. With NSEW, only EW ablation creates

even widths, and only NS ablation creates odd widths. Iteration value can be mapped to width by simple $w = (q + 1)/2$ integer division of quench function q to width w.

In [2], the multiple scans each iteration make it necessary to consider the quench function value to be either $2n$ or $2n - 1$ where n is the iteration value [2, pp 39–40]. But from a vision engine standpoint these needs conflict. On one hand, the algorithm should save a medial line pel from destruction, and based on a test that depends on a 1×4 configuration *at that time*, assigns either a width $2n$ or $2n - 1$ to the preserved pel. Fine. The problem from a conditional processing point of view is that in distinguishing medial line pels from the actively ablated object at iteration number $2 \times n$ confuses many global tests that could otherwise be implemented on generic parallel processing hardware. Instead of saving the pels with a width value about twice the iteration number, I iterate four times as often and can save a width value of half the iteration number, which never equals a subsequent iteration's index.

A clever manipulation of state bits is required to avoid confusing these width-stored pels with the active object, particularly between the first and second iterations, when the nascent medial lines will be indistinguishable from the actively ablated object. Because of this, I have chosen to implement only one scan per iteration so that the iteration number can always remain higher than any recorded quench function value. As a consequence, fast Montoto thinning as presented here can only thin objects less than 128 pels in width, before all saved pels must have their iteration values converted to width[1]. On the other hand, FMT can be implemented with only a single state bit overlaying data, and depends only on the most basic configuration of a vision engine to be implementable as a sequential homotopic thinning. The accuracy of the quench function is only comparable to Bel-Lan and Montoto's thinning-1, but the homotopy is identical, with a modification to allow four-way intersections of thinned lines to join the medial line set. In fact,

[1] Assuming an eight-bit image buffer

with adjustments to the medial line set, many different thinning algorithms can be performed by FMT. Only the look-up table masks need be changed.

The change in the algorithm of [2] is mostly for convenience in parallel implementation while preserving the valuable homotopic character of the transformation. Overall, I have made two major changes to the published algorithm, both made for the sake of parallel implementability in the fast algorithm. The definition of the quench function values that are preserved in the image are changed, and as a consequence of my elimination of the 1×4 neighborhood test, all morphological operations can run on 8 bit data and 1 bit state with the 3×3 information contained in a simple neighbor coding of the original segmented object, at the price of some added sensitivity to single-pel protrusions from object boundaries. I have found that a pair of NC images can be computed in parallel and contain all data needed to satisfy both the requirements of the hit or miss transformation that defines ablation according to [2, eqns.4–7], as well as the requirements of the thinning-2 test, [2, eqns.2′-3′].

Whether measuring an inscribed square or other shape, the quench function that results from fast Montoto thinning is a meaningful substitute for other types of pore throat width measurements. The value of the quench function in the FMT presented here gives twice the size of the pore's inscribed square width, while its position along the medial axis gives the location of the pore throat's width at that particular point. The essential consequence of this is that fast Montoto thinning provides information sought after by pore throat size distributions that are computed indirectly by successive erosions and object counting at every iteration. By use of a more advanced morphological process, homotopic sequential thinning rather than sequential erosion, it is possible to measure in one nonstop processing sequence what is done in a much more roundabout way by the morphological aspects of petrographic image analysis.

7.2 Unopened Thinning

A by-product of the fast Montoto thinning algorithm is the total object area remaining at various cycles of ablation, or remaining area curve. The measure of the remaining active feature area is computed at every step in the iterative thinning operation as a test to see if the algorithm has finished. When the remaining feature area is zero, the algorithm has finished.

At every iteration, a histogram count of pels in the active area is recorded. The active area defines the interior of the object for the next iteration and starts with the area of the original segmented feature. These values monotonically decrease to zero. The difference between remaining active area between FMT cycles measures the area that is ablated at every iteration in the sequence, whether or not it is preserved as a quench function value. Also, for any two-point moving average of the quench function histogram, one can obtain that area in the original image that was located at distance (1/2 the quench function value $\times$pel side), from the edge of the original objects.

When considering the remaining area curve, a change in slope between ranges of quench function values can discriminate between the ablation of objects with different boundary roughness. Large convex objects ablate much more slowly with sequential thinning than do elongated or dendritic objects, which are eliminated much earlier in the sequential ablation process. These regions can be used to discriminate between topological classes of features of similar sizes. Original object can be segregated by size using conditional dilation, also called particle extraction, as discussed on page 144 This distribution differs in meaning from the histogram of the quench function values in the transformed result, because in general only a few of the pels identified at each step in sequential thinning belong to the medial line set. The others are eliminated, by zeroing, at every iteration. Both the remaining area curve and quench function histograms are useful extracts, neither of which requires curve vectorization.

Now I will turn to some examples of images of varying topological complexity to show a means of presenting thinned results. Figure 7.1, shows two natural shapes, an oak and a bignonia vine leaf from the Stanford area. By placing these on a copy stand with an industrial black and white video camera, the leaves on a white paper background were easily segmented into binary objects. Figure 7.2 was prepared in a similar way. I used it as a benchmark since it was also used in Bel-Lan and Montoto's paper as a benchmark at similar resolution. It is [2, Fig. 7]. Figure 7.3 is a template used in the production of a physical model of multiphase fluid flow. This approach was used by Prof. F. Orr [1] and also by J. Trygstad et al. [52] in fluid flow studies. In both cases, a rock pore space image, with connectivity improved *by hand* to permit percolation in the image, was etched in glass and bonded to a cover plate. The connectivity of these images, derived from pore space data, was improved manually by drawing connections between well-oriented pores on the template before etching into glass. Figure 7.4 is a borehole televiewer image provided by Colleen Barton, of Stanford's Rock and Borehole geophysics program, the image is from an SRB televiewer scanning of the Cajon Pass scientific drilling project. The image shows a fracture intersecting the borehole. I will describe the segmentation procedure later, but the fracture object has been thinned to schematically represent the fracture as a processing step in a machine vision approach to automated fracture angle detection in borehole televiewer data, which can be voluminous. The last example, Figure 7.5 is a thin section digital micrograph of the Kern River formation tar sand member, from Bakersfield, CA. The prepared section was provided through the courtesy of Prof. S.A. Graham of the departments of Geology, Geophysics, and Applied Earth Sciences at Stanford.

In Figure 7.1, the two leaves are symbolic of simple convex pores and more elongated or dendritic pores (no pun intended). In the thinned result, note how the mostly convex leaf, the larger bignonia-shaped one, has only a single arc of quench function values running along its axis of symmetry. Strictly speaking, this

is a simple, untoothed leaf [49] from a Violet bignonia vine in Woodside, CA. The quench function is only nonzero along the leaf's medial axis line and starts at the very tip of the leaf from a single pel projection at the very sharp end pointed left in the original image. The quench function values along this single line steadily increase to the right until the axis starts representing the stem of the leaf, where they rapidly become small.

The more complicated shape is an oak leaf from Woodside, CA, technically a lobed simple leaf [49] from the Valley oak, also called the California White oak [49, pp 404–405]. The thinned result shows a series of branches extending from its main axis or midvein that also extends out the leaf's stem. Although the medial axis is very close the the natural midvein in the oak leaf, its position is only a function of the boundaries of the leaf's lobes, and no image information was used to condition the position of the thinned result to a visible midvein.

In the results shown in Figure 7.1, the original image is plotted in the halftoned window in the upper right hand corner and the segmented version, which is operated on by the thinning, is shown below it. To the left of the original input image is its histogram of brightness values, ranging over the eight bit range 0–255. I adjusted the programmable gain and bias or *black level* on the digitizer's analog to digital converter unit, ADC, which is also called the analog subsystem. This made many of the pels in the image either 0 *or* 255, so these two very populated points in the histogram have been clipped.

A priori, images that are to be segmented by brightness levels are better segmented when object levels differ greatly from background levels, so I increased contrast by programming a lower analog to digital converter's gain and slightly higher ADC bias before digitizing. The connection from theory to practice is operationally very simple, because with an interactive program module from Recognition Technology, Inc. [42], ADC gain is adjusted with left-right mouse motion and ADC bias is set by up-down motion, while viewing the vision engine's monitor. Since the

analog video signal from the camera is a perfect match for the digital data rate of the synchronous channels inside the engine, every frame is digitized and available in succession, in real time as it arrives from the video camera. So an interactive adjustment of programmable ADC parameters is updated with effects displayed over the entire image 30 frames per second.

To the left of the segmented image window in Figure 7.1, I plot the histogram showing both leaves as brightness values in the thinned result, which is shown in a larger image window below its histogram. The two histograms are nonlinearly related through segmentation and thinning. The larger plot in the lower left part of the figure shows the remaining feature area. This is the size of all objects remaining, after the number of iterations of sequential ablation shown in the bottom axis. Starting from the area of all objects in the segmented input image at iteration 0, cumulative remaining active area decreases toward an area of 0 at the final iteration. After the thinning has progressed some 100 iterations, the Valley oak leaf has been reduced to its medial line representation and makes no contribution to the active feature. The bignonia leaf remains since its convexity makes pels near its center over twice as far from a boundary as any pels in the oak leaf. It slowly decreases in area up to the 248th iteration. On the cumulative remaining area curve, notice a small reduction in slope at the point where the oak leaf finished thinning, after which ablation removed pels from the only remaining object, the bignonia leaf.

If one were to filter the thinned image based on an heuristic from the lower histogram, all values above 80 could be saved and sent to Montoto thickening. This could recover the larger parts of the Bignonia leaf, and none of the oak. This is an example of feature segmentation through a thinning and thickening cycle.

In the quench function histogram from the thinned result, the bignonia leaf is spread throughout the distribution, with a few quench function values across the range [1–248]. The oak leaf's lobes all have similar widths and created the peaks in the ranges 20–50 of the quench function histogram. The implication of this is

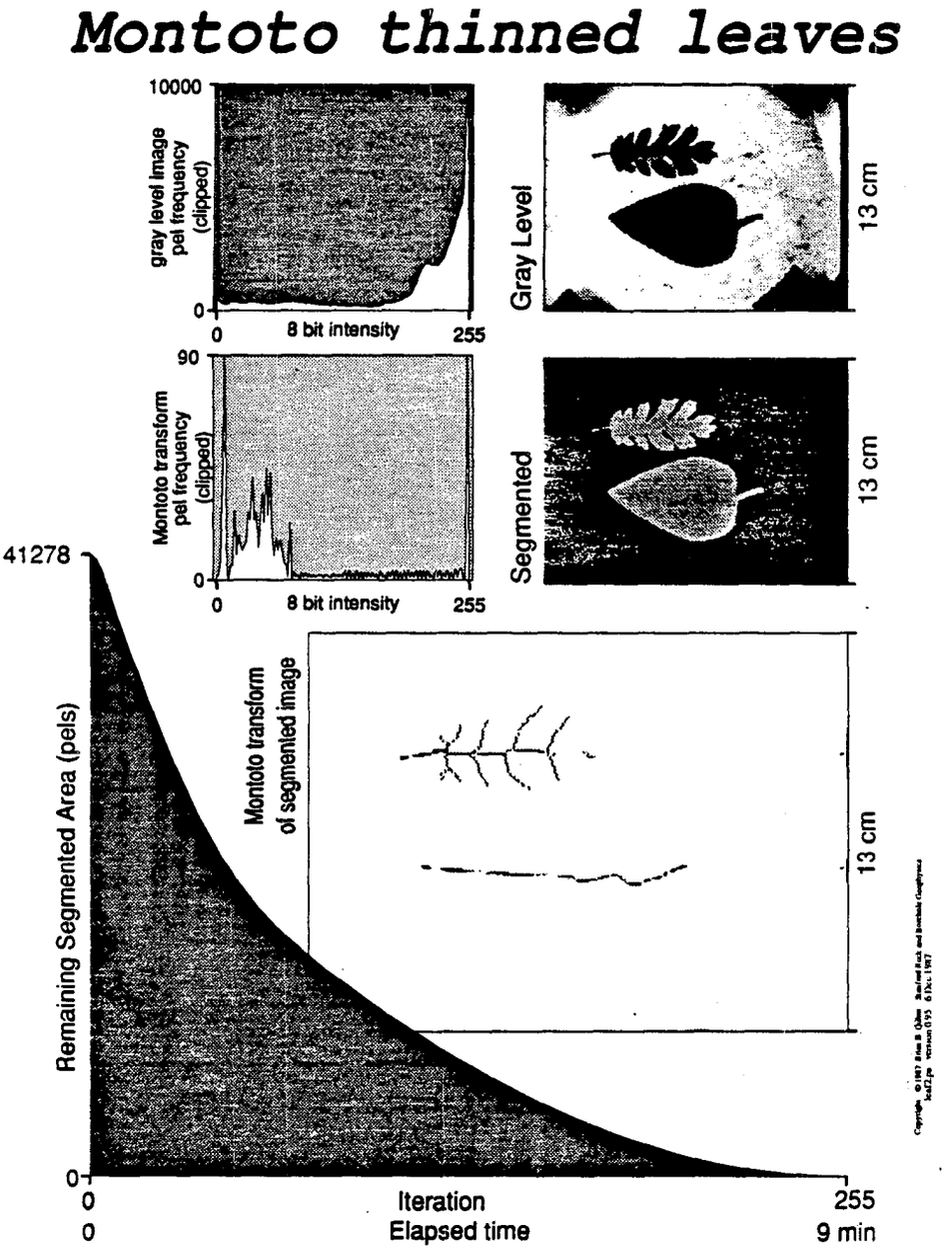


Figure 7.1: This figure illustrates a use of homotopic sequential thinning to distinguish differences in topological complexity. The Violet bignonia leaf is larger and nearly convex, while the Valley oak leaf has deep and sometimes notched lobes. The apparent disconnection of the oak leaf's thin arcs is due to the halftoning process used to create this figure, rather than artifacts from the FMT process itself.

that through the histogram types of analysis plotted in Figure 7.1 and the others like it, it is possible to numerically discriminate between similarly sized objects with differing topological complexity.

The fracture system in Figure 7.2 is derived from fluorescin impregnation of fractures in granite studied in [2] and represents a well connected fracture system that was used in Bel-Lan and Montoto's computations of thinning benchmarks. The same types of analyses are presented as in Figure 7.1. However, since the fracture system is narrower and better connected than the leaves of Figure 7.1, the thinned result is computed more quickly, in 1.3 minutes instead of 9, as shown along the bottom of the remaining area curves in both figures. The plotting of the thinned result is a halftoned representation of the quench function values that is less than ideal because the connectivity of the quench functions appear to have been broken by thinning, but actually only appear broken by the halftoning. In fact, the connectivity of the original image was very well preserved, but the pel-wide lines can not be plotted properly without vector encoding and plotting, which was not used in this plot.

The segmentation has been selected to honor the outline expressed in the paper, and the image has been scaled so that the segmented object area is at least as large as the area in their digital benchmark. They used a 400×500 gridding of this fracture system scaled to cover 14,819 pels [2, p.44]. I use a 480×512 gridding where the segmented object covered over 16,000 pels.

In Figure 7.3, the pore space that was used in Joyce Trygstad's SPE paper [52, Fig. 3] has been thinned to produce a quench function array that can be represented as an equivalent network. The thinned results were computed in 45 seconds of processor time, although at considerably lower resolution than represented by her composite photo-template with respect to the original digital images obtained with PIA. The quench function values disposed throughout the thinned result are sufficient to define an equivalent numerical network representation. Since most of

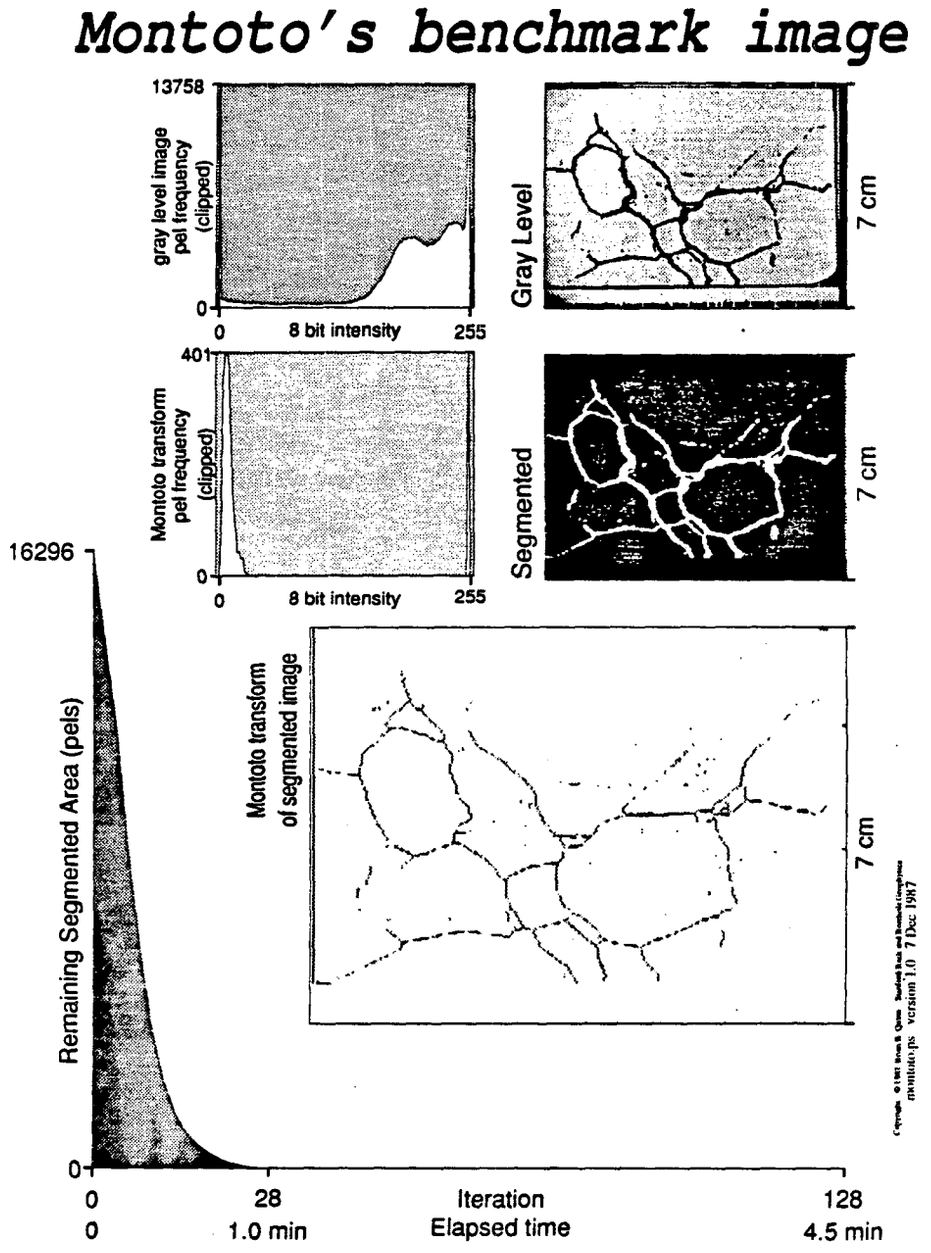


Figure 7.2: In this figure, from Bel-Lan and Montoto's Fig. 7, a fluorescin impregnated fracture in granite has been imaged.

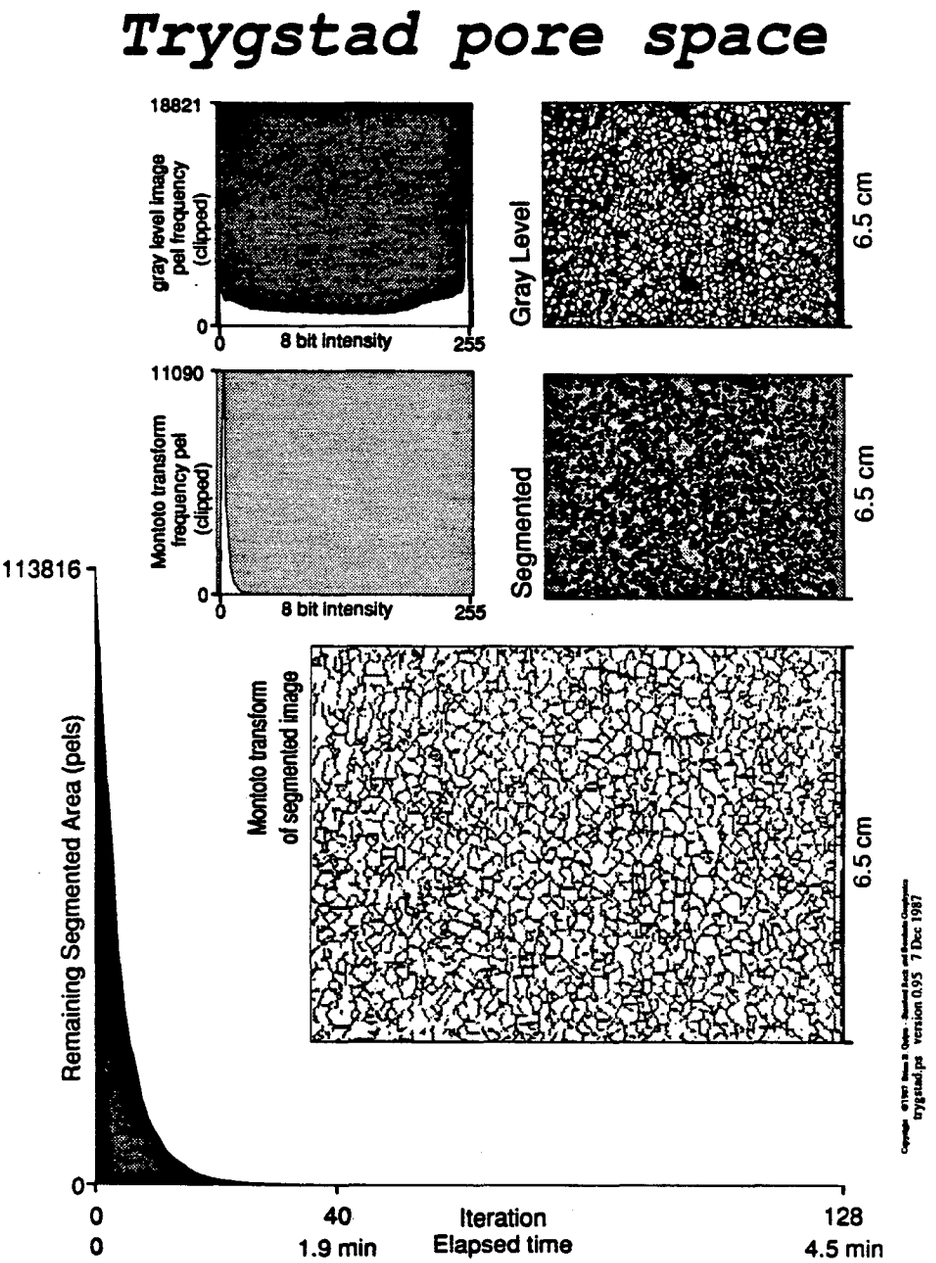


Figure 7.3: In this figure, from Trygstad et al. Figure 3, a composite of digital images from 12 thin sections collected by PIA has been sampled at 512×480, segmented and thinned.

the lines in this image were already thin ones drawn by hand to create a template for a physical model, quench function values are skewed toward smaller values.

Figure 7.4 shows an example of thinning applied to a borehole televiewer image from Colleen Barton of Stanford's Rock and Borehole Geophysics Laboratory. In the image, the televiewer data has been digitized by Colleen and processed on her system from raw analog data into raster images of acoustic reflection time and amplitude. She separated the amplitude image, justified the variable length scans into a standard 512 element length, and edited a particular run of 481 scan lines from the continuous coverage of the televiewer record. Finally, she wrote the justified array into a file of 512×481 `unsigned char` variables. I copied that file from her machine to the image analysis system over the campus Ethernet, segmenting and thinning it there.

Since pels in Figure 7.4 represent amplitudes of the televiewer signal received, in this image the fractures in the borehole walls are the features that are not perpendicular to the televiewer's acoustic transducer. These rough fractures scatter relatively greater amounts of acoustic energy than the smooth walls. Thus, the segmentation selects those low-amplitude components of the image, and in fact was a threshold of the lowest 20% of the amplitude values, that is, pels in the range [0–50]. Thinning reduces the fracture objects to a medial line representation that can be vectorized to estimate the height of the sinuous shape in the thinned result. This sinuousity is a product of the unwrapped cylindrical coordinates used in the construction of the image, and the intersection of planar features with that cylindrical space. Steeper fractures appear as sinusoids of greater height, and horizontal ones are sinusoids of zero height.

As homotopic thinning linked to line following, the FMT-CSC approach defines a machine vision algorithm to automatically identify fracture azimuth and angles, corresponding to geological strike and dip over televiewer logged intervals. Because these fractures in granite scatter more of the reflected energy than smooth walls, a

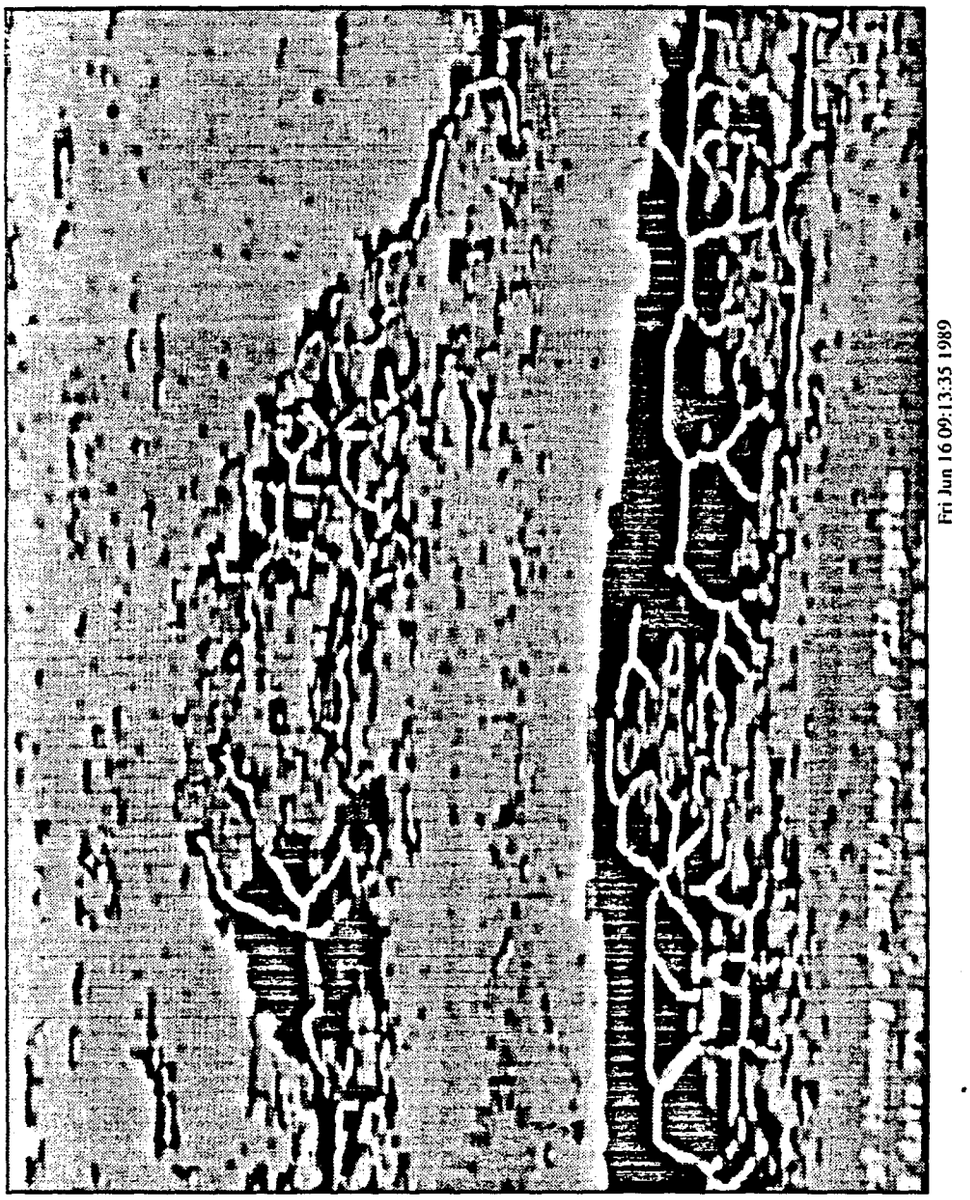


Figure 7.4: Here, digitized borehole televiewer amplitude data has been justified and rasterized by Colleen Barton of SRB, then segmented and thinned.

simple segmentation by thresholding has identified the openings of fractures in this amplitude data. The thinned representation's medial axis lines can be used with line following to estimate the sinusoid that is implied by the fracture aperture, even if the aperture varies widely across the image. With line following, this procedure would be an important part of a machine vision algorithm to automatically log fracture angle, azimuth, and aperture with depth, from borehole televiewer data.

The last example in this section, Figure 7.5, shows a digital micrograph of the Kern River formation tar sand from Bakersfield, California. The thin section was provided by Prof. S.A. Graham at Stanford. It was prepared with blue-dyed epoxy impregnation following hydrocarbon extraction. Optical filtering of the 400X image with red and blue dichroic filters produced a digital image pair that were ratioed, red-filtered into blue-filtered image. High blue/red ratios were used to segment the pore space from the rock.

For now, the plot of quench function histogram indicates the pore throat size distribution in this 400X image contains no objects wider than 10μ. The axis of the quench function histogram in this figure has been changed to indicate an implied size distribution that relates one pel width to 1/2 the quench function value. Red and blue dichroic filters were used to collect a pair of digital images that were ratioed and thresholded above blue/red ratios of 2:1 to produce the segmented image. Thinning reduces this pore space to an almost interconnected path around the weathered mica fragment in the center of the image. The quench function histogram represents a pore width distribution.

The width values plotted in these figures are based on the maximally inscribed square approach to width estimation as presented by, though not identified as such by Bel-Lan and Montoto [2]. That paper is the basis for my approach, although in the parallel algorithm I present here is distinct from it. The thinning error measure described by [2, eqn. 10] still applies:

$$E = \frac{S - \sum(w_{i,j}+1)/2}{S} \times 100 \qquad \text{(percent)} \tag{7.1}$$

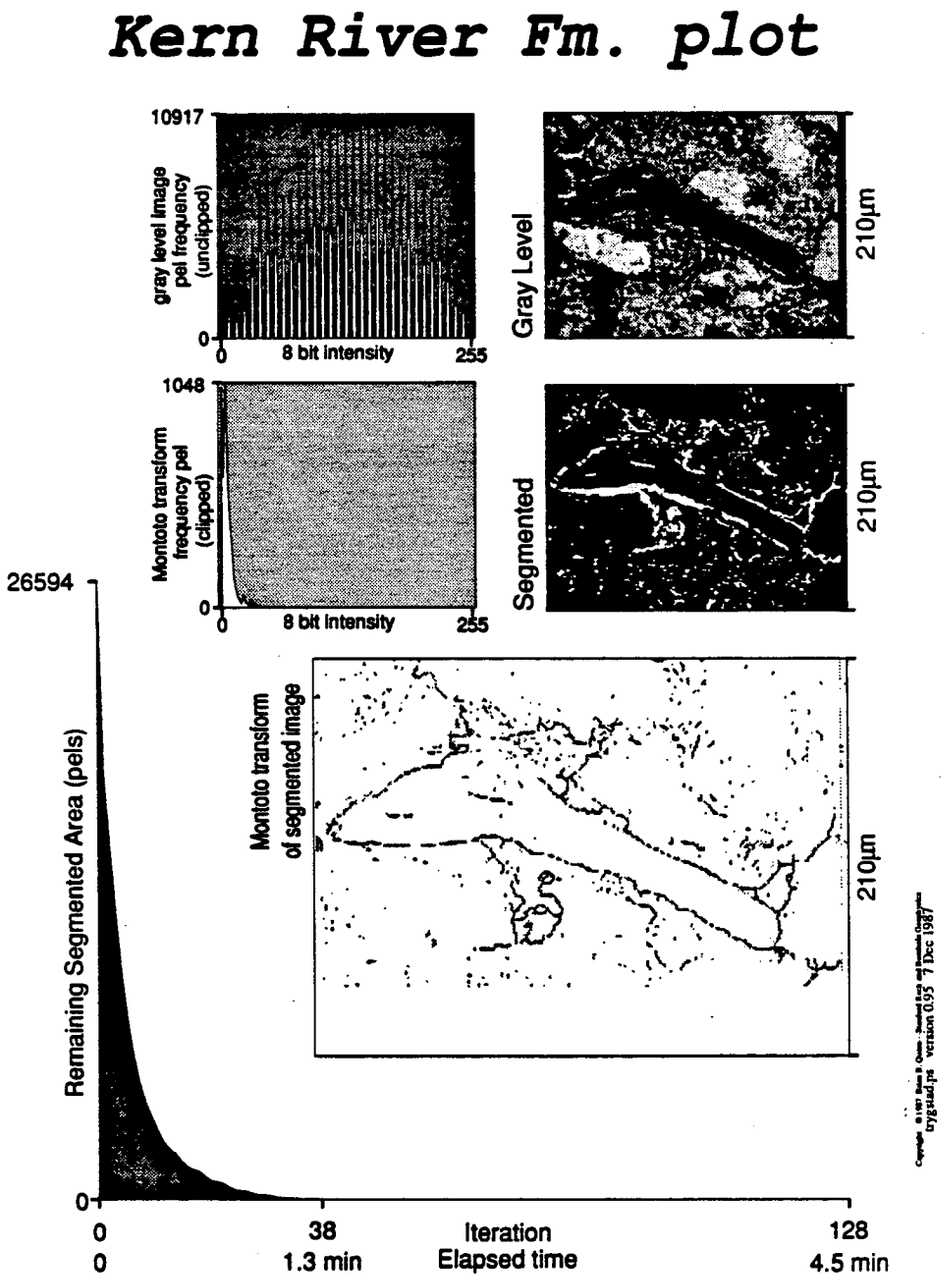


Figure 7.5: This 400X digital micrograph details a mica fragment nearly surrounded by pore space in the Kern River formation in a thin section provided by Prof. S.A. Graham of Stanford.

Where S is the segmented area of features in the input to thinning. The width values $w_{i,j}$ throughout the thinned result are summed. Since I have used a simpler sequence with one ablation each iteration, an alternative width measure appropriate to fast Montoto thinning is based on $^1/_2$ the iteration value computed from an ablation sequence in the NEWS directions.

7.3 Collinear Segment Channelization

To make use of the quench function values disposed throughout the image as an input to a network model, even with perfectly derived medial axes, the thinned curves present a problem. This problem, recognized by Orr [1] and Trygstad et al. [52], is that the two dimensional information obtained in thin section does not contain enough information to image the connectivity of pore space which must exist in three dimensions. For example, a thin section in which the pore space has been filled with blue epoxy will have a two dimensional lack of percolation from one side of the image to another, while it is apparent that because the blue dyed epoxy found its way into every imaged pore, now blue in the thin section, there must be some path connecting these pores, although that path is outside of the view of the thin section.

Other authors, like Ehrlich [17], argue that there is sufficient information in the thin section to make good estimates of bulk permeability based on pore space *texture*. If this is the case, then it should be possible to morphologically enhance the connectivity of the pores observed in two dimensions, as much as needed to make the connections that are otherwise made by hand when creating a two dimensional etched glass physical model representative of pore space, particularly those derived from digital images [52, 1, 5]. From the range of morphological processes available, it should be possible to find an operation, or a hybrid of morphology and statistics [47] that will be able to adequately perform this connection between pores. It will produce a representative percolating model in two dimensions automatically from a pore space image and thereby enable a network model to be derived from such an image. This modified system of quench functions could be studied numerically for flow properties in a variety of ways.

Among the available morphological tools, there are two types of approaches to automatically improving the pore to pore connections to create representative two dimensional percolation. One type uses the array or image itself, performing

morphological operations on the image before and after the thinning is performed. The other type involves summarizing the vectorized thinned curves which contain the quench function values. The latter approaches are more in agreement with the fundamental principles of morphological processing. Within this class are two approaches. One approach, alluded to in Serra [47, p.422], is an application of universal kriging for linear interpolation between disconnected segments in a thinned result. The other approach makes use of a parabolic segment curve fitting routine that can bridge gaps using "stepping stone" pels between larger connected lines. These stepping stones are small isolated pore features that may sometimes be found in-between the disconnected throats of larger pores. The curve fitting approach is presented in Bookstein [6].

7.3.1 Homotopic Closing

The simplest approach to improving the percolation of thinned results is a pre-processing of the segmented image by semi-homotopic morphological closing. This is a global way of smoothing outside borders by dilation, even making connections to neighboring pore elements or porels [18] by means of a simple-minded concept of nearness based on bridging some of the gaps between porels. Then, rather than finishing this closing with erosion, which would disconnect all merged porels except those with more parallel faces of lower curvature, homotopic *thinning* replaces erosion to preserve all connections between porels where the closer parts of porel borders have merged.

Semantically, the semi-homotopic adjective is intended to convey that one half, specifically the second half, of this sequential processing sequence preserves all possible topological measures by using homotopic thinning rather than sequential erosion. What I think is really needed is a better second half to semi-homotopic morphological closing than just plain Montoto thinning. I have identified a broader, alternate set of medial axis templates that should be used as part of this pre-processing step.

They are identified in an appendix.

Pre-closing or pre-dilating an image before thinning is a very simple means by which pores in the image that have a small gap between them can be made to coalesce and be represented in a connected way in the thinned result. Unfortunately, this simple approach is at odds with a fundamental premise of mathematical morphology, that is the *regular model.* The class of topological sets for which certain types of morphological operations are valid is called the regular model. The regular model, often invoked when morphological operations are used to study an image's connectivity, does not permit these coalescings of porels to remain after closing by some structuring set [47, p.373]. The size of that structuring set is often elementary, convex and as small as possible, the set H. As a mathematical tool, the regular model allows one to specify quite explicitly which Euclidean images, $\mathcal{R}^2$ or continuous space, are represented sufficiently well in their digital versions to perform computations that say something about these continuous spaces. The regular model requires that the objects in the image be sufficiently well delineated along their near edges so that opening and closing of the image by a primitive element, like H, will not yield different results. Almost certainly, if these two results are identical, they will match the original image as well.

When dilation bridges a gap between two neighboring pores, the erosion which follows it in morphological closing might not rebreak that gap and certainly could result in a different set than would result from opening, where first erosion and then dilation is performed. The morphological opening will eliminate features that are already thin in the original image. Closing also will fill in small holes in the midst of larger objects. This could be a significant change in its own way. When considered as pre-processing for a homotopic thinning, pre-dilation of a segmented image changes, sometimes significantly, the connections or homotopy between the various pores in the image. It provides as input to thinning an image that is no longer a part of the regular model representation of the original pores. Any mathematical description

that justifies the thinned result as a representation of an image's heterogeneity would disallow this particular preprocessing sequence [47, p.144–146].

Nonetheless, as a means for estimating what is not directly shown, only implied in the original image, pre-closing is remarkably efficient and easy to implement in parallel. If it is not close enough to the original image's topology, especially when reconstruction of the original from the thinned image is desired, then it is important to keep within the regular model for a processing sequence used to describe the image directly. However, when an indirect measurement of the image is to be made, particularly one that requires a trifling bit of percolation to characterize the image's flow topology numerically, this simple technique could be used as a preliminary test. As a test for whether an application of kriging or any other routine that uses the more expensive vectorizing step is justified, this is probably a very good approach.

With the preclosing operation, one operates on an image that has been removed from the regular model, if it belonged in it to begin with, but several benefits can be realized all the same. In semi-homotopic preclosing, neighboring porels that merged remain connected, because the dilation is followed by homotopic thinning rather than erosion. The roughness of the surface of objects in the original image is smoothed by many of the structuring elements that might be used in the dilation process. This can eliminate single pel "hairy" protrusions [45, p.240] in the borders of objects which are known to introduce spiky members, stray branches, bones, or any number of like maladies into the skeleton. By ridding the skeleton of these, semi-homotopic preclosing simplifies the topology of the thinned result, and requires only minor modifications to the processing sequence of thinning. If an image analysis system thins, it will surely have dilation utilities available as well. While mathematically disallowed by the regular model, this pragmatic approach to rapidly calculating simpler, more connected networks from an input image does have a place in connectivity studies.

7.3.2 Polygon Coding Of Arcs

Another alternative to the approach of channelizing or connecting collinear segments of thinned results is a characterization of quench functions. This involves a polygon coding of thinned curves, which is not a raster operation. When speaking of curves and polygons, one speaks in terms of vector elements, the sort of things that would be plotted by graphics routines. FMT processed images can be converted automatically to this form with my CSC algorithm.

The polygon forming procedure is performed in two steps. First CSC creates lists describing the FMT image contents, then run-length encoding, or moving window averages of the directions along thinned arcs create a polygonal approximation to the original arc [22, 39]. Spatially, this process involves into measuring different *covering* representations of the quench curves, following them with coarser and coarser structuring elements as longer and longer moving windows are averaged along their length. Since a chain code representation of these curves is a directional coding, from one neighbor to the next along a line or border, a run length encoding effectively reduces the number of vertices in a polygonal shape.

When several of these moving windows are used for each of the curves that have been vectorized, the shape is represented in terms of chain code direction trends. An estimate of each trend, from the endpoints of each curve toward its midpoint, can be used to extrapolate linear extensions from the endpoints of the lines onward, which can then be rendered in the raster space. When all such extrapolations are plotted for all endpoints, those that are well oriented may cross each other, and in this way, the connectivity of well-oriented or collinear curve segments of quench functions can be used to recognize where such connections are likely to exist in the third dimension, outside the thin section's plane of observation.

If these extrapolations should overlap and cross one another, a morphological pruning approach, as is used in the calculation of a skiz, referred to on page 138, can be used to again simplify the topology of the now better-connected network.

Using the morphological procedure symbolized in the expressions of that section, one is able to perform what is sometimes referred to in graphics as clipping, where the lines cross, and not continuing either extrapolated segment past that point.

7.3.3 Parabolic Curve Fitting

An extension of the approach just described, which was based on universal kriging, may be useful when studying the connectivity of pores in sedimentary systems with well rounded grains, where pores often connect around grains. In these cases, particularly where a single, isolated pel or block of pels provides a stepping stone between two larger pore chambers on either side of a grain contact, an approach suggested by Bookstein [6] may be used to extend this to a quadratic, specifically parabolic representation. This is equivalent to expressing parabolic segments of thinned lines. In this way, the procedure interpolates parabolically between endpoints of quench function curves, and then clips connections as before. To make use of these so-called stepping stones, information from not-so near neighbors must be used. If identified in the course of a parallel process, this search for stepping stones could be morphologically implemented. That is to say that a family of near neighbor mappings in, say, a hexagonal region about each point, can be defined in terms of a hit or miss transformation (which is global), and then consolidated into some number of bit slices.

Extended look regions for use in nearest neighbor mapping when searching for the stepping stones that define a parabolic fit are morphological versions of the grid that is used in old fashioned topographic corrections in gravity surveys. Those templates were the kernel of a linear spatial convolution of mean topography. Within a set of circular shell segments, each shell segment is weighted according to its r^{-2} contribution to the gravity at the center of the convolving mask, the point for which the topographic correction is made. The coefficient for each cell within a circular shell is weighted by the angle subtended by the cell. That sort of weighting can

be based on the continuous *influence function* of which the mask is a discretized approximation. The weighting function w_i for gravity mask cell i has the form:

$$w_i = \int_{r_0}^{r_1}\int_{\theta_0}^{\theta_1} r^{-2}\theta \, dr \, d\theta \tag{7.2}$$

Where the cell is in the shell between radii r_0 and r_1, and the angle subtended by the cell at that range is from θ_0 to θ_1. These weights are used in the topographical correction to gravity field at a point. In $\mathcal{R}^2$, the topographical correction template is drafted and printed onto mylar, which is then overlain atop a topographic map and held in place. A completely analogous digital convolution could be applied in $\mathcal{Z}^2$ to find Bookstein stepping stones [6].

At this time there appear to be three opportunities for improving the connectivity of two dimensional networks of quench functions derived from pore space images by a thinning transformation. The first method, semi-homotopic closing, is adequate for improving the qualitative connectivity for preliminary studies and as a test for the advantages of using better but more expensive connectivity enhancements for a given image. The second and third methods are applications of universal kriging in linear and parabolic applications of unbiased interpolation. Universal kriging is a more sophisticated approach to finding subtle relational properties between porels in extracted lists of arcs. These more sophisticated approaches should be studied in the future.

7.4 Global Thinned Arc Analyses

In the context of this chapter's discussion of the consequences and verification of fast Montoto thinning, this section will describe one major class of analyses delineated by the type of processor can calculate these measures. Namely, measures that are derived from histogramming and cross plotting of intermediate results of sequential homotopic thinning. One commonly used process is the histogramming of brightness distributions in a gray level image. Another less common type makes use of the areas in intermediate results of sequential thinning. The third is an alternative approach, a hybrid between these raster histogramming processes and those in the next section, which are based on curve vectorization. Also, morphological processes like hit or miss transformations with the structuring set $\{E\}$, to identify endpoints in thinned results, or with structuring set $\{F\}$, to identify nodes, can be used as input to a histogramming technique. Also, local configuration or neighbor coded input images can be used with histogramming to find populations of topological classes in the image, like edges, thin points, interior points, endpoints, or a variety of nodes.

7.4.1 Segmentation Problems

Histogramming is a popular approach to the characterization of gray level digital images. By summing the frequency of occurrence of each brightness value, a concise summary of a huge number of image elements can be represented in a simple function. At present the approach I use to calculate these histograms with a vision engine is a relatively slow, nine step process. The code takes most of a second to run on a 512×480 image with a PC-AT host. Special purpose hardware is available from some vision engine manufacturers at this time for this problem, the device is a histogramming co-processor unit within the array processor system. These make the histograms available within 1/30 second from the time the raster image data starts to arrive. This result is much more useful because it is available synchronously, at the same time that the processed results become available.

When histogramming the original gray level image, many types of segmentation can be derived from the distributions. Histograms may be multidimensional for multiband images. When applying histogramming to the problem of image segmentation, I usually avoid cross plotting since I not not have commercial software for this purpose that is compatible with the rest of my system. My way around this is to segment from pie slices of cross-plots, using a one-dimensional segmentation of the ratio between two spectral bands, like dichroically separated red and blue bands. Thus my classification of pores is based on a range of values in the gray level result that is a ratio between two spectral bands.

A better way to work this classification scheme, with appropriate software, is a segmentation defined on cross-plots of two or more spectral bands' brightness distributions [32]. Cross-plot is a term for a scattergram of co-occurrences of spectral brightness in multiband image elements. Then stochastic classification can automatically delineate blocks for certain features [35, 34, pp 461–470]. Even better, an interactive system can be built to do this classification. Viewing the cross plot on the screen, a user can define *polygonal* closed regions of the cross plot, and view the effects of this segmentation in another window. Alternately, selected feature objects a priori may be pointed to on a display of the original or enhanced image, and regions in the cross plot can be marked manually in this indirect way. This back picking and polygonal cross plot segmentation has been implemented in an image analysis system by Rush Record of Everest Technologies, Houston.

The polygonal regions in the cross plot that contain the desired relation between two spectral bands can be used interactively to experience dramatically the improved segmentation. The one dimensional segmentation of ratioed images that I use is a small subset of the segmentations possible with a general cross plot segmentation routine. I segment on pie shaped slices of the cross plot regions, where the sharp end of the pie points at the origin of the cross plot.

And so histogramming analyses based on the original raster image are mostly

useful for segmentation. Since segmentation of a gray level image to an indicator function is a necessary precursor to fast Montoto thinning, I mention some of the choices here. A segmentation which can identify more details in the pore set will necessarily improve the representation of the pore space in a thinned result.

7.4.2 Intermediate Result Histograms

Histogramming may also be used to study the gray level distribution in a thinned result, such as the quench function distribution from Montoto thinning of the indicator. A simple plot of the quench function distribution yields the size distribution of maximally inscribed blocks. Thus, measures of pore throat size distributions can be based on these histograms. More specifically, the pore throat size distribution of Ehrlich [17] can be derived from a segmentation of quench functions in the thinned result in a cross plot against a *gray level erosion* of the quench function image. In gray level erosion [45, 47, p.432] the local minima are enlarged.

In this way, only those quench function values that stretched off the $x = y$ axis of the cross plot, extending horizontally if the gray level eroded quench function distribution is plotted on the x axis, would be local minima of the quench functions. This is a sufficient condition to identify those isthmi in porels that would break and be detected by boundary coding at some cycle of sequential erosion. This could produce *exactly* the pore throat size distributions of PIA, but computed entirely in global, parallel-implementable operations, plus histogramming. These are exactly the types of morphological codes I set out to adapt to vision engines.

7.4.3 Orientation Histogramming

When histogramming neighbor coded image data, the histogram contains a wealth of information, although it is usually not in the familiar form of smooth distributions that often result from histogramming pictures, or images with smooth gray level transitions. This nonlinear histogram is a valuable concept for the NC-based operations, since the histogram can be used to group pels into classes of morpholog-

ically similar groups, border points facing certain directions, *simple* points, which would be eliminated in an iteration of thinning [45, p.232], or interior points. This information might be used as a conditional data set when designing *adaptive* sequential thinning algorithms, which could ablate from the directions that will most rapidly thin the image.

In other cases, iterative classification schemes may be applied, based on Rosenfeld's segmentation based on successive relaxation. This, too, is an adaptive filter conditioned by statistics of distributions. Specifically, those co-occurrence distributions are used in an neighbor code compatible image clustering scheme to iteratively increment or decrement a classification probability in a discrete range between 0–1[45, p.154–158].

When thinking about what histograms of NC data mean, consider those elements of an image object that are surrounded on all eight sides in a rectangular grid by pels that are also members of the object. In this case, solid objects are filled with `0xff` or 255-valued pels. All eight neighbors are object, so all eight bits are set to one in the NC, which equals hexadecimal `0xff`.

In homotopic thinning like Montoto thinning, the histogram of interior points that remains at every point in the sequence can be used to calculate a difference in cumulative area from one iteration to the next. This is particularly useful because the histogram points are a by-product of the sequential thinning operation. The count of remaining active image elements is used to check whether the synchronous part of the FMT has finished its work. Details of porel surface areas or perimeters that are present at different points in a sequential thinning are accessible only by encoding the boundaries of the porels at every intermediate step. This coding approach is a natural one for a sequentially structured morphological computer program.

When one must make a distinction between where in the system the global (parallel) parts of the code will run, where the local (sequential) parts will run,

and how much time it takes for the processors to communicate raster data between each other, one stops boundary coding too often. In some sense, my interest in morphological programming of vision engines has precluded my duplication of PIA's techniques. From the historical context of PIA's development in the early 1980's, it is a sequential processing oriented approach to rock image analysis. At present, we have more powerful image analysis hardware and more powerful approaches, like thinning, become more desirable.

When running morphological filtering on general purpose machines, a single data cache for passes through an image file is sufficient, and raster-vectorizing code at the tail end of the processing sequence may have its data already loaded and waiting, for array processors are ideally suited to the calculation of neighbor codes.

Because pore throat size distributions are valuable, and since institutions without PIA are not often provided with such code, I have developed my own boundary follower, CSC.

7.5 Arc Vectorization Analyses

Many of the histogramming types of analyses, however interesting, are somewhat limited in their utility when relating the results of a sequential thinning of an image to interesting physical properties. Vectorization, the second type of analysis, is much more powerful than global operations in this regard. In fact, with a curve vectorizing routine, like Cederberg-Sobel compilation, thinning of objects in a digital image of a rock could be analyzed in every way now used in petrographic image analysis. When curve vectorization is linked with a parallel machine running sequential thinning code, particularly one which produces a quench function, the results of thinning can be used to measure throat size distributions, and additional information is available from the position of quench function values in the image. CSC is a separate processing procedure from the homotopic thinning of the fast Montoto transform.

Adapting the curve vectorization to the specific needs of rock images and rock property databases in general, the automatic formatting and generation of network model databases from quench function values is a problem unto itself. Many interesting analyses are the result of a combined sequential thinning and curve vectorization. I discuss these here, so that they can be contrasted with the types of analyses that are obtained from histogramming of the raster data alone.

7.5.1 Boundary Coding Of Everything

The basis of many types of texture analysis in petrographic image analysis has been published by Ehrlich et al. [17, 19, 24, 25, 3]. It involves the vectorization of the boundary of porels in the original image. This boundary coding of segmented and morphologically transformed results is a powerful way of measuring textural statistics. From the original image, it is possible to derive the aspect ratio of objects, their true maximum and minimum diameters, as well as their shadow, or ferret (hard core) diameters. The orientation of objects in an image can be determined, as well as the isolated object count, of course. The relationship between the boundary code

of an object and its area has been known for some time [23].

When boundary coding intermediate results of sequential processing, one may study the changes in connections which occur following basic morphological operations like erosion and dilation. Those connections which occur between pores as a result of dilation will be expressed in a lower isolated object count. Likewise, objects which break apart during successive erosion cycles define, through vectorization, places where pore throat isthmi have eroded through. This is an important basis of some petrographic image analysis measurements.

From a modification of CSC's topological mapping and an appropriate neighbor coding, it is possible to measure surface roughness or polygon boundaries with crack codes automatically extracted around the boundary pels of each porel. Of course, the two dimensional analogy of surface area to volume ratio measurements, the ratio of object perimeter to area or the dimensionless shape factor parameter, can be measured from chain codes. Also, a classic result from Ehrlich [20] is that the boundaries of the convex hulls or covering representation with respect to the centroid of a porel can be encoded in a Fourier series. This technique has been used to characterize the angularities of grains [16] and can be applied to any nearly convex feature once boundary encoded[2]. In all, a very rich set of textural analyses can be made from the results of CSC's curve vectorization of image objects.

7.5.2 Coding Thinned Arcs

Together with the results of sequential thinning, an even greater set of analyses relevant to fluid flow properties can be derived from CSC's lists. The object count in thinned results is equivalent to the object count in the original segmented image, since objects can be thinned to a point $\{o\}$ but not to $\{\emptyset\}$. Knowing only the thinned results, it is possible to very quickly calculate estimates of the orientation

[2]The only provision is that the boundary must be describe by a *single valued* function of radius from the object's centroid. Although I have asked him, Prof. Ehrlich did not say what he calls this restriction on shape

and aspect ratios of elongated objects in the image. The less elongated an object is, the poorer the quality of estimates derived in this way because large, low aspect ratio objects such as circles, thin to very sort line segments, nearly points at their centers. In thinned lines, endpoints of thinned results with large quench function values indicate that that end of the thinned result is some distance from the perimeter of the original object, which makes it more difficult to describe accurately what orientation or aspect ratio the object possessed. For elongated or interconnected pore spaces, particularly those which taper to points, the thinned result will have an extension of the thinned arcs in the result out to those cusps of small subtended angle [47, p.383] The quench function arcs will run out to a single pel protrusion, with gradually lessening values, centered near the medial axis of that spiky pore throat. In vectorized representations of quench function arcs in a thinned result, the endpoint values can be used to test for validity of estimates of object orientation and aspect ratios from the thinned result.

The motivation for doing this on the thinned image rather than in the original is that the thinned result is often simpler to vectorize. For elongated objects it is faster to vectorize a medial axis arc than to go completely around the border of the object for the orientation information alone. The caveat is that for more rounded objects, this efficiency is gained at the price of accuracy.

Another curve coding use of thinned results is a boundary follower, that is a curve coder that finds its way around closed loops only once. With these, it is possible to classify the flow topology of multiply connected objects. With structuring element sets, it is possible to define a hit or miss transformation that will identify the nodes and endpoints as well as the edges of loops in these thinned results. This information can group segments into collections of nodes. From these collections, given in an Appendix, I map neighbor coded thinned images to the particular topology of Cederberg-Sobel coding. I then am able to list all nodes and arcs and connections in the image through a single data pass.

7.5.3 Vectorization

And so, linking FMT with CSC curve vectorization, many exciting possibilities result. In addition to all standard petrographic image analysis attributes for the texture of objects in an image, additional measurements relevant to the topology, that is connectivity among the pores is quantifiable. The vision engine is less crucial to the line following or vectorization problem, and yet a division of effort between the engine and the host computer can significantly speed the vectorization. I have realized this by using the vision engine for all neighbor code calculations, and based the CSC algorithms on neighbor coded image inputs.

My image analyzer is a Recognition Technology 1210 vision engine built on a Multibus I backplane. A Multibus to PC-AT bus bidirectional parallel connection allows an IBM PC-AT with Ethernet connections to an IBM RT-PC to control the vision engine under the interrupt-driven MS-DOS operating system. Interrupt driven systems like MS-DOS are better for real-time programming of synchronous machines. With an IBM RT-PC running IBM-AOS[3], the Unix software development environment supports database management for raster to vector conversion.

The challenge of using raster vectorization rather than omnidirectional line following techniques is that the host is responsible for the management of more queues, the growing lists of directional steps linked to quench function values that are being tracked in parallel for each object in the raster array. In some omnidirectional implementations these queues are expressed as lists of coordinates, with or without pel values attached, rather than as lists of offsets, for clarity. I have used this concept in CSC.

The representation of objects in an image must follow the accumulation of object information in a database. This is because in any given raster image, the object count, much less the topology, is not known before following the boundaries or curves. Clearly then, in order to access the image's topology, it is necessary to use

[3]A.K.A. 4.3 bsd Unix

CSC or some other curve vectorization as a complementary routine to sequential thinning.

Chapter 8

Conclusions

Broadly, there are three conclusions from my development of fast Montoto thinning that are relevant to its use in the study of rock properties. First, as was stated by Bel-Lan and Montoto [2, p.37] their thinning transform is relevant to medial axis measurement of elongated objects in an image. Once I implemented FMT on the (Recognition Technology) vision engine, I found FMT to be more useful than I had expected. Not only did it thin elongated pore features, as predicted by Bel-Lan and Montoto, but it also appeared to thin fat, interconnected pores quite well. This result was encouraging, because it was relevant to my original motivation for studying sequential homotopic thinning, namely, characterization of pore spaces in thin section images. Lastly, the fast Montoto transform is an incomplete analytical tool without an associated curve and boundary vectorizing routine. To that end, I developed Cederberg-Sobel compilation to list thinned arcs in a form that is compatible with other numerical approaches to object classification.

To restate the last conclusion, while fast Montoto thinning is a digital algorithm for homotopic thinning, the results of its calculation can not be communicated to other analytical routines or used in pattern recognition without first summarizing the thinned objects into a descriptive list. Together, FMT and CSC can take binary image heterogeneity and automatically produce circuit description lists.

8.1 Elongate Object Classification

The paper that provided the starting point for FMT, by Bel-Lan and Montoto, stated in its conclusions that the type of objects that were represented well by their algorithm were elongated ones, and decisions to use thinning could be based on the ratio of segmented object area to width in the expression S/w^2 where the width is squared [2, p.40]. My conclusion here is a refinement of theirs, for a similar algorithm in 1981.

The refinements I wish to add are that the roughness of the boundary in the objects being thinned plays an important role in how successfully it will be reduced to a representative medial axis, a observation also made by Serra [47, pp 375–379]. For instance, if the object is a circle which has two single-pel protrusions from opposing ends, protruding just one pel further than the covering representation of a circle drawn in an eight connected sense, these two protrusions alone can start branches of medial axes that run from the boundary all the way through the sequential thinning.

Admittedly, Bel-Lan and Montoto solved this problem, but the solution carried the cost that I could not implement it readily in my vision engine. For a fairly minor improvement, they devised a fairly inexpensive solution for general purpose machines. This same small improvement would have halved the speed of thinning on my array processor, so I neglected to include it. In some rockphysics applications, this sensitivity is actually an advantage. Also, since except for vuggy porosity, most porels in rock thin sections have pointed sides to their texture, the result of sequential thinning applied to these pores may be more satisfactory than anticipated. What exactly does happen in the case of vuggy porosity, or circular objects in general?

Depending on the orientation, any slight eccentricity in the the nearly circular shape, the object will be thinned to a small line segment near the circular centroid of such convex objects. The quench function values all along this short line will have similar, large values, large with respect to the length of the line segment in which the quench function is located. This criterion is one put forth by Rosenfeld as

representative of the "true" skeleton, and it is also true of many skizes [45, p.240][47, pp 397–400].

In practical detail, thinning should start a medial axis for those borders that are spiky. Analytical spikiness means that the conditional bisector of order one, S^1, as described in Section 5.5.3 and illustrated in [47, pp 383–385], is able to identify the tips such cusps with one iteration. Some skeletonizing routines, however, are more sensitive to these spikes than others that use more sophisticated conditioning of the sequential ablation or that effects the thinning, and are as sensitive as the S^0 conditional bisector to the identification of boundary roughnesses that are led to by a string of quench function values.

If these stray branches or bones of the skeleton, are fewer in number, a representation for flow topology characterization is a better one. This desirable property of exhibiting S^1 conditional bisector sensitivity to the formation of stray branches is partly attributable to Bel-Lan and Montoto's thinning-2 condition. Thinning-2 is implemented as a feature of the fast Montoto transform [2, pp 42–44] and partly honored even for boundary coding by CSC, since it is given a distince topological class in Table T-2.

With these representations, the tapering reduction of quench function values along the arc of a thinned result are those quench functions which were the result of a low aspect ratio objects in the indicator function image. With four directional ablation and conditioning with a structuring set that discards only 8-simple points [2, p.39][45, pp 232-233], the results of fast Montoto thinning do not often leave one common type of skeletal artifact. That is, in some types of morphological skeletonization, a square is reduced to an ×shaped skeleton, where each 90° corner of the original square initiates a line that ultimately becomes part of the skeleton. In the case of fast Montoto thinning, the conditional sequential thinning is constrained to attach medial line members to an enlarged interior set of the porel. That is, a few of the border pels identified by ablation are "wasted" each iteration, fed back,

as it were, to the intermediate result that is being iterated upon to avoid some of these artifacts. Just as a small amount of feedback can stabilize linear electronic circuits, this morphological feedback[1] stabilizes the medial axis element search, and oftener than with most other thinning algorithms, only the salient porel features are characterized by strings of quench functions.

An exciting corroborative possibility is that the steady state flow network permeability can be linked to bulk permeability measurements through a clever manipulation of pore throat size distributions. This insightful stochastic interface between automated network analysis and laboratory petrophysical service analysis was developed at Stanford recently by Phillipe Doyen of Stanford's Rockphysics Laboratory [14]. His work developed a means of determining the effective pore throat size when represented as porosity and pore throat size distributions.

Also, the ability to use the spatial disposition of these pore throat sizes as a constraint in numerical modeling of flow properties can provide a link between petrographic image observations and numerical techniques such as multiphase flow modeling using cellular automata. These fluid flow models use lattice gases to "flow" viscous "fluids" through a gridded network. This has been developed as a rock porespace application by Dan Rothman at M.I.T.

Numerical techniques such as multiphase flow modeling with cellular automata or in reservoir simulation studies may benefit from a more succinct representation of a flow network, particularly one derived from a natural image. The link between petrographic image observations and these other techniques at different scales can be the circuits produced by FMT-CSC application.

The model of heterogeneity as a linear circuit, automatically derived from a binary image, is the final output of FMT-CSC. Elongated and multiply connected binary image features are automatically modeled as topologically representative arcs with shape description data along their lengths, compiled to lists for use as a linear

[1]The first of its kind, to my knowledge, due to Bel-Lan and Montoto

system of equations or for object segmentation in a set of descriptive files.

8.2 Network Image Extraction

The second main conclusion from my implementation of fast Montoto thinning is that beyond the low aspect ratio features that were recognized in 1981 by Bel-Lan and Montoto as a class of objects that were useful to study with their transformation, fast Montoto thinning is relevant in the analysis of objects that are fat and interconnected, forming multiply connected systems such as pore throat/chamber systems. Many types of pores, when imaged under thin section, will show pores somewhat interconnected systems of lower aspect ratio objects, like pore chambers. Connections between porels are spoiled by the two dimensional sampling of the thin section slice in bulk rock. Since fast Montoto thinning is a type of sequential homotopic thinning, the connections, however thin, between the porels in the original digital image are preserved in the thinned representation. For purposes of network modeling, percolation can be improved by a number of techniques, either before or after the thinning. The homotopy, or preservation of topology from the original indicator image in the thinned result, is affected by a number of things which I also discuss in this section. Namely, the orientation of the original image may affect the accuracy of the quench function values in the result, as they are representative of medial axes in the original indicator image. Some orientations work better than others at placing the thinned results accurately along a medial axis. Also, certain configurations of border shapes together with small holes within the objects can result in loops or loop segments in the quench function arcs. Fast Montoto thinning exhibits a semi-stability around holes, in that certain holes around objects that are being thinned, depending on their proximity to borders of certain irregular shapes, may not be accurately represented by a fully closed loop surrounding the hole.

Finally, when thinning fat, interconnected systems, the results are affected by the sequence of border checks, that is the directions from which the sequential ablation occurs. At low aspect ratio, pore chambers that have cusps connecting them to pore throats around their perimeters can be studied. These cusps produce skeletal trails

of quench function values that increase their value and merge near the center of the pore chamber, identifying a pel or a few pels that symbolize the pore as a node between three or four quench function arcs. This sort of approach applied in parallel over an entire image can result in a network representation of all pore chambers and their connections to throats that are implied by the image. The topology supported by CSC permits no more than four arcs to arrive at a single node. Several node pels may be clustered, however, to accommodate large numbers of converging arcs.

These filaments of quench function values radiating from the centers of pore chambers provide points that can be checked for orientation with neighboring pore chambers' threads of quench function values. In this way, the disposition of porels within the image may be evaluated by checking the distance between endpoints in the thinned arcs, of which only a small number would be generated for any pore chamber.

This property of fast Montoto thinning would make it a useful technique when characterizing grains that were imaged in some suspension, in that grains that appeared in the image to be in contact, or very nearly so, would be identified numerically by being linked with a continuous arc of quench function values. Those that were not quite touching would have a gap between their quench function values. It would be natural in the Montoto thinning of such an image that even with circular grains, contact points between the grains would leave a arc of quench function values leading toward the center of each contacting disk. To do this more reliably, an extended set of medial axis neighbor codes is given in an Appendix. How to image grain suspensions in a two dimensional imaging environment is an open question.

For approaches to pore space connectivity evaluation that would model the fluid flow in the two dimensional plane of the image, approaches to synthetically connecting isolated image elements can be arranged to increase the usefulness of thinned results in the characterization of interconnected or nearly connected porels. For crudely improving the connectivity of isolated porels in the image, semi-homotopic

closing[2] offers a pragmatic solution in that after a certain number of dilation cycles, objects that are near one another in any direction can be made to coalesce through a global image operation. In this way, neighboring objects will be shown connected in the thinned result. An advantage of this approach to improving porel percolation is that it is global, and can thus be implemented in parallel over the entire image without consideration of how many porels are found in the image.

An approach that is morphologically more correct is to perform thinning on the original segmented image and then, using CSC vectorization, identify all isolated arcs of quench function values in the thinned result. By characterizing the chain codings for each curve, a form of universal kriging can provide unbiased interpolations from the end of each curve in the object by way of linear or, with the technique of Bookstein [6], parabolic interpolation. By plotting or rendering these interpolated extensions in the thinned result image, the image may be revectorized after a sequential pruning operation simplifies connections which are the result of two extrapolations crossing one another. In this way, only well oriented thinned results will be extrapolated, which is morphologically more correct than the global extrapolation of the image input to sequential thinning. In fact, a wide variety of techniques can be used to approach this problem, using FMT and CSC together.

Unfortunately, the results of fast Montoto thinning are affected by the orientation of the given input image. For instance, if the input image to fast Montoto thinning is a segmented result of a thin section view, digitized, then it is possible to rotate the thin section before digitization. Thinning a French curve, a drafting instrument with boundaries of constantly varying radius of curvature, I have found orientations that sometimes resulted in a smooth thinned curve very near its medial axis, but in other orientations, the square gridding of the digitizing process introduced many points along the curve boundary that were mistakenly identified as protrusions, and resulted in many arcs of nonzero quench function values reaching from the original

[2]Pre-dilation of the image objects followed by homotopic thinning with extended medial axis codes

object perimeter to the main medial axis line of the skeleton.

A single pel hole in an otherwise convex object can have a profound influence on that object's skeleton, just has the hole has a profound effect on the object's topology. If the gridded circle mentioned before were thinned, the result is a point or very short line segment of quench function values. Normally, when even a single pel hole is present in such an object, the thinned result is a loop near the median point between the hole and the boundary of the object. In some situations where these holes occur very near the boundary of the object, and also when the object's boundary has certain shape properties, only loop segments are present in the thinned image. This is a consequence of the usually desirable property of fast Montoto thinning that 90° corners are not identified as medial line segment members. Thus, when the sequential thinning breaches a gap between the boundary and the hole, in certain boundary configurations this can occur as a 90° corner. As such, it will not meet the medial axis conditioning criteria, and so will not be preserved as a quench function arc.

As with many other types of sequential thinning, fast Montoto thinning is not strictly homotopic. It does not preserve the connectivity properties of all digitized objects at all scales and orientations. A certain problem I have observed when testing FMT on objects around my lab is found mainly when thinning man-made objects. Objects with straight edged holes within their boundaries do not produce identical loops for nominally identical holes. That is, given two rectangular holes in an object near an edge of the object, such as sprockets along the edges of 35mm film, with slight rotations of the film image, the thinning will not produce a symmetrical loop around all sprockets in the film. This results from as little as a single pel difference in the discretization of these holes when segmentation is performed from a gray level input image. Serra in [47, p. 388] has recognized this as a general problem in sequential thinning operations run on digital systems. Rosenfeld has mentioned in [45] that many thinning routines are not stable with rotation. FMT

is not alone in this problem, but it need not be a great limitation on its utility.

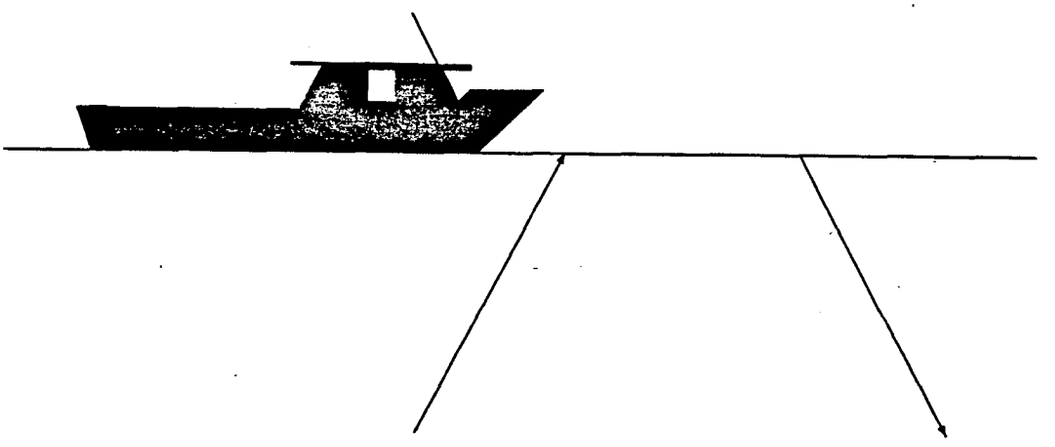
Watching FMT's intermediate results displayed on a graphics screen, it is possible to see how differences in discretization as small as a single pel between two holes result in situation similar to those seen in Conway's Life game. As ablation occurs from different directions, the holes in objects grow from iteration to iteration. When two holes growing toward each other meet, it is possible for two matched holes that differ in their discretized size, or because of rotation, to affect the symmetry of the loops formed around those holes. This problem is found sometimes even if the loops do not break into segments, and remain closed around holes, thereby preserving the object's homotopy.

As a less pathological detail in the implementation, the shape of thinned arcs along medial axis lines are affected by the border check sequence [2]. I have experimented with a small variety of sequences. There are more choices to be made in the ablation direction sequence when using 8 directional or hexagonal ablation, but these would leave only four or six-way connected thinned results.

Generally, the smoothness of the medial axis improves considerably when opposing directions are used in the ablation sequence. The calculation of width values is also simplified if the first two ablation directions are orthogonal. This observation has led to my use of the NEWS ablation sequence. It is not very satisfactory to march the ablation directions around in a circle. Potentially, by using eight directional ablation, the homotopy of the image can be better preserved in some situations by having the quench function values represent the size of inscribed shapes other than squares. However, without allowing four-way connected configurations to occur in the medial line set, Combinations of 8 and 4 way ablation leads to the intractable situation of trying to generate a curve that is both 4 and 8 way connected. It is not possible to have anything but a horizontal or vertical line be both 4 and 8 connected [45, p.239]. In practice, the medial lines just become disconnected at points where the connectivity rules change.

When thinning with eight directional ablation, it becomes more important to use opposing directions in successive iterations. Since a circular progression of eight ablation directions would only repeat every eight iterations, this sequence would cause the direction of a medial axis to change slowly as new points were added, producing a wave-like ripple along what is expected to be a smooth medial axis. While the same is true when using four directional ablation around a circle of directions, the wavelength of these ripples is half of what is caused by eight directional ablation, since the directions cycle twice as often. Since these medial lines oscillate about the smooth one, the wave amplitude of this ablation sequence artifact is noticeably greater in circular eight-ablation than in circular four-ablation.

Overall, despite the artifacts, most natural images of pore space do not exhibit the pathological conditions found in digitizations of French curves or 35mm film sprockets. In fact, FMT produces a very satisfactory medial axis representation with a quench function for a wide range of fat, interconnected pores, and many classes of pores in rock, even those with a fairly high aspect ratio together with a few pore throats connecting them to others in the image.



8.3 FMT And CSC Together

My third conclusion is not the result of great insight. However, I feel strongly that it is an important point to reiterate at this time. That is, fast Montoto thinning is analytically incomplete without an associated curve vectorizer. Transforms from mathematical morphology that produce results for any local configuration of an object are found in counterpoint to global operations over an entire image which run on parallel machines, but which must have curve follower function support to produce practical results.

Curve followers read from the image array connected sets of object features such as porel boundaries or thinned medial axis curves. Even simply to count the number of objects in a feature, one needs raster to vector support to avoid the parallel calculation of centroids followed by histogramming of centroids. Vectorization is distinct from morphological processing and in fact requires its own independent set of criteria to develop.

This separation of needs is well developed by Pratt [40] and by Rosenfeld [45], but is not emphasized by Serra [47]. It implies that vectorization is a picture processing, but not a morphological processing operation. In fact, while it is easy to derive a wealth of textural statistics from a boundary coding of an object, it is notoriously difficult to make fast morphological types of calculations on the boundary code without first rendering it in a raster form [45]—this is like zero-padding a FORTRAN file to take it from flat form to an array form.

The use of a line follower was referred to only briefly in Bel-Lan and Montoto [2, p.40], and yet any analytical use of the thinning technique they presented demands its application for any practical result short of characterizing the algorithm's rate of progress. To cross this gap and use FMT in analytical work for the earth sciences I have devised Cederberg-Sobel compilation.

Cederberg [11], provides a reasonably explicit description of a raster tracking algorithm that can be used for both line following and object boundary following,

while implementable on a raster system, even partly on a vision engine. This means specifically that a small executable code could work through a long scan of data without either reading the entire image into core memory or ever needing to backtrack. This is an important feature when working with less easily accessible data as read out from a vision engine's caches, or for well logs like the borehole televiewer, where a scan line may be nominally 512 points, but the vertically logged interval may be enormous. For very large images, the expense of reading a tape both forward and backwards is too great, and the more adaptable raster line follower that can track all features present on a horizontal scan line in parallel is better suited to sequential storage devices like streaming or video tapes.

CSC tracks either lines or object boundaries in a sequential scanning of an image. The topology it extracts from an image is based on templates that operate on neighbor coded images. Through my reduction of all lines into morphological classes expressed by templates, the raster tracking procedure is separable into global and locally applied parts. The global aspect is the computation of Sobel neighbor codes. The local part is the exraction of boundary and arc lists. For systems equipped with a vision engine, this is exceptionally valuable, since the image being vectorized is already in the engine's caches. The host machine, reading a single byte neighbor code, has at once all the information needed in Cederberg's three line scanning. The same advantage is realized in FMT.

When operating on a neighbor coded image instead of the original, the logic for responding to all possible code sequences is codified by a many-to-one mapping of the 256 possible neighbor codes onto a particular topology. This mapping is realized as a 256 entry nonlinear look-up table function. In fact, since a single morphological test relation of each pel to its eight neighbors is expressed by a single byte value, any CSC topology is realizable as table of 256 pointers, to a smaller number of database management functions.

The database management functions append values to the end of queues, link

the tails of two queues, branch a single strand into two, and manage memory. Once a line segment has been followed to a branch or merging, i.e. a node, its raster-chain code can be converted to an omnidirectional chain code, depending on the direction by which the raster-scanned line is traversed [11, p.232–233].

Aside from the histogramming types of analyses described earlier, without curve vectorization, the results of Montoto thinning are mainly visualizable. This problem is shared in common with machines calculating cellular automata for a different approach to the study of flow systems. When making use of the results, however, there is a benefit to vectorizing thinned results rather than summarizing the properties of lattice gas or cellular automata images. Lines are vectorized more easily and more succinctly.

With curve vectorization, it is possible to access image texture and connectivity properties that can be combined into dimensionless parameters that may better correlate with the scalar properties known *a priori* for the source of the thinned image. The work of Ehrlich has developed a detailed methodology for this approach using data clustering, clustering with respect to synthetic end-members, and multivariate regression, all stochastic techniques of considerable importance to PIA in detail, as well as this problem of relating dimensionless image parameters to bulk rock properties in general [16, 17, 25, 38, 46, 53, 54, 56].

With the FMT-CSC combination, the current pore throat size sieving approach of PIA can be derived from a combination of Montoto thinning, gray level erosion of the thinned arcs, and curve vectorization via CSC. This is not to imply that such a combination of processing steps would exhaust all available image information. After all, once one starts to use morphological processing more sophisticated than the basics of erosion and dilation, why stop? It is striking that, together with CSC, boundary and morphological analysis with FMT alone will provide more textural information than the texture measures used to find pore throat size distributions in PIA. In other words, standard PIA techniques can benefit from a replacement of its

morphological approach with this more comprehensive one.

This is a significant foundation from which to advance toward improved analysis of petrophysical image properties of reservoir rocks. When combined with conditional dilation, for example, it is possible to perform more subtle *petrofabric* analyses such as the pattern defect detection algorithm [47, p.409–411]. Most of the more advanced morphological processing of defect detection can be derived from a combination of FMT with erosion and conditioned dilation, also called blow-up.

Network modeling derived from rock pore space images is a significant enhancement to porel texture analysis, particularly when the thinning process is used with a "good continuation" criterion to porel merging, based on a decision process using merge merits for criteria [45, p.138–148]. FMT and CSC bring together physical properties of rocks, measured in the laboratory and imaged in thin section, with engineering efforts to model fluid flow through varying rock types, conditioned by geophysical field measurements.

FMT-CSC forms network or circuit models from binary image data represented as quench function arcs that are then vectorized. This process can automatically fill databases that allow comparisons among data from physical models of two dimensional fluid flow, pore system studies from thin section, and other types of image data. This procedure is analogous to collecting remote sensing data and using FMT-CSC to automatically convert its classification to a vectoral geographical information system format. Merging these two data types together allows one to bring the geological image analysis in the U.S.A. up to international standards of image analysis applied to science in general.

An important symbiosis can be realized by an image analysis system that combines a sequential homotopic thinning algorithm together with a well matched curve vectorization routine. This is particularly true when a vision engine is used to increase the efficiency of all global processing steps in the image analysis, as is nearly all morphological image processing. This goal has been realized by the FMT-CSC

pair of algorithms.

With increased efficiency of morphological processing, many elegant and *application specific* image analyzing filters can be designed. These can provide previously unimagined deterministic measures as input to stochastic relational processes like regression. Importantly, more sophisticated morphological image processing can provide whole new *classes* of image measures, rather than the 300 or so combinations of standard texture class measures that have been studied by Robert Ehrlich and his students in PIA [19, 54, 33].

An appreciation for the reasons behind my division of image processing into parallel-implementable processes, like linear combinations and morphological transforms, and non parallel or local processes, like feature extraction and raster to vector conversion, allows one to reassess geological image processing in a forward-looking way. An example of this is the clear distinction between morphological processing (FMT) and curve vectorization (CSC) in my work. Through this division of effort, geological machine vision now has tools to perform curve vectorization more efficiently with preprocessing help from a vision engine, and also is able to vectorize more interesting images than PIA's successively opened porels.

Appendix A

The CSC Topology

This appendix presents a topology, in the sense of mapping each of the 256 Sobel neighbor codes into a particular class of raster scan coding entity. The mapping I present here is only one realization of very many possible topologies, and so to emphasize its definition of a particular style[1] and the level of detail involved with its design, I call this a topological font. The nuances of the meaning of a font carry well with this set. Like a typographical font, a lot of detailed design work was involved, and the full 256 character set is represented. Also like a typographical font, this is suited best to a certain style of usage—in this case, fast Montoto thinned images. The nature of this CSC font is fairly general, and it can be used with partially thinned images, blob (unthinned) images, and reasonably simple contour maps.

Variations of this set could be adapted to have more than four arcs arriving at a single pel. That could improve the detail of an image of contours, for example. A different font could group the neighbor codes into any other sort of topological sets, but probably these sets would be variations of the classes I present here, rather than completely different groupings.

[1]This particular style is suited especially to the raster coding of images that contain FMT output. Other styles of interest to earth science applications include blob boundaries, and topographical contours. These are served by this design, but it is not optimized for them.

Table T-2

This table lists all binary neighbor codes (NC) in rectangular grids. The NC are sorted by topological type and labeled by hex and decimal Sobel code value (Hex). Glyph shows the neighborhood graphically. Element shows the toplogical element (Elem.) that the NC represents. Mline rates each NC for some medial line tests.

The glyph gives lines to each neighbor. Node glyphs are marked with a large dot in the center, pnodes with an outlined square.

Pnodes in the II′ series can be identified only at runtime. For example, all of element "B", will belong to elements 17–24, but its precise type depends on what it is connecting together, given the input image.

The Mline column shows NC that are edges or strand links (c), Montoto medial lines (m), or alternate medial lines (am).

A few NC are not nodals (nan). Such NC are internal to a blob and not used either as edges or medial lines. Some are among the NC identified by Bel-Lan and Montoto's thinning-2 criteria (NAN thinning-2).

Each group's row refers to rows in table T-1.

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
NAN internal—row0 (2ea.)							
1–0x00		1	nan	256–0xff		1	nan
NAN thinning-2—row0 (4ea.)							
128–0x7f		2	nan	248–0xf7		2	nan
224–0xdf		2	nan	254–0xfd		2	nan

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type I—row1 (4ea.)							
2–0x01		3	cm	65–0x40		3	cm
33–0x20		3	cm	129–0x80		3	cm
Nodal Type I′—row1 (4ea.)							
17–0x10		4	cm	5–0x04		6	cm
9–0x08		5	cm	3–0x02		7	cm
Nodal Type II—row2 (12ea.)							
34–0x21		8	cm	182–0xb5		8	c
64–0x3f		8	c	192–0xbf		8	c
66–0x41		8	c	193–0xc0		8	cam
97–0x60		8	cam	194–0xc1		8	c
130–0x81		8	cam	225–0xe0		8	c
161–0xa0		8	cm	226–0xe1		8	

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type II'–Row 2 (12 ea.)							
25–0x18		A	cam	13–0x0c		D	cam
21–0x14		B	c	11–0x0a		E	cm
29–0x1c		B	c	15–0x0e		E	c
19–0x12		C	cm	92–0x5b		E	c
31–0x1e		C	c	252–0xfb		E	c
244–0xf3		C	c	7–0x06		F	cam
Nodal Type I'I–Row 2 (44 ea.)							
18–0x11		61	cm	242–0xf1		61	c
32–0x1f		61	c	10–0x09		62	cm
49–0x30		61	cam	16–0x0f		62	c
81–0x50		61	c	41–0x28		62	cm
113–0x70		61	c	57–0x38		62	c
145–0x90		61	cm	73–0x48		62	cm
160–0x9f		61	c	110–0x6d		62	c
241–0xf0		61	c	121–0x78		62	c

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type I'I–Row 2 (continued)							
137–0x88		62	cm	200–0xc7		63	c
144–0x8f		62	c	232–0xe7		63	c
208–0xcf		62	c	253–0xfc		63	c
240–0xef		62	c	4–0x03		64	cam
249–0xf8		62	c	35–0x22		64	cm
250–0xf9		62	c	63–0x3e		64	c
6–0x05		63	cam	67–0x42		64	cm
8–0x07		63	c	127–0x7e		64	c
37–0x24		63	cm	131–0x82		64	cm
61–0x3c		63	c	132–0x83		64	c
69–0x44		63	cm	196–0xc3		64	c
125–0x7c		63	c	215–0xd6		64	c
133–0x84		63	cm	228–0xe3		64	c
136–0x87		63	c	255–0xfe		64	c

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type II'I–Row 3 (48 ea.)							
26–0x19		65	cm	246–0xf5		66	cm
89–0x58		65	cam	20–0x13		67	cm
153–0x98		65	cm	51–0x32		67	cm
22–0x15		66	c	83–0x52		67	cm
24–0x17		66	c	95–0x5e		67	cam
30–0x1d		66	c	115–0x72		67	cm
53–0x34		66	cam	147–0x92		67	cm
85–0x54		66	c	148–0x93		67	cm
93–0x5c		66	cam	159–0x9e		67	cm
117–0x74		66	cam	212–0xd3		67	cam
149–0x94		66	cm	243–0xf2		67	cm
152–0x97		66	cam	14–0x0d		68	cam
157–0x9c		66	cm	45–0x2c		68	cm
216–0xd7		66	cm	77–0x4c		68	cm
245–0xf4		66	cm	141–0x8c		68	cm

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type II'I–Row 3 (continued)							
12–0x0b		69	cm	140–0x8b		69	cm
43–0x2a		69	cm	143–0x8e		69	cm
47–0x2e		69	cm	204–0xcb		69	cm
59–0x3a		69	cm	219–0xda		69	cm
75–0x4a		69	cm	236–0xeb		69	cm
79–0x4e		69	cm	251–0xfa		69	cm
108–0x6b		69	cm	39–0x26		70	cm
123–0x7a		69	cm	71–0x46		70	cm
139–0x8a		69	cm	135–0x86		70	cm
Nodal Type I'II–Row 3 (48 ea.)							
50–0x31		71	cm	177–0xb0		71	cm
82–0x51		71	c	209–0xd0		71	cam
96–0x5f		71	cm	210–0xd1		71	cam
114–0x71		71	cam	42–0x29		72	cm
146–0x91		71	cm	48–0x2f		72	cm

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type I'II–Row 3 (continued)							
58–0x39		72	cm	40–0x27		73	cm
74–0x49		72	cm	62–0x3d		73	cm
80–0x4f		72	cam	70–0x45		73	c
105–0x68		72	cm	72-0x47		73	cam
122–0x79		72	cam	101–0x64		73	cm
138–0x89		72	cm	126–0x7d		73	cm
169–0xa8		72	cm	134–0x85		73	cam
174–0xad		72	cm	165–0xa4		73	cm
176–0xaf		72	cm	168–0xa7		73	cm
185–0xb8		72	cm	189–0xbc		73	cm
201–0xc8		72	cm	197–0xc4		73	cm
202–0xc9		72	cm	198–0xc5		73	cam
233–0xe8		72	cm	229–0xe4		73	cm
234–0xe9		72	cm	230–0xe5		73	cm
38–0x25		73	cm	36–0x23		74	cm

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type I'II–Row 3 (continued)							
68–0x43		74	cam	183–0xb6		74	cm
99–0x62		74	cm	191–0xbe		74	cm
163–0xa2		74	cm	195–0xc2		74	cm
164–0xa3		74	cm	227–0xe2		74	cm
Nodal Type III and III'–Row 3 (4 ea.)							
98–0x61		75	cam	27–0x1a		76	cm
162–0xa1		75	cm	23–0x16		77	c
Nodal Type II'II–Row 4 (54 ea.)							
90–0x59		78	cam	88–0x57		79	cam
154–0x99		78	cm	94–0x5d		79	cam
217–0xd8		78	cm	118–0x75		79	cam
218–0xd9		78	cm	120–0x77		79	cm
54–0x35		79	cam	150–0x95		79	cam
56–0x37		79	cam	158–0x9d		79	cam
86–0x55		79	cm	181–0xb4		79	cm

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type II'II–Row 4 (continued)							
184–0xb7		79	cam	173–0xac		81	cm
213–0xd4		79	cam	205–0xcc		81	cm
214–0xd5		79	cam	206–0xcd		81	cm
221–0xdc		79	cam	237–0xec		81	cm
222–0xdd		79	cam	238–0xed		81	cm
52–0x33		80	cm	44–0x2b		82	cm
84–0x53		80	cam	60–0x3b		82	c
116–0x73		80	cam	76–0x4b		82	cm
179–0xb2		80	cm	107–0x6a		82	cm
180–0xb3		80	cm	111–0x6e		82	cm
211–0xd2		80	cm	124–0x7b		82	cam
223–0xde		80	cam	171–0xaa		82	cm
46–0x2d		81	cm	172–0xab		82	cm
78–0x4d		81	cam	175–0xae		82	cm
109–0x6c		81	cm	187–0xba		82	cm
142–0x8d		81	cam	188–0xbb		82	cm

Hex	Glyph	Elem.	Mline	Hex	Glyph	Elem.	Mline
Nodal Type II'II–Row 4 (continued)							
203–0xca		82	cm	103–0x66		83	cam
207–0xce		82	cm	167–0xa6		83	cm
235–0xea		82	cm	199–0xc6		83	cm
239–0xee		82	cm	231–0xe6		83	cm
Nodal Type I'III–Row 4 (10 ea.)							
178–0xb1		84	cam	102–0x65		86	cam
106–0x69		85	cm	104–0x67		86	cam
112–0x6f		85	c	166–0xa5		86	cm
170–0xa9		85	cm	190–0xbd		86	cam
186–0xb9		85	cam	100–0x63		87	cm
Nodal Type III'I–Row 4 (10 ea.)							
28–0x1b		88	cam	55–0x36		89	cam
91–0x5a		88	cm	87–0x56		89	cam
155–0x9a		88	cm	119–0x76		89	cam
156–0x9b		88	cm	151–0x96		89	cm
220–0xdb		88	cam	247–0xf6		89	cm

Bibliography

[1] I. Bahralolom, R.E. Bretz, and F.M. Orr. Experimental investigation of the interaction of phase behavior with microscopic heterogeneity in a co_2 flood. In *Proceedings of the 60th Annual Technical Conference of the Society of Petroleum Engineers, Las Vegas, NV*, Tulsa, OK, September 22-25 1985. SPE of AIME.

[2] A. Bel-Lan and Luis Montoto. A thinning transform for digital images. *Signal Processing*, 3:37–47, 1981. A medial axis transform that reduceds binary shapes to width encoded strands.

[3] James C. Bezdek, Robert Ehrlich, and William Full. FCM: the fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2–3):191–203, 1984. FCM clusters mixtures into c sets with parial membership or composition of the various endmember constituents. Provides a useful tool when trying to describe pore-ness and throat-ness quantitatively in an image.

[4] H. Blum and R. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978. A thorough treatement, at least for the continuous domain.

[5] J. Bonnet and R. Lenormand. Constructing micromodels for the study of multiphase flow in porous media. *Revue de L'Inst. Franc. du Pet.*, 42:477–480, 1977. From Trygstad et al.'s references.

[6] Fred L. Bookstein. Closing gaps, and gaps with a stepping-stone, by means of parabolas. *Computer Graphics and Image Processing*, 10:372–374, 1979. Another valuable paper from a well-packed CGIP volume.

[7] T. Bourbié, O. Coussy, and B. Zinszner. *Acoustics of Porous Media.* Gulf Publishing Co., Houston, TX, 1987.

[8] T. Bourbié and B. Zinszner. Hydraulic and acoustic properties as a function of porosity in fontainebleau sandstone. *J. Geophys. Res.*, 90(B-13):11,524–11,532, 1985.

[9] Ronald N. Bracewell. *The Fourier Transform and its Applications.* Mc Graw-Hill, New York, 1978.

[10] Ronald N. Bracewell. Two dimensional imaging, ee262 class notes. 200pp pre-publication typeset notes, with some innovative presentations of linear image processing., Winter 1985.

[11] Roger L.T. Cederberg. Chain-link coding and segmentation for raster scan devices. *Computer Graphics and Image Processing*, 10:224–234, 1979.

[12] Jon F. Claerbout. *Fundamentals of Geophysical Data Processing.* Mc Graw-Hill, New York, 1976.

[13] Jon F. Claerbout. *Imaging the Earth's Interior.* Blackwell Scientific Publications, Palo Alto, CA, 1985.

[14] Phillipe M. Doyen. *Transport and Storage Properties of Inhomogeneous Rock Systems.* PhD thesis, Stanford University, April 1987. Primarily useful are the first two chapters, pp 1–65.

[15] Jean Serra (ed.). *Image Analysis and Mathematical Morphology Volume 2: Theoretical Advances.* Academic Press, London, 1988.

[16] Robert Ehrlich and Maureen Chin. Fourier grain-shape analysis: A new tool for sourcing and tracking abyssal silts. *Marine Geology*, 38:219–231, 1980. Describes spectral properties for any perimeter which can be defined as a single-valued polar function.

[17] Robert Ehrlich, Sterling J. Crabtree, and Stephen K. Kennedy. Direct estimation of flow properties from petrographic thin section analysis. In W. C. Park, D. M. Hausen, and R. D. Hagni, editors, *Applied Mineralogy (Proc. 2nd Int. Congress on Appl. Mineralogy in the Minerals Industry, Los Angeles)*, pages 205–221, New York, NY, February 1984. Am. Inst. Min., Metall., and Pet. Eng., Metall. Soc. Ehrlich presents IA for mining engineers. Most readable of Ehrlich's many rewrites on the subject.

[18] Robert Ehrlich and K. O. Horkowitz. SPE 13263 a strong transfer function links thin-section data to reservoir physics. In *Proc. SPE Annual Tech. Conf. and Exposition, Houston, TX*, page 7, Dallas, TX, September 1984. Society of Petroleum Engineers. Ehrlich presents IA to petroleum engineers, with some description of his spectral variables and erosion-dilation.

[19] Robert Ehrlich, Stephen K. Kennedy, Sterling J. Crabtree, and Robert L. Cannon. Petrographic image analysis, I. analysis of reservoir pore complexes. *Journal of Sedimentary Petrology*, 54(4):1365–1378, December 1984. Ehrlich describes IA for petrographers, but here he is not at his most eloquent.

[20] Robert Ehrlich and Bernhard Weinberg. An exact method for characterization of grain shape. *Journal of Sedimentary Petrology*, 40(1):205–212, March 1970. Ehrlich's original writeup of perimeter coding when grain shape can be described by a single-valued polar function. This limitation may indicate the value of using Freeman chain coding rather than Ehrlich's method for boundary harmonic analysis.

[21] I. Fatt. The network model of porous media: Iii. dynamic properties of networks with tube radius distribution. *Trans. AIME*, 207:164, 1956.

[22] Frank H. Feng and Michael J. Gillotte. Grain separation in digital titanium alloy photomicrographs— its application to non-destructive material test. In *Proc. IEEE Comput. Soc. Conf. of Pattern Recognition and Image Processing, RPI, Troy, NY*, pages 160–167, New York, NY, June 1977. IEEE. Useful descriptions of boundary coding, alternatives to Freeman chains, Polygonal perimeter fitting references.

[23] H. Freeman. Computer processing of line-drawing images. *Computing Surveys*, 6:57–97, 1974.

[24] William E. Full, Robert Ehrlich, and James C. Bezdek. FUZZY QMODEL—a new approach for linear unmixing. *Mathematical Geology*, 14(3):259–270, 1982. Assignment of partial membership in endmember groups by sample.

[25] William E. Full, Robert Ehrlich, and Stephen K. Kennedy. Optimal configuration and information content of sets of frequency distributions. *Journal of Sedimentary Petrology*, 54(1):117–126, March 1984. A clear, succinct description of maximum entropy spectra with examples. Refreshing to read.

[26] William E. Full, Robert Ehrlich, and J. E. Klovan. EXTENDED QMODEL—objective definition of external end members in the analysis of mixtures. *Mathematical Geology*, 13(4):331–344, 1981. A practical idea for creating synthetic endmembers to help discriminate mixtures more flexibly than depending on the presence of endmembers in the mixture. A poorly written article.

[27] R.M. Haralick. *Glossary of Computer Vision, in RTILIB Tutorial.* Recognition Technology, Inc., Westboro, MA, 1986. Included as a 50pp addedum, I am not aware of it being published elsewhere.

[28] H.J.A.M. Heijmans and C. Ronse. The algebraic basis of mathematical morphology, part i: Dilations and erosions. Philips Research Laboratory—Brussels unpublished manuscript M248, received from C. Ronse, Av. E. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium., June 1988.

[29] Yin Hezhu. Stanford Rockphysics Laboratory, personal communication, 25 Feb. 88.

[30] Paul Horowitz and Winfield Hill. *The Art of Electronics.* Cambridge University Press, New York, 1980.

[31] Andre Journel and Charles Huijbregts. *Mining Geostatistics.* Academic Press, New York, 1978.

[32] Sterling J. Crabtree Jr., Robert Ehrlich, and Christopher Prince. Evaluation of strategies for segmentation of blue-dyed pores in thin sections of reservoir rocks. *Computer Vision, Graphics, and Image Processing*, 28:1–18, 1984. Describes, with a hint of pretense in its presentation, the advantages of RGB over composite video's IYQ true-color data coding schemes when segmenting pores. Jim Garrison has solved this problem and will explain it to you in sixty seconds instead of eighteen pages.

[33] Stephen K. Kennedy. Microcomputer image acquisition and shape and size analysis with applications in sedimentology. *Compass of Sigma Gamma Epsilon*, 62(3):205–210, 1985. A short note reviewing Ehrlich's work and announcing Kennedy's purchase of an IBM XT system.

[34] Thomas M. Lillesand and Ralph W. Kiefer. *Remote Sensing and Image Interpretation.* John Wiley & Sons, New York, NY, 1979.

[35] Prof. R. Lyon. Stanford Remote Sensing Laboratory, Personal Communication.

[36] T.R. Madden. Microcrack connectivity in rocks: A renormalization group approach to the critical phenomena of conduction and failure in crystalline rocks. *J. Geophys. Res.*, 88:585–592, 1983. Presents results from renormalization of crack networks in a cubic lattice. This is an extension of his physical models using resistor networks, where a crack is postulated to exist perpendicular to every missing resistor.

[37] Georges Matheron. Random sets theory and its application to stereology. *Journal of Microscopy*, 95, part 1:15–23, Feb. 1972. From Serra's references.

[38] Luis Montoto. Digital multi-image analysis: Application to the quantification of rock microfractography. *IBM Journal of Research and Development*, 26(6):735–745, November 1982. Presents an application of Montoto's thinning algorithm and some IA analysis of the resulting features are novel. A Rose diagram of medial line directions presents results that are difficult but possible to get through geometrical covariance calculations.

[39] Theodosios Pavlidis and Steven L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, C-23(8):860–870, August 1974. A precise piecewise approximation algorithm is described but it may be too inefficient to compute for complicated multi-pore images.

[40] William K. Pratt. *Digital Image Processing.* John Wiley & Sons, New York, NY, 1978.

[41] B.B. Quinn. Quantified pore connectivity using montoto's transform. *EOS*, 68(44):1490–1491, 1987. (abstract).

[42] Recognition Technology Inc., 160 E. Main St., Westoboro, MA. *RTILIB/500 Subroutine Reference Manual*, Dec. 1986. Recognition Technology Inc.'s version 4.00 system support.

[43] Wolfgang Riepe and Monika Steller. Characterization of coal and coal blends by automatic image analysis. *Fuel*, 63:313–317, March 1984. German IA work on coal with estimation of volume fractions from histograms, calls dilation with template "blow up", imaging system by Leitz with Plumbicon sensor.

[44] C. Ronse and H.J.A.M. Heijmans. The algebraic basis of mathematical morphology, part ii: Openings and closings. Philips Research Laboratory—Brussels unpublished manuscript M291, received from C. Ronse, Av. E. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium., February 1989.

[45] Azriel Rosenfeld and Avinash Kak. *Digital Picture Processing*, volume 2. Academic Press, Orlando, 1982.

[46] K. Ruzyla. Characterization of pore space by quantitative image analysis. *SPE Formation Evaluation*, pages 389–398, August 1986. High resolution IA at Exxon: 704 ×896 ×8 bit images with a Vidicon. suggests use of Carman-Kozeny relation for K estimation from ϕ. Study of pore size distribution.

[47] Jean Serra. *Image Analysis and Mathematical Morphology.* Academic Press, London, 1982.

[48] Irwin Sobel. Neighborhood coding of binary images for fast contour following and general binary array processing. *Computer Graphics and Image Processing*, 8:127–135, 1978. I was directed to this reference by Indranil Chakravaty of Schlumberger-Doll Research.

[49] Audubon Society. *Field Guide to Trees.* Audubon Society, New York, 1978.

[50] S.R. Sternberg. Grayscale morphology. *Computer Vision, Graphics, and Image Processing*, 35(3):333–355, 1986. Found by way of reference from C. Ronse's unsolicited manuscript.

[51] Jean-Paul Tremblay and Paul G. Sorenson. *The Theory and Practice of Compiler Writing.* McGraw-Hill, New York, 1985.

[52] J. C. Trygstad, Robert Ehrlich, and N. C. Wardlaw. SPE/DOE 14891 physical modeling of microscopic rock-pore heterogeneities. In *Proc. SPE/DOE 5th Symp. on Enhanced Oil Recovery, Tulsa, OK*, pages 151--161, Dallas, TX, April 1986. Society of Petroleum Engineers. IA pore images are etched into glass and studied for K. This should be done numerically thinning and extrapolation.

[53] N. C. Wardlaw and J. P. Cassan. Estimation of recovery efficiency by visual observation of pore systems in reservoir rocks. *Bull. of Canadian Petroleum Geology*, 26(4):572–585, December 1978.

[54] Joan S. Weszka. A survey of threshold selection techniques. *Computer Graphics and Image Processing*, 7:259–265, 1978. Good article for orientation to thresholding and introduction to jargon. Prophylactic reading of this article is strongly recommended before studying Crabtree, Ehrlich, and Prince.

[55] David P. Yale. *Network Modelling of Flow, Storage and Deformation in Porous Rocks.* PhD thesis, Stanford University, August 1984.

[56] M. Yanuka, F. A. L. Dullien, and D. E. Elrick. Serial sectioning and digitization of porous media for two- and three-dimensional analysis and reconstruction. *Journal of Microscopy*, 135(2):159–168, August 1984. 3-d IA with a PC system, photographs were projected and shot with a Sony video camera which was digitized. 254 $\times$242 $\times$8 bit imaging. Some interesting references.