IMAGE TEXTURE ANALYSIS:

Classification and Segmentation

TAN Tieniu

A thesis submitted for the

Degree of Doctor of Philosophy of the University of London

June 1989

Signal Processing & Biomedical Systems Section

Department of Electrical Engineering

Imperial College of Science, Technology and Medicine

University of London

This thesis is dedicated to

my family,

my people,

and my motherland.

谨以此文献给:

我的家人

我的人民

和我的祖国.

------ 谭铁牛

一九八九年夏于伦敦

ABSTRACT

Texture is an ubiquitous surface characteristic of real-world scenes, and is a prominent clue in the human perception of the surrounding world. Texture analysis is essential to the analysis of many types of images. Classification of a given texture sample into one of a finite number of prototype textures (texture classification), and partitioning of a given image into disjoint regions of homogeneous textures (texture segmentation) are the two fundamental problems in texture analysis. This thesis is devoted to the development of effective and efficient algorithms towards the resolution of these two problems, and thus gaining a better understanding of texture. Existing approaches to texture analysis, being mostly *ad hoc* in nature, are mainly concerned with texture classification. For texture segmentation, very limited success has been reported in the open literature as it is a much more challenging problem.

The first class of methods developed in this thesis is based on *local vector mapping*, a new concept introduced in the thesis. This concept enables one to offer a consistent and convenient explanation, a unification and generalization of some existing approaches. Functionally, a local vector mapping operation is composed of local neighborhood specification, local vector formation and vector mapping or transformation. Successful algorithms for both texture classification and segmentation are developed based on this concept.

The second class of methods draws upon the recent findings in the physiology and psychophysics research communities. As the human visual system (HVS) is no doubt the most effective and efficient system to perform texture analysis, computational algorithms should mimic the HVS. One of the most popular models for the processing of pictorial information in the HVS is the so-called *spatial channel* model. This model hypothesizes that the input pictorial information to the visual cortex is processed via a set of parallel, quasi-independent channels each of which is tuned to a specific band of spatial frequency and orientation. By adopting this model, very successful algorithms for texture classification and segmentation are developed in this thesis.

In addition to the two major classes of approaches mentioned above, the thesis also contains chapters on the description of new algorithms for feature extraction in texture classification based on local surface fitting and primitive analysis. These algorithms can be considered as supplementary to the first class of algorithms.

ACKNOWLEDGEMENTS

The study presented in this thesis was pursued under the supervision of Prof. A. G. Constantinides at the Department of Electrical Engineering, Imperial College, London. I wish to express my gratitude to Prof. A. G. Constantinides for his constant encouragement and many helpful discussions during the course of this project.

I am also indebted to the members of the Signal Processing and Biomedical Systems Section with whom it has been my pleasure to work over the last few years. In particular, I wish to extend my thanks to Ming Hong Chan, Makis Livadas, Tom Hibbert, Nurdal Turkbeyler and Yanbin Yu, with whom I had numerous, lengthy and beneficial discussions on the project. My special thanks must go to Tom Hibbert for proof-reading a draft of this thesis, and to Jose Baben for his generous and patient assistance in preparing the pictorial data for the thesis.

The joint financial support of the Chinese Educational Commission and the British Council which made this work possible in the first instance is gratefully acknowledged.

Finally, I would like to express my deep appreciation for the efforts of my family who contributed decidedly in my education by providing me with a wealth of moral support and encouragement.

CONTENTS

ABSTRACT 3

ACKNOWLEDGEMENTS 4

CONTENTS 5

LIST OF FIGURES 13

LIST OF TABLES 21

LIST OF ABBREVIATIONS 24

STATEMENT OF ORIGINALITY 27

1. INTRODUCTION 31

1.1 What is Texture? 31

1.2 Why Texture Analysis? 36

1.3 Objectives of This Thesis 39

1.4 Organization of The Thesis 41

1.5 References 44

2. TEXTURE CLASSIFICATION: A REVIEW 47

2.1 Introduction 47

2.2 Classical Statistical Approaches 47

2.2.1 GLDM: Gray Level Dependence Matrix 48

2.2.2 GSDM: Generalized Spatial Dependence Matrix 49

2.2.3 HAFA: Histogram and Autocorrelation Function Analysis..... 50

2.2.4 FPSA: Fourier Power Spectrum Analysis 51

2.2.5 GLDA: Gray Level Difference Approach..... 52

2.2.6 GLEA: Gray Level Edge Element Analysis..... 53

2.2.7 GLRL: Gray Level Run Length 54

2.2.8 GLPA: Gray Level Profile Analysis..... 55

2.2.9 LLCM: Local Linear Convolution Method 55

2.3 Structural Approaches 57

2.3.1 Strong Texture Analysis 57

2.3.2 Weak Texture Analysis 58

2.3.3 Syntactic Texture Analysis 59

2.4 Model-Based Approaches 60

2.4.1 ARM: Autoregressive Model..... 60

2.4.2 GMRFM: Gaussian Markov Random Field Model 61

2.4.3 GRFM: Gibbs Random Field Model 62

2.4.4 MRFM: Mosaic Random Field Models 64

2.4.5 RWM: Random Walk Models..... 65

2.4.6 FSM: Fractal Surface Model 66

2.4.7 MSFM: Multichannel Spatial Filtering Model 67

2.5 Remarks 70

2.6 References 71

3. TEXTURE DESCRIPTION BY LOCAL VECTOR MAPPING 79

3.1 Introduction 79

3 2 Local Vector Mapping 79

3.2.1 Local Neighborhood 79

3.2.2 Local Vector Formation 81

3.2.3 Vector Mapping 85

3.2.4 LLCM as a Special Case of LVM 89

3.3 Computing LVMs: Some Examples 91

3.3.1 Linear Local Vector Mapping (LLVM) 91

3.3.2 Non-Linear Local Vector Mapping (NLVM) 95

3.4 LVM Texture Features 99

3.4.1 Definitions of LVM Texture Features 99

3.4.2 LVM Texture Features versus Visual Texture Features 101

3.5 Texture Classification by LVM: Experimental Results 105

3.5.1 Texture Feature Vector Generation 105

3.5.2 The Classifiers 107

3.5.3 Texture Classification By LLVM Features 108

3.5.4 Texture Classification by NLVM Features 111

3.6 Conclusions 114

3.7 References 116

4. LOCAL SURFACE FITTING FOR TEXTURE DESCRIPTION 119

4.1 Introduction 119

4.2 Local Surface Fitting (LSF) 119

4.3 A Case Study: Local Polynomial Fitting 123

4.4 Texture Feature Extraction Based on LSF 125

4.5 Conclusions 126

4.6 References 126

5. TEXTURE FEATURE EXTRACTION BASED ON PRIMITIVE ANALYSIS 127

5.1 Introduction 127

5.2 Primitive Extraction and Primitive Refinement 128

5.2.1 Primitive Extraction 128

5.2.2 Primitive Refinement 130

5.3 Texture Feature Extraction 130

5.4 Texture Classification and Experimental Results 132

5.5 Conclusions 135

5.6 References 137

6. GABOR FILTERS IN TEXTURE ANALYSIS I: TEXTURE FEATURE EXTRACTION AND CLASSIFICATION 139

6.1 Introduction 139

6.2 Gabor Filters and The Visual Cortex 141

6.2.1 Mathematical Descriptions of Gabor Filters 141

6.2.2 Some Properties of Gabor Filters 144

6.2.3 The Visual Cortex 148

6.2.4 The Cortical Channel Model 150

6.3 Channel and Filtering Pattern Design 153

6.3.1 The Frequency Component Resolution Characteristics of The HVS 154

6.3.2 Filtering Pattern for Texture Feature Extraction 160

6.4 Texture Feature Extraction via Gabor Filtering 162

6.4.1 Outline of The Proposed Algorithm 162

6.4.2 Texture Feature Extraction 164

6.4.3 Perceptual Interpretation of The Defined Features 168

6.4.4 Texture Classification Scheme 171

6.4.5 Experimental Results 172

6.5 A Standard Technique for Texture Feature Extraction 175

6.5.1 Texture Description by GLDM 175

6.5.2 Texture Classification Results 177

6.6 The Proposed Algorithm versus The Standard Technique 178

6.6.1 Performance Comparison 181

6.6.2 Computational Considerations 181

6.6.3 Noise Performance 182

6.6.4 Other Considerations 182

6.7 Conclusions 184

6.8 References 185

7. TEXTURE SEGMENTATION: A REVIEW 189

7.1 Introduction 189

7.2 Uniform Texture Region Extraction 189

7.2.1 FSC: Feature Space Clustering 190

7.2.2 STC: Statistical Classification 191

7.2.3 SEA: State Estimation Approach 193

7.2.4 GIS: Generalization of Intensity Segmentation Methods 195

7.2.5 SFA: Spatial Filtering Approach 197

7.3 Texture Edge Detection 198

7.3.1 TGO: Texture Gradient Operator 198

7.3.2 SFT: Spatial Filtering Technique 199

7.3.3 TEA: Theoretical Estimation Approach 199

7.3.4 TBM: Texton-Based Method 200

7.4 Conclusion 201

7.5 References 202

8. TEXTURE SEGMENTATION BASED ON LVM 207

8.1 Introduction 207

8.2 Uniform Texture Region Extraction Based on LVM 207

8.2.1 The Basic LVM Algorithm 208

8.2.2 The Successive LVM Algorithm 214

8.2.3 Experimental Results 216

8.3 Texture Edge Detection Based on LVM 228

8.3.1 The Algorithm 228

8.3.2 Experimental Results 233

8.4 Discussions and Conclusions 239

8.5 References 240

9. GABOR FILTERS IN TEXTURE ANALYSIS II: TEXTURE EDGE DETECTION 242

9.1 Introduction 242

9.2 Texture Edge Detection Via Gabor Filtering 243

9.2.1 The Principles of The Algorithm 243

9.2.2 Theoretical Justification of The Cortical Channel Model 248

9.2.3 Spectral Peak Identification 252

9.3 Experimental Results 253

9.3.1 Segmentation of Artificial Textures 253

9.3.2 Segmentation of Noisy Textures 259

9.3.3 Segmentation of Natural Textures 262

9.4 Conclusions 265

9.5 References 267

10. CONCLUSION 269

10.1 Summary of The Thesis 269

10.2 Suggestions for Further Research 275

10.3 References 278

LIST OF FIGURES

Figure 1.1. Illustration of Texture Smoothness, Coarseness, Regularity and Directionality. Shown are textures (a) D32, (b) D12, (c) D31, (d) D93, (e) D34, (f) D110, (g) D15 and (h) D111. 33

Figure 1.2. Illustration of Deterministic, Structural and Stochastic Textures. 34

Figure 1.3. Illustration of Texture Self-Similarity. (a) Original image; (b) The image after region swapping; (c) The blurred version of (a) and (d) The blurred version of (b) 36

Figure 1.4. Illustration of Preattentive and Attentive Texture Perception 37

Figure 1.5. A Moth at Rest on the Side of a Tree [9]. 39

Figure 1.6. A Fish in the Background of Sand [10]. 40

Figure 3.1. Examples of Valid Local Neighborhoods. 81

Figure 3.2. Examples of Valid Local Vector Formation Procedures. (a) Row concatenation; (b) Column concatenation; (c)-(e) Some variants of (a) and (b) and (f) Peano scanning 83

Figure 3.3. Illustration of the Differences between Distinct Local Vector Formation Schemes. (a) A 5x5 local neighborhood; (b) The local vector obtained via column concatenation and (c) The local vector obtained via row concatenation. 84

Figure 3.4. Four Laws' 5x5 Masks Used in Texture Analysis. (a) Horizontal edge; (b) High frequency spot; (c) V-shape and (d) Vertical line. 89

Figure 3.5. Examples of LDCTed Images. (a) Original Calf texture; (b) $g_1(i,j)$; (c) $g_2(i,j)$; (d) $g_3(i,j)$; (e) $g_4(i,j)$; (f) $g_5(i,j)$; (g) $g_6(i,j)$; (h) $g_7(i,j)$ and (i) $g_8(i,j)$. 93

Figure 3.6. Histograms of Images Shown in Fig.3.5(a) to (i) Respectively..... 94

Figure 3.7. Examples of LPTed Images. (a) $g_1(i,j)$; (b) $g_2(i,j)$; (c) $g_3(i,j)$; (d) $g_4(i,j)$; (e) $g_5(i,j)$ and (f) $g_6(i,j)$..... 95

Figure 3.8. Histograms of Images Shown in Fig.3.7(a) to (f) Respectively. 96

Figure 3.9. Examples of NLVMed Images. (a) $h_{var}[\vec{V}(i,j)]$; (b) $h_{\max}[\vec{V}(i,j)]$; (c) $h_{\min}[\vec{V}(i,j)]$; (d) $h_{led}[\vec{V}(i,j)]$; (e) $h_{grl}[\vec{V}(i,j)]$; (f) $h_{lhl}[\vec{V}(i,j)]$ and (g) $h_{lvl}[\vec{V}(i,j)]$. 98

Figure 3.10. The 10 Texture Classes Used in LVM Texture Classification. (a) Wood Grain; (b) Hand-made Paper; (c) d53; (d) Cotton Canvas; (e) Oriental Straw; (f) Beach Sand; (g) Herringbone; (h) Calf; (i) Grass Lawn and (j) Pressed Cork. 106

Figure 5.1. Illustration of Primitive Extraction and Refinement. 131

Figure 5.2. The 14 Texture Classes Used in Texture Classification Experiments. (a) Pressed Cork (D4); (b) Expanded Mica (D5); (c) Grass Lawn (D9); (d) Woolen Cloth (D19); (e) Pressed Calf Leather (D24); (f) Beach Sand (D28); (g) Beach Sand (D29); (h) Oriental Straw Cloth (D53); (i) Straw Matting (D55); (j) Handmade Paper (D57); (k) Oriental Rattan (D65); (l) Wood Grain (D68); (m) Raffia (D84) and (n) Pigskin (D92). 134

Figure 6.1. Gabor Filters under Different Aspect Ratio λ. (a) and (b), (c) and (d), and (e) and (f) are three symmetrical (the real part of $h(x,y)$ in (6.1)) and anti-symmetrical (the imagery part of $h(x,y)$ in (6.1)) Gabor filter pairs with λ being 0.5, 1.0 and 2.0 respectively..... 142

Figure 6.2. The Symmetrical Gabor Filter $H_c(u,v)$ on The Frequency Domain for Fixed v, and under Different Central Frequency u_0 and Space Constant σ_x, where $u_{01}<u_{02}$ and $\sigma_{x1}<\sigma_{x2}$..... 145

Figure 6.3. The Four Elemental Gabor Filters on The Spatial Time Domain. (a) $h_c(x,y)$; (b) $h_s(x,y)$; (c) Profile of the Middle Row of (a); (d) Profile of the Middle Row of (b); (e) $-h_c(x,y)$; (f) $-h_s(x,y)$; (g) Profile of the Middle Row of (e) and (h) Profile of the Middle Row of (f)..... 152

Figure 6.4. Schematic Diagram of the Proposed Cortical Channel Model (6.32). 153

Figure 6.5. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Stochastic Texture D57 [23]. The Averaging Radius R in pixels is 5. For detailed caption, see text. 155

Figure 6.6. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Stochastic Texture D57 [23]. The Averaging Radius in Pixels is 10. For detailed caption, see text. 157

Figure 6.7. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Structural Texture D53 [23]. For detailed caption, see text. 159

Figure 6.8. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Structural Texture D77 [23]. For detailed caption, see text. 160

Figure 6.9. Illustration of the Proposed Filtering Pattern for Texture Feature Extraction. The number in each circle is the reference order of the underlying filter such as Filter 1, Filter 2, etc..... 162

Figure 6.10. The Block Diagram of the Proposed Algorithm for Texture Feature Extraction and Classification. 163

Figure 6.11. The 16 Channel Outputs Images for a D57 Sample..... 166

Figure 6.12. Histograms of the Images Shown in Fig.6.11. 167

Figure 6.13. Outputs from the Classifier Training Program for Texture Classes (a) Pressed Cork (D4) and (b) Expanded Mica (D5)..... 170

Figure 6.14. The Noise Performance of the Proposed (GF) and the Classical (GLDM) Texture Features..... 183

Figure 6.15. The Deviation-to-Mean Ratios of the Proposed (GF) and the Classical (GLDM) Texture Features..... 184

Figure 8.1. Block Diagram of the Basic LVM Algorithm for UNITRE. 208

Figure 8.2. Illustration of Intensity Thresholding of LVMed Images. 209

Figure 8.3. Selection of Optimal Intensity Thresholds. 210

Figure 8.4. Block Diagram of the Successive LVM Algorithm for UNITRE. 215

Figure 8.5. An Example of the Successive LVM Algorithm for UNITRE. 216

Figure 8.6. Segmentation of Artificial Textures by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_2(i,j)$; (c) Thresholding of (b) at t=210; (d) GLPF of (c) with W=5 and $\sigma=6.0$; (e) GLPF of (c) with W=13 and $\sigma=4.0$; (f) GLPF of (c) with W=19 and $\sigma=4.0$; (g) Intensity segmentation of (d); (h) Intensity segmentation of (e) and (i) Intensity segmentation of (f). 218

Figure 8.7. Segmentation of Artificial Textures by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_9(i,j)$ and (c) Final segmentation obtained with t=180, W=19 and $\sigma=4.0$. 219

Figure 8.8. Segmentation of Artificial Textures by the Basic LVM Algorithm. (a) A space shuttle binary image; (b) A textured version of (a); (c) The LVMed image $g_6(i,j)$; (d) Thresholding of (c) at t=170; (e) GLPF of (d) with W=13 and $\sigma-2.0$ and (f) Intensity segmentation of (e)..... 220

Figure 8.9. Segmentation of Noisy Textures by the Basic LVM Algorithm. (a)-(c) Noisy versions of Fig.8.6(a) at SNR=8db, 6db and 4db respectively; (d)-(f) The LVMed images $g_7(i,j)$ of (a)-(c) respectively and (g)-(i) Final segmentations of (a)-(c) respectively, obtained with t=160, W=13 and

$\sigma = 4.0$. 221

Figure 8.10. Segmentation of Natural Textures (Cotton Canvas and Wood Grain) by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_3(i,j)$; (c) Thresholding of (b) at t=160; (d) GLPF of (c) with W=13 and σ = 4.0 and (e) Intensity segmentation of (d). 223

Figure 8.11. Segmentation of Natural Textures (Hand-made Paper and Calf) by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_7(i,j)$; (c) Thresholding of (b) at t=145; (d) GLPF of (c) with W=11 and σ = 4.0 and (e) Intensity segmentation of (d). 224

Figure 8.12. Segmentation of Artificial Textures by the SLVM Algorithm. (a) Original image; (b) The LVMed image $g_2(i,j)$; (c) The SLVMed image; (d) GLPF of (c) with W=13 and σ = 2.0; (e) GLPF of (c) with W=13 and σ = 4.0; (f) GLPF of (c) with W=19 and σ = 4.0 and (g)-(i) Intensity segmentations of (d)-(f) respectively. 225

Figure 8.13. Segmentation of Natural Textures (Hand-made Paper and Calf) by the SLVM Algorithm. (a) Original image; (b) The LVMed image $g_7(i,j)$; (c) The SLVMed image; (d) GLPF of (c) with W=11 and σ = 4.0 and (e) Intensity segmentation of (d) 226

Figure 8.14. Segmentation of Natural Textures (Wood Grain, Calf, Hand-made Paper and Pressed Cork) by the SLVM Algorithm. (a) Original image; (b) The LVMed image $g_3(i,j)$; (c) The SLVMed image; (d) GLPF of (c) with W=9 and σ = 4.0; (e) GLPF of (c) with W=13 and σ = 4.0 and (f)-(g) Intensity segmentations of (d) and (e) respectively. 227

Figure 8.15. Segmentation of Natural Textures (Cotton Canvas and Wood Grain) by the SLVM Algorithm. (a) A binary space shuttle image; (b) A textured version of (a); (c) The LVMed image $g_3(i,j)$; (d) The SLVMed image; (e) GLPF of (d) with W=13 and σ = 2.0 and (f) Final segmentation 228

Figure 8.16. Block Diagram of the Proposed MUCTED Algorithm. 229

Figure 8.17. Major Operations Involved in the Proposed MUCTED Algorithm. 232

Figure 8.18. Texture Edge Detection in Natural Textures (Cotton Canvas and Wood Grain) by the MUCTED Algorithm. (a) Original image; (b)-(e) The four channel textural gradient images; (f) The total textural gradient image; (g) After thresholding (f) at t=90; (h) After thinning (g) and (i) The detected texture edges superimposed on (a). 234

Figure 8.19. Texture Edge Detection in Natural Textures (Wood Grain and Hand-made Paper) by the MUCTED Algorithm. (a)-g) are analogous to Fig.8.18(a)-(g); (h) After majority filtering and small-isolated-edges elimination operations on (g) and (i) Texture edges detected. 235

Figure 8.20. 8 Synthetic Textured Images Shown in Fig.8.20(a)-(i) Respectively. 236

Figure 8.21. The 8 Total Textural Gradient Images Computed from Fig.8.20(a)-(i) Respectively. 237

Figure 8.22. The 8 Detected Texture Edge Maps for Fig.8.20(a)-(i) Respectively..... 238

Figure 9.1. Block Diagram of Texture Edge Detection via Gabor Filtering. 245

Figure 9.2. Illustration of Iterative Identification of Spectral Peaks. (a) Original Image; (b) Its Power Spectrum; (c) Identification of the First Spectral Peak (After the First Iteration); (d) After the Second Iteration; (e) After the Third Iteration; (f) After the Fourth Iteration; (g) After the Fifth Iteration and (h) the Four Peaks Identified (shown in white dots). 254

Figure 9.3. Segmentation of Artificial Textures. (a) Original image; (b) Its power spectrum; (c) The four spectral peaks detected; (d)-(g) The four channel output images after Gabor filtering; (h)-(k) The four channel texture gradient images; (l) Total gradient image; (m) After thresholding (l); (n) After thinning (m) and (o) Edge map superimposed on (a). 256

Figure 9.4. Segmentation of Artificial Textures. (a) Original image; (b) Its power spectrum; (c) The total texture gradient image; (d) After thresholding (c); (e) After thinning (d) and (f) Edge map superimposed on (a)..... 257

Figure 9.5. Illustration of the Segmentation Resolution of the Proposed Algorithm. (a) A binary image; (b) Its textured version; (c) Power spectrum of (b); (d)-(e) The two channel output images after Gabor filtering; (f) The total gradient image; (g) After thresholding (f); (h) After thinning (g) and (i) Edge map superimposed on (b)..... 258

Figure 9.6. Segmentation of Artificial Textures. (a) A gray image; (b) Its three-region segmentation; (c) A textured version of (b); (d) Power spectrum of (c); (e)-(g) The three channel output images after Gabor filtering; (h) The total texture grdaient image; (i) Texture edges detected and (j) Edge map superimposed on (c)..... 260

Figure 9.7. Segmentation of Noisy Textures. (a) Original image; (b) Its power spectrum; (c)-(f) The four channel output images after Gabor filtering; (g) The total grdaient image; (h) After thresholding (g); (i) After thinning (h) and (j) Edge map superimposed on (a)..... 261

Figure 9.8. Segmentation of Natural Textures (Oriental Straw Cloth and Netting). (a)-(j) are analogous to Fig.9.7(a)-(j)..... 263

Figure 9.9. Segmentation of Natural Textures (Cotton Canvas and Herringbone Wave). (a) Original textured image; (b) Its power spectrum; (c) The channel output image after Gabor filtering; (d) The texture gradient image; (e) Texture edges detected and (f) Edge map superimposed on (a)..... 264

Figure 9.10. Segmentation of Natural Textures (Wood Grain and Cotton Canvas). (a) Original textured image; (b) Its power spectrum; (c) The total texture gradient image; (d) After thresholding (c); (e) After thinning (d) and (f) Edge map superimposed on (a)..... 265

Figure 9.11. Segmentation of Natural Textures (Oriental Straw Cloth and Oriental Rattan). (a)-(f) are analogous to Fig.9.10(a)-(f). 266

LIST OF TABLES

Table 3.1. The TF_3 Features from The Local Discrete Cosine Transform/Mapping 101

Table 3.2. The TF_3 Features from The Local Polynomial Transform/Mapping 102

Table 3.3. The TF_1 Features from The Defined Non-linear Local Vector Mappings 103

Table 3.4. The TF_2 Features from The Defined Non-linear Local Vector Mappings 104

Table 3.5. The Classification Accuracies by Using Individual LPT Texture Features 109

Table 3.6. The Classification Confusion Matrix by Using The 5 LPT TF_2 Features 110

Table 3.7. The Classification Confusion Matrix by Using The LDCT TF_2 Features 111

Table 3.8. The Classification Accuracies by Using Individual NLVM Texture Features 112

Table 3.9. The Classification Results by Using The TF_1 Features of *NLVM.grl*, *NLVM.lht* and *NLVM.lvt* 113

Table 3.10. The Classification Confusion Matrix by Using The TF_1 and TF_2 Features of *NLVM.grl*, *NLVM.lht* and *NLVM.lvt* 114

Table 3.11. The Classification Confusion Matrix by Using The TF_1 and TF_2 Features of All The Defined NLVMs 115

Table 5.1. Texture Classification Results by Using 28 Dark and Bright Primitive Features from Methd A. 136

Table 5.2. Texture Classification Results by Using 14 Dark Primitive Features from Methd A. 136

Table 5.3. Texture Classification Results by Using 28 Dark and Bright Primitive Features from Methd B. 136

Table 5.4. Texture Classification Results by Using 28 Dark and Bright Primitive Features from Methd C. 136

Table 6.1. Texture Classification Confusion Matrix by Using All 32 Gabor Filtering Features 173

Table 6.2. Texture Classification Confusion Matrix by Using the 16 Channel Means 173

Table 6.3. Texture Classification Confusion Matrix by Using the 16 Channel Deviations 173

Table 6.4. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega_0 = 4$. 174

Table 6.5. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega_0 = 8$. 174

Table 6.6. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega_0 = 16$. 174

Table 6.7. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega_0 = 32$. 175

Table 6.8. Texture Classification Confusion Matrix by Using All 40 GLDM Features at d=1 and 2. 179

Table 6.9. Texture Classification Confusion Matrix by Using All 20 GLDM Features at d=1. 179

Table 6.10. Texture Classification Confusion Matrix by Using All 20 GLDM Features at d=2. 179

Table 6.11. Texture Classification Confusion Matrix by Using the 8 Energy Measures at d=1 and 2. 179

Table 6.12. Texture Classification Confusion Matrix by Using the 8 Entropy Measures at d=1 and 2. 180

Table 6.13. Texture Classification Confusion Matrix by Using the 8 Correlation Measures at d=1 and 2. 180

Table 6.14. Texture Classification Confusion Matrix by Using the 8 Local Homogeneity Measures at d=1 and 2. 180

Table 6.15. Texture Classification Confusion Matrix by Using the 8 Inertia Measures at d=1 and 2. 181

LIST OF ABBREVIATIONS

ARM	☞	AutoRegressive Model
CSA	☞	Converging Squares Algorithm
DCT	☞	Discrete Cosine Transform
DFT	☞	Discrete Fourier Transform
ERA	☞	Edge Repetition Array
FD	☞	Fractal Dimension
FFT	☞	Fast Fourier Transform
FIR	☞	Finite Impulse Response
FPSA	☞	Fourier Power Spectrum Analysis
FPT	☞	Fixed Percentile Thresholding
FSC	☞	Feature Space Clustering
FSM	☞	Fractal Surface Model
GF	☞	Gabor Filter/Filtering
GGLD	☞	Generalized Gray Level Dependence matrix
GIS	☞	Generalization of Intensity Segmentation
GLDA	☞	Gray Level Difference Approach
GLDM	☞	Gray Level Dependence Matrix
GLEA	☞	Gray Level Edge element Analysis
GLPA	☞	Gray Level Profile Analysis
GLPF	☞	Gaussian Low-Pass Filter/Filtering
GLRL	☞	Gray Level Run Length
GMRFM	☞	Gaussian Markov Random Field Model
GOP	☞	General Operator Approach
GRFM	☞	Gibbs Random Field Model

H-OF	☞	Higher-Order Fractal
HAFA	☞	Histogram and Autocorrelation Function Analysis
HVS	☞	Human Visual System
KLT	☞	Karhunen-Loeve Transform
LDCT	☞	Local Discrete Cosine Transform/mapping
LLCM	☞	Local Linear Convolution Method
LLVM	☞	Linear Local Vector Mapping
LPF	☞	Local Polynomial Fitting
LPT	☞	Local Polynomial Transform/mapping
LSF	☞	Local Surface Fitting
LVM	☞	Local Vector Mapping
MADAC	☞	Minimal Aggregate Difference Absolute Classifier
MADSC	☞	Minimal Aggregate Difference Square Classifier
MAP	☞	Maximum *A Posteriori*
MIT	☞	Moment-Invariant Thresholding
ML	☞	Maximal Likelyhood
MMADAC	☞	Modified Minimal Aggregate Difference Absolute Classifier
MRFM	☞	Mosaic Random Field Model
MSFM	☞	Multichannel Spatial Filtering Model
MUCTED	☞	MUlti-Channel Texture Edge Detection
NLVM	☞	Non-linear Local Vector Mapping
PWD	☞	Pseudo-Wigner Distribution
RSST	☞	Recursive Shortest Spanning Tree
RV	☞	Random Variable
RWM	☞	Random Walk Model

SEA	☞	State Estimation Approaches
SFA	☞	Spatial Filtering Approaches
SFT	☞	Spatial Filtering Technique
SLVM	☞	Successive Local Vector Mapping
SNR	☞	Signal to Noise Ratio
STC	☞	STatistical Classification
STS	☞	Supervised Texture Segmentation
TBM	☞	Texton-Based Method
TEA	☞	Theoretical Estimation Approach
TEXTED	☞	TEXTure Edge Detection
TGO	☞	Texture Gradient Operator
UNITRE	☞	UNIform Texture Region Extraction
UTS	☞	Unsupervised Texture Segmentation
pdf	☞	probability density function

STATEMENT OF ORIGINALITY

To the best knowledge of the author, the following aspects of the thesis are believed to be original:

1. With respect to the definitions for texture of Chapter 1:
    a. The proposal of an alterative definition for texture, which is applicable under a much wider spectrum of circumstances than either of the two commonly accepted definitions (i.e., statistical and structural definitions).
2. With respect to the texture feature extraction and classification procedure of Chapter 3:
    a. The introduction of local vector mapping, a conceptually new scheme for texture feature extraction.
    b. The unification and the alternative interpretation of some existing approaches towards texture feature extraction and classification under the general LVM scheme. In principle, all local-operation-based approaches to texture feature extraction such as LLCM approaches can be conveniently and compactly formulated as LVM problems.
    c. The definition and implementation of some concrete LVMs such as the local discrete cosine mapping (LDCT), local polynomial mapping (LPT) and some non-linear local vector mappings.
    d. The definition and extraction of texture features based on LVM operations. These features or LVM texture features are defined to be the first four central moments of the LVMed images, and are interpreted in terms of visual texture features such as coarseness and directionality.
    e. The application of LVM texture features in texture classification. The high classification accuracies achieved indicate the capability of LVM of statistical complexity reduction, i.e., mapping higher-order statistic differences into lower-order statistic differences.
3. With respect to the local surface fitting procedure for texture feature extraction of Chapter 4:
    a. The introduction and description of the local surface fitting scheme for texture

feature extraction.

b. The interpretation of a LSF problem as a LVM problem, thus establishing the relationship between the LSF and LVM schemes.

4. With respect to the primitive analysis based approach of Chapter 5:

   a. The proposal of three simple techniques for primitive extraction.

   b. The recognition of the need for the joint application of several types of primitives in order to achieve better classification performance.

   c. The verification of the common assertion that structural texture analysis approaches are particularly effective for structural textures but much less so for stochastic textures.

5. With respect to the investigation of Gabor filters applied in texture feature extraction and classification of Chapter 6:

   a. The proposal of a cortical channel model, which attempts to model individual channels in the MSFM for the HVS in the context of pictorial information processing and feature extraction.

   b. The investigation on the frequency component resolution characteristics of the HVS under both stochastic and structural textures. The undertaken experiments demonstrate the limited resolution of the HVS for frequency components, and the significance of the spectral peaks in the human visual perception of textures.

   c. The design of channel and filtering pattern for texture feature extraction based on the proposed cortical channel model. The chosen filtering pattern consists of 16 parallel channels, each of which is tuned to a specific band of spatial frequency and orientation.

   d. The development of a new procedure for texture feature extraction and classification. The defined texture features or GF features are the mean values and the standard deviations of the 16 channel output images. These features are interpreted in terms of some visual texture features.

   e. The comparison between the proposed GF features and the conventional GLDM

features in terms of discrimination capability, computational complexity, noise sensitivity and texture sample-invariance.

6. With respect to the texture segmentation review of Chapter 7:

   a. An updated survey of the approaches towards texture segmentation found in the open literature; the classification of UNITRE approaches into five major categories (i.e., FSC, STC, SEA, GIS and SFA), and the classification of TEXTED approaches into four major categories (i.e., TGO, SFT, TEA and TBM).

7. With respect to the investigation on texture segmentation based on LVM of Chapter 8:

   a. The development and successful implementation of a UNITRE algorithm based on a single LVM operation.

   b. The recognition of the need for the application of Gaussian low-pass filtering before final segmentation in order to obtain minimum uncertainties around texture edges.

   c. The recognition of the necessity of applying successive LVM operations in order to map differences in higher-order statistics (e.g., second-order statistics) into differences in the lowest-order statistics (i.e., the average intensities).

   d. The development and successful implementation of a UNITRE algorithm based on successive LVM operations; the illustration of the effectiveness of LLVM+local variance mapping in texture segmentation.

   e. The recognition of the need for the parallel application of multiple LVMs or SLVMs in texture segmentation, and subsequently the development and successful implementation of the MUCTED algorithm for texture edge detection.

8. With respect to the multi-channel texture edge detection procedure of Chapter 9:

   a. The application of the cortical channel model in texture edge detection.

   b. The theoretical justification of the proposed cortical channel model in texture analysis, and the confirmation of the need for the application of Gabor filters in the model.

   c. The development and successful implementation of a new multi-channel texture edge detection algorithm based on the proposed cortical channel model; the number of

channels and filter parameters are automatically determined by means of spectral peak identification.

1. INTRODUCTION

As a result of rapid development in computer technology, image processing and analysis has experienced vigorous growth in the past two decades. At the abstract level, an image is an impression of a real-world scene or our environment. Technically, the term *monochrome image* or simply *image* refers to a two-dimensional light intensity function $f(x,y)$, where x and y denote spatial coordinates and the value f at any point (x,y) is proportional to the brightness or *gray level* of the image at that point. For computer implementation of image processing and analysis algorithms, an image has to be first discretized both in spatial coordinates (x,y) and in the brightness f. The resultant version is usually denoted by $f(i,j)$ and is called a *digital image* or simply an image. A digital image can be considered as a matrix whose row and column indices i and j specify a point in the image and the corresponding matrix element value identifies the gray level at that point. The elements of such a 2-D digital array are called *image elements, picture elements, pixels,* or *pels*, with the last two names being commonly used abbreviations of "picture elements". The term *pixel* will be extensively used in this thesis.

The lowest-level image property is probably pixel gray level or pixel intensity. Any pixel can be perceived as dark (having relatively low intensity) or bright (having relatively high intensity) without explicitly referring to any other pixels. The inter-pixel relations such as line-likeness, coarseness, patchiness, regularity, or directionality are associated with a higher-level image property called *texture*. In fact, intensity property (or tonal property) and texture property are the two most fundamental properties of any monochrome image. Nevertheless, intensity and texture are not two independent concepts. They bear an inextricable relation very much like that between a particle and a wave [1]. When a small region of little intensity variation is observed, tonal property dominates texture property. On the other hand, when a region of wide intensity variations is considered, texture property is predominant. At an extreme, when a single pixel is examined, the only meaningful property is intensity. In this thesis, our attention is focused on texture property.

1.1 What is Texture?

Texture is an important characteristic for the analysis of many types of images. It can be seen in a very wide spectrum of images ranging from aircraft or satellite multispectral scanner

images to microscopic images of cell structures or tissue samples. It is indeed ubiquitous. Despite its importance and ubiquity in image data, a formal or precise definition for texture does not exist or at least has not been established [1]. This definition dilemma is, we think, mainly due to the vast diversity of texture samples observable in the nature, and also due to our incomplete understanding of the mechanisms of texture perception of ourselves. After all, how may a dictionary define texture literally? Two explanations may be found in a dictionary [2]:

1. Texture is a natural structure having an appearance or consistence as if woven, a tissue or a web (of a spider);
2. Texture is an arrangement of small constituent parts, perceived structure of skin, rock, soil, organic tissue, literal work, etc..

Technically, two similar definitions may be identified in the open literature on texture [3]:

A. Texture is defined to be a stochastic field of homogeneous intensity variations;

B. Texture is defined to be a field of a set of basic patterns/elements or primitives arranged according to some placement rules or grammars.

Definition A, called *statistical definition*, concentrates on the global impression a texture field offers to a human observer whereas Definition B, known as *structural definition*, emphasizes how a texture field is constructed. Neither of these two definitions is universal in that, statistical definition is inappropriate for textures of regular arrangements of sub-patterns having inhomogeneous intensity variations, and structural definition is not pertinent for textures generated from random mechanisms where no clear sub-patterns or primitives can be defined.

Texture can be perceived as being smooth or rough, coarse or fine, regular or irregular (random), directional or non-directional, etc.. Fig.1.1 shows four pairs of textures (a) D32 and (b) D12, (c) D31 and (d) D93, (e) D34 and (f) D110, and (g) D15 and (h) D111 taken from **Brodatz** [4]. The first pair D32 and D12, corresponding to a piece of Pressed Cork and Tree Bark respectively, illustrate the differences in smoothness or roughness. Clearly, D32 is felt much smoother than D12 (because it has been "pressed"). The second pair Beach Pebbles (D31) and Calf Hide (D93) show the differences in coarseness. In D31, very large regions of nearly

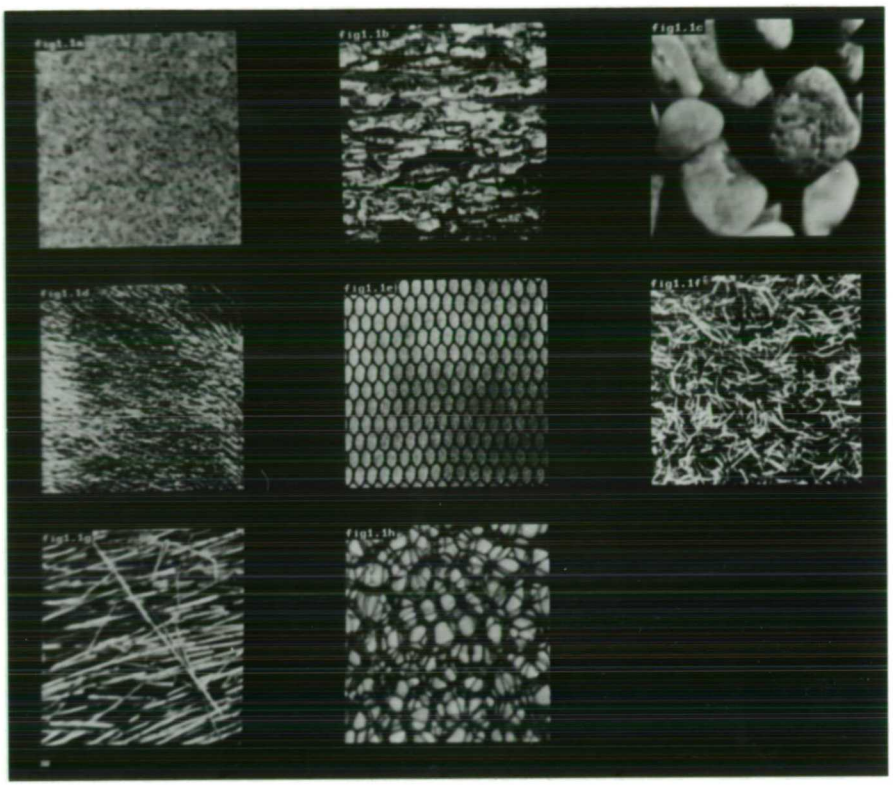


Figure 1.1. Illustration of Texture Smoothness, Coarseness, Regularity and Directionality. Shown are textures (a) D32, (b) D12, (c) D31, (d) D93, (e) D34, (f) D110, (g) D15 and (h) D111.

constant intensities can be seen (thus coarser) whereas in D93, the intensities from pixel to pixel change significantly (thus finer). The third pair D34 (Netting) and D110 (Handmade Paper made of grassy fiber) give examples of a regular texture (D34) and an irregular or random texture (D110). Finally, the last pair Straw (D15) and Plastic Bubbles (D111) demonstrate the differences in directionality.

The vast diversity of textures may be loosely divided into three categories namely *deterministic textures, structural textures* and *stochastic textures*. A deterministic texture is a texture which can be precisely described by a mathematical formula or a texture which is

composed of regular arrangements of well-defined sub-patterns or primitives according to well-defined placement rules. Examples are textures generated from 2-D mathematical functions and some man-made structures like the chessboard texture. A structural texture is a texture which is a layout of certain sub-patterns or primitives according to some placement rules and in which spatial variations on primitives and placement rules are permitted. An extreme in this category is the case where primitives are randomly placed and no placement rules are inferred. Finally, a stochastic texture is a texture which is a homogeneous 2-D data field that does not possess any extractable primitives or/and placement rules. Fig.1.2 presents a texture set which contains samples from all of these three categories.

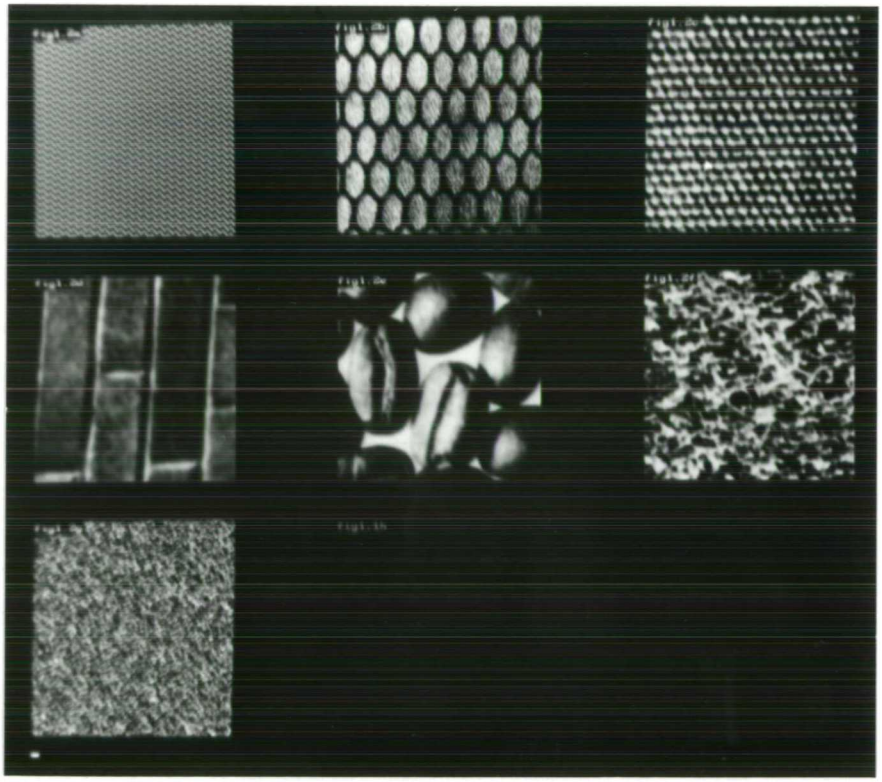


Figure 1.2. Illustration of Deterministic, Structural and Stochastic Textures.

For instance, Fig.1.2(a) is an artificial texture generated from a 2-D sinusoidal function and is

uniquely specified by the underlying sinusoid. Fig.1.2(b) (D34) is a deterministic texture as it can be easily and unambiguously described as follows: D34 is a regular layout of identical hexagons, and every hexagon neighbors 6 other hexagons by sharing a common boundary (or sideline) with each of its 6 neighbor hexagons. Fig.1.2(c) to (e) are textures D77 (Cotton Canvas), D94 (Brick Wall) and D74 (Coffee Beans). They are examples of structural textures since sub-patterns or primitives (like a single coffee bean or a single wall brick) are easily identifiable in these textures. It is worthwhile noticing that no placement rules can be inferred from D74. Fig.1.2(f) is another texture from [4]. It is a texture of Grass Lawn (D9). Obviously, while it is homogeneous, no clear sub-patterns can be extracted from it. Thus, it is a stochastic texture. An extreme of stochastic textures is a random field of IID (independently, identically distributed) random variables as shown in Fig.1.2(g). It is a Gaussian noise field.

An important characteristic of texture is its self-similarity which can be explained as follows. Imagine two portions of identical geometrical shape (say two square regions) are taken away from a given texture field, and swap the two regions to generate a "new" texture field. Then, self-similarity implies that this "new" texture field is visually identical to the original texture field. This is demonstrated in Fig.1.3. Fig.1.3(a) shows the original texture and the two square regions to be swapped. Fig.1.3(b) is the image after region swapping. This "new" texture would be visually identical to the original had the subjective contour due to region alignment not existed. However, one should be convinced by glancing at Fig.1.3(c) and (d) in which the effect of subjective contour is reduced by means of Gaussian low-pass filtering. For fair comparison, the same amount of filtering has been applied on the original texture to get Fig.1.3(c). This spatial-invariance of texture prompts us to propose the following definition for texture:

Texture T is such a phenomenon that, for a small connected region R ($R \in T$) of some predefined area S, there always exists a resolution level R_l at which region R offers to a human observer the same impression as the entire region T would.

Clearly, for different resolution level R_l, the area S of the "observation" window R is necessarily different. We claim that this definition is more universal than either of the conventional definitions (statistical or structural) in that it is applicable to all kinds of textures (deterministic, structural or stochastic).

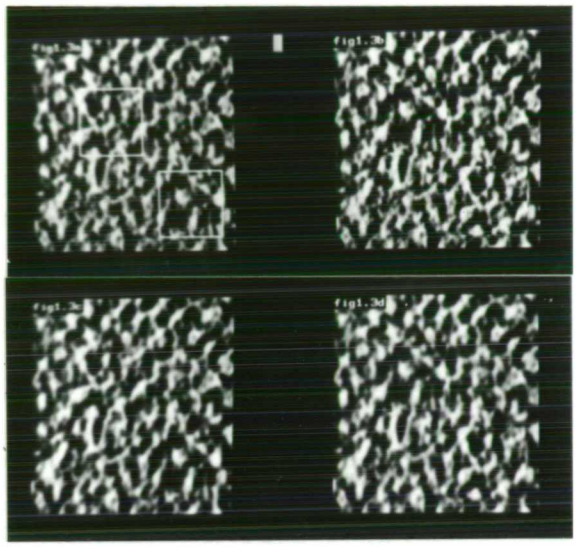


Figure 1.3. Illustration of Texture Self-Similarity. (a) Original image; (b) The image after region swapping; (c) The blurred version of (a) and (d) The blurred version of (b).

1.2 Why Texture Analysis?

So why does one pursue texture analysis? Let us approach this question by first looking at what texture analysis deals with. Over the years, it is generally agreed that the main issues of concern in texture analysis are [3]:

- **Texture Classification:** Given a texture sample, to which of a finite number of prototype textures does it belong?
- **Texture Description:** Given a texture sample, how can it be adequately described?
- **Texture Segmentation:** Given the image of a scene, how can the boundaries or texture edges between the major (differently) textured regions be established?

These three problems are listed in order of increasing difficulty. The first problem, also known as *texture recognition* or *texture discrimination*, is usually solved by means of texture feature extraction (data reduction) followed by standard pattern recognition, where after feature

extraction and before pattern recognition, the sample to be classified is represented by the extracted feature vector (a pattern). The second problem, being closely related to *texture synthesis* [5] which is concerned with how textures visually similar to parental samples can be generated, is more difficult because one could easily find two perceptually distinct textures and yet it could be still extremely difficult to describe the differences. The third problem is the most difficult one mainly because it is usually unknown at which level of complexity a texture is completed up to its physical boundary, and also because it is not well understood what kind of grouping mechanisms is required for identifying uniform texture regions.

From the point of view of human visual perception of texture, two different mechanisms have been suggested [6]. They are called **Preattentive Texture Perception** and **Attentive Texture Perception**. To be illustrative, let us consider an image shown in Fig.1.4.

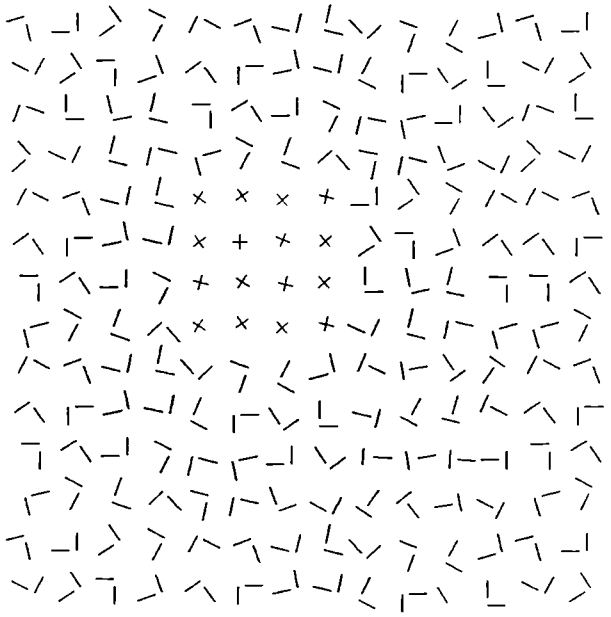


Figure 1.4. Illustration of Preattentive and Attentive Texture Perception.

At a glance, one may notice that an area made up of xs or +s stands out clearly from the surrounding background composed of random-rotated Ls. However, a detailed pixel-by-pixel inspection or scrutinization is indispensable in order to discover a third area containing random-rotated Ts at the bottom right of the image. The extraction of the xs region from the

surrounding region is called *preattentive texture segmentation* by means of preattentive texture perception, while the discovery of the Ts region from the background Ls region is named as *attentive texture segmentation* with the help of attentive texture perception. Preattentive and attentive texture perception would have been termed as *impressionistic* and *deliberate* texture perception in one of the earliest texture studies by **Pickett** [7]. Since preattentive perception is predominant in human visual perception especially at its early stage [6], our attention in this thesis is mainly concerned with preattentive texture perception.

Now, let us back to the initial question of why texture analysis is pursued. As texture is a fundamental property of many types of images and as continuous advances in image analysis and processing have brought increasing benefit for industry, the motives for conducting texture analysis are apparent. It is believed that a solution to the texture analysis problem will greatly advance the entire image analysis/processing and pattern recognition fields and it will bring much benefit to many possible industrial applications [3]. The areas which require the application of texture analysis range from machine vision to cytology including machine/computer vision, object detection and pattern recognition, medical diagnosis, industrial automation, remote sensing, geological exploration, military surveillance and cell analysis. In machine/computer vision, texture analysis is a essential problem to be solved in any low-level vision systems which involve the segmentation and recognition of image regions of uniform texture to produce symbolic descriptions of the scene concerned for higher level vision systems [8]. In object detection and pattern recognition, the role of texture analysis can not be overestimated. In many cases, object detection and recognition is possible only by means of texture analysis as illustrated in Fig.1.5 [9]. It may be that the moth in the image could be identified by its silhouette, but discovering that silhouette may first require detection of the difference in texture between the region occupied by the moth and the rest of the tree bark background. A similar example is given in Fig.1.6 which shows a fish at the ocean bottom [10]. The fish can adjust the spots on its own skin surface to match the texture of the surrounding. Clearly, the detection of the fish is possible only by invoking texture analysis. In biomedical engineering, diagnosis efficiency can be dramatically enhanced by means of texture analysis [11,12]. In industrial automation, texture analysis has been applied in material inspection and automatic quality control [14,15]. In remote sensing, texture analysis is useful for automatic land use classification, crop yield assessment and terrain classification [16,17,18]. In geological



Figure 1.5. A Moth at Rest on the Side of a Tree.

exploration, texture analysis can assist a geologist in rock type identification and classification [19], and finally in cytology, texture analysis has been used in cell discrimination and classification [13].

We have only listed the major examples of the applications of texture analysis. Other areas, known or to be known, could have been mentioned where texture analysis can be applied.

1.3 Objectives of This Thesis

Undoubtedly, before any practical application is possible, the availability of dedicated algorithms for texture analysis is essential. This thesis is devoted to the development of effective and efficient algorithms for texture analysis. Since texture description is more related to another worthwhile research area texture synthesis, our interests here lie primarily in the

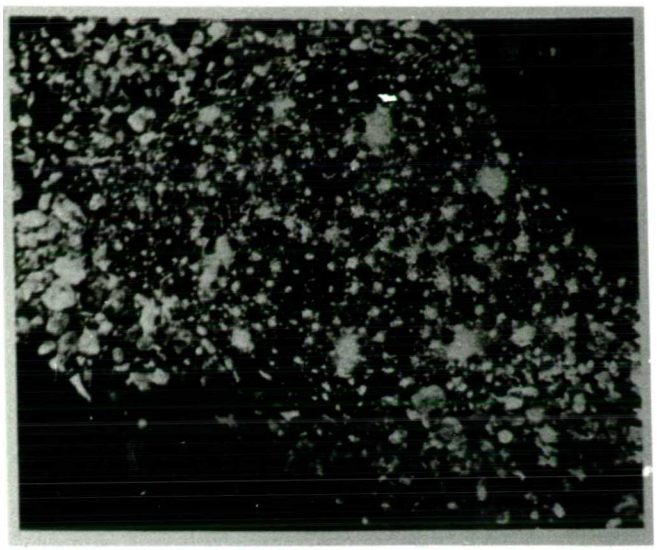


Figure 1.6. A Fish in the Background of Sand.

other two issues in texture analysis, i.e., texture classification and texture segmentation as identified in the preceding section.

Although extensive research on texture analysis has been conducted in the past two decades, most of the work is devoted to texture classification [20,21] with attention on texture segmentation being much more limited. Moreover, most of, if not all, the approaches taken previously are *ad hoc* in nature [1] and thus only achieved limited success. In part of the thesis, we review and examine most of the existing approaches towards texture in some detail. This kind of survey has two aims: firstly, it makes it clear that continuous efforts are required to advance texture analysis and to bring this interdiscipline to the point of practical applications as this thesis is intended to. Secondly, knowledge on some of the existing techniques is necessary for the presentation of the materials new to this thesis.

As texture is a regional image feature (in that it can be perceived neither at a single pixel nor at only a few pixels), the appropriateness of texture feature extraction methods based on some types of *local operations* is easy to see. In fact, a very large group of approaches reported in the open literature have been rooted on local operations [22] in texture classification. In this thesis, we attempt to unify and generalize these local-operation-based approaches by

introducing a new concept called **Local Vector Mapping (LVM)**. This concept also proves useful in the design of successful algorithms for texture segmentation as shown in the thesis.

Although a few existing approaches are shown to be successful in performance, they are in general of high computational complexity and thus not apt to practical implementation [21,23]. However, there does exist, in nature, a system which is both effective and efficient. This system is known as the **Human Visual System (HVS)**. Therefore, successful computational algorithms for texture analysis are seemingly bound to be those which imitate the way of the HVS in texture perception. To develop such algorithms, pertinent models for texture perception in the HVS have to be first sought. Over the years, some useful models have been established. In particular, the model hypothesized by **Compbell** and **Robson** [24,25] is now widely accepted. This model may be referred to as **Multichannel Spatial Filtering Model (MSFM)**. This thesis is also intended to investigate the appropriateness of this model in both texture classification and texture segmentation.

In summary, the objectives of this thesis are:

- To review and examine the existing approaches towards texture classification and segmentation (Chapters 2 and 7);
- To unify and generalize some of the existing approaches towards texture analysis based on local operations (Chapter 3);
- To develop new effective and efficient algorithms for texture feature extraction and classification (Chapters 3, 4, 5 and 6);
- To develop new effective and efficient algorithms for texture segmentation (Chapters 8 and 9);
- To identify worthwhile directions of further research (Chapter 10).

1.4 Organization of The Thesis

The main body of this thesis is composed of 10 chapters including this introductory chapter. Chapters 2, 3, 4, 5 and 6 deal with texture feature extraction and classification, Chapters 7, 8 and 9 with texture segmentation whereas Chapters 1 and 10 serve as complementary chapters.

Because of its importance, enormous efforts are being made to solve the texture analysis problem. This is particularly true in the context of texture feature extraction and classification. Three major categories of the approaches to texture feature extraction and classification may be identified, namely *Classical Statistical Approaches, Structural Approaches* and *Model-Based Approaches* [22]. Approaches belonging to the first category are gray level dependence matrix, generalized spatial dependence matrix, histogram and autocorrelation function analysis, Fourier power spectrum analysis, the gray level difference approach, gray level edge element analysis, gray level run length, gray level profile analysis, and the local linear convolution method. Structural Approaches include weak texture analysis, strong texture analysis and syntactic approach. Finally, in the third category, models currently used include time series model, Markov random field model, Gibbs random field model, mosaic random field model, random walk model, fractal surface model and human visual model. All these approaches are reviewed and described in Chapter 2. The principles, the merits and the weaknesses of these approaches are briefly stated in this chapter.

In Chapter 3, a new concept is introduced. This concept, formally termed as Local Vector Mapping (LVM), facilitates the unification and the subsequent generalization of some of the existing approaches to texture analysis as reviewed in the preceding chapter. The three components of a LVM, namely local neighborhood specification, vector formation and vector mapping, are described and explained in detail. Under the general framework of LVM, some concrete and useful LVMs are defined in this chapter. Procedures and experimental results for texture feature extraction and classification are also presented.

Texture is a surface characteristic as well as a local regional image property. Different physical surfaces will certainly result in distinct 2-D texture surfaces. These considerations prompt us to hypothesize that local surface description is useful for texture analysis [26]. This hypothesis is pursued and confirmed in Chapter 4. Both the general framework and concrete examples are presented.

Both texture analysis approaches described in Chapter 3 and Chapter 4 are statistical in nature in that they attempt to extract texture statistics in some way, and make no efforts to examine how the textures under study are constructed. As a consequence, these approaches may not be effective for structural textures. Therefore, effective approaches for structural

textures are needed. Determined by their nature, structural approaches are well suited for this target [21]. However, existing structural approaches are in general complex in computation. Chapter 5 describes a weak texture analysis approach with reasonable computational requirements, which is particularly effective for structural texture discrimination.

Realistic applications will no doubt involve all sorts of textures from highly deterministic to highly random, and a desirable algorithm for texture feature extraction should be generally applicable. As the HVS is usually the most effective and efficient system for texture analysis, a successful texture classification algorithm would be the one mimicing the HVS. Chapter 6 presents such an algorithm. In this chapter, recent and relevant findings on human visual perception in physiology and psychophysics are first reviewed to facilitate the development of the new algorithm. Comparison between this new algorithm and a standard technique is also given in this chapter.

Investigations on texture segmentation start in Chapter 7 which presents a state-of-the-art review on the approaches towards the problem. Texture segmentation approaches are classified into two major categories, i.e., the UNITRE approaches and the TEXTED approaches. The former group includes feature space clustering, statistical classification, state estimation, generalization of gray level segmentation and spatial filtering; while the latter consists of texture gradient operator, spatial filtering, the theoretical estimation approach and the texton-based method. All these approaches are explained in this chapter. The principles, appropriate examples, the relative merits and weaknesses of these approaches are described in brief.

The extension of Chapter 3 to texture segmentation is presented in Chapter 8. The general texture segmentation problem is examined from two different angles. A number of successful algorithms are described in some detail in this chapter. Their principles and implementation results are given.

The extension of Chapter 6 to texture segmentation is presented in Chapter 9. As the input image concerned in this chapter contains more than one texture type, the extension is far from trivial. In fact, unlike in texture classification, one has to be confronted with the following peculiar problem in texture segmentation. Texture segmentation involves simultaneously both the sufficient discrimination of textures in adjacent regions and the accurate identification of the boundaries between these regions. Since texture is undefinable at a single pixel, these two tasks

are conflicting and are constrainted by the *uncertainty principle* as discussed in [27]. Some aspects of this uncertainty principle in texture segmentation are studied in this chapter, together with the description and implementation of a novel algorithm for texture edge detection.

Finally, Chapter 10 concludes this thesis with a summary of the materials presented, followed by an outline of some possible directions for further research.

1.5 References

[1] **R.M.Haralick**, *Statistical and Structural Approaches to Texture*, Proc. IEEE, vol.67, 1979, pp.786-804.

[2] **Oxford University**, *The Compact Edition of the Oxford English Dictionary*, Oxford: Oxford University Press, 1971.

[3] **H.Wechsler**, *Texture Analysis: A Survey*, Signal Processing, vol.2, 1980, pp.271-282.

[4] **P.Brodatz**, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, 1966.

[5] **A.Gagalowicz** and **Song De Ma**, *Sequential Synthesis of Natural Textures*, Comput. Vision, Graphics and Image Proc., vol.30, 1985,pp.289-315.

[6] **B.Julesz** and **J.R.Bergen**, *Textons, The Fundamental Elements in Preattentive Vision and Perception of Textures*, Bell Sys. Tech. Jour., vol.62, 1983, pp.1619-1645.

[7] **R.M.Pickett**, *Visual Analysis of Texture in the Detection and Recognition of Objects*, in B.S.Lipkin and A.Rosenfeld: *Picture Processing and Psychopictorics*, New York, Academic Press, 1970.

[8] **A. Rosenfeld**, *Image Analysis: Problems, Progress and Prospects*, Pattern Recognition, vol.17, pp.3-12, 1984.

[9] **H.B.Cott**, *Adaptive Coloration in Animals*, New York, Dover Pubs, 1957.

[10] **H.M.Raafat** and **A.K.C.Wong**, *A Texture Information-Directed Region Growing Algorithm for Image Segmentation and Region Classification*, Comput. Vision, Graphics and Image Proc., vol.43, 1988, pp.1-21.

[11] **D.Terzopoulos** and **S.W.Zucker**, *Detection of Osteogenesis Imperfecta by Automated Texture Analysis*, Comput. Graphics and Image Processing, vol.20, pp.229-243, 1982.

[12] **I.E.Magnin** *et al.*, *Mammographic Texture Analysis: an Evaluation of Risk for Developing Breast Cancer*, Opt. Eng., vol.25, pp.780-784, 6/1986.

[13] **G.H.Landeweerd** and **E.S.Gelsema**, *The Use of Nuclear Texture Parameters in The Automatic Analysis of Leukocytes*, Pattern Recognition, vol.10, 1978, pp.57-61.

[14] **H.Giebel** *et al.*, *A System for Automatic Inspection of Glass Bottle Using Texture Analysis Procedures*, Proc. of the 6th Inter. Conf. on Pattern Recognition, Munich, 1982, pp.1068-1070.

[15] **R.T.Cain**, *SURVEY: Automated Visual Inspection*, Comput. Vision, Graphics and Image Proc., vol.41, 1988, pp.346-381.

[16] **M.Nagao** *et al.*, *Agricultural Land Use Classification of Aerial Photographs by Histogram Similarity Method*, IEEE Proc. 3rd Int. Joint Conf. on Pattern Recognition, Coronado, Calif. Nov.8-11, 1976, pp.669-672.

[17] **R.W.Conners** *et al.*, *Segmentation of a High-Resolution Urban Scene Using Texture Operators*, Comput. Vision, Graphics and Image Proc., vol.25, pp.273-310, 1984.

[18] **J.S.Weszka** *et al.*, *A Comparative Study of Texture Measures for Terrain Classification*, IEEE Trans. Syst., Man, Cybern., vol.SMC-6, 1976, pp.269-285.

[19] **W.D.Stromberg** and **T.G.Farr**, *A Fourier-Based Textural Feature Extraction Procedure*, IEEE Trans. Geoscience and Remote Sensing, vol.GE-24, pp.722-731, 1986.

[20] **T.N.Tan**, *Texture Analysis Survey and Local Vector Mapping Algorithms*, Internal Report, TR-01-TNT, SPABSS/EE/ICST, London, 1987.

[21] **L.V.Gool** *et al.*, *Survey: Texture Analysis Anno 1983*, Comput. Vision, Graphics and Image Proc., vol. 29, 1985, pp.336-357.

[22] **T.N.Tan**, *LVM Image Texture Analysis: Texture Classification*, Internal Report, TR-02-TNT, SPABSS/EE/ICST, London, 1987.

[23] **T.N.Tan**, *Texture Edge Detection via Gabor Filtering*, Internal Report, TR-08-TNT, SPABSS/EE/ICST, London, 1988.

[24] **F.W.Compbell** and **J.G.Robson**, *Application of Fourier Analysis to The Visibility of Gratings*, J.Phy., vol.197, 1968, pp.551-566.

[25] **D.A.Pollen** and **S.F.Ronner**, *Visual Cortical Neurons as Localized Spatial Frequency Filters*, IEEE Trans. Syst. Man Cybern., vol.SMC-13, 1983, pp.907-916.

[26] **T.N.Tan** and **A.G.Constantinides**, *Texture Classification by Local Surface Fitting*, Proc of BPRA 4th Inter. Conf. on Pattern Recognition, pp.48-57, Springer-Verlag, 1988.

[27] **R.Wilson** and **G.H.Granlund**, *The Uncertainty Principles in Image Processing*, IEEE Trans. on Pattern Ana. Mach. Intell., vol.PAMI-6, 1984, pp.758-767.

2. TEXTURE CLASSIFICATION: A REVIEW

2.1 Introduction

Texture Classification has been the focus of interest in texture analysis for the last 20 years. The problem of texture classification can be briefly stated as follows. There are a finite number of prototype textures or texture classes C_i, $i = 1,2,...,N$, and a number of so-called training samples belonging to each class are available. Based on the information extracted from these known samples, a decision rule is designed which classifies a given texture sample of unknown class into one of the N classes. The important steps involved are texture feature extraction and classifier design (i.e., the decision rule design). As classifier design is well developed [40,80], the critical step is feature extraction, i.e., the choice of a set of texture features which reduce the dimension of input data to a computationally reasonable level while preserving as much of the classifying information present in the original data as possible.

Over the years, a wide variety of techniques have been proposed and developed for texture feature extraction and classification. We classify these techniques into three major categories, namely **Classical Statistical Approaches, Structural Approaches** and **Model-Based Approaches**. It should be pointed out that such a division is sometimes rather artificial as the distinction between some of the approaches is not always clear. Nevertheless, such a division will simplify our presentation of material. For other alternative classifications, we refer to [6,20,28]. In this chapter, we shall discuss various techniques under each category. To keep the size of this thesis within the normal PhD thesis standard, each technique can only be briefly described. Further details can be found in the references cited therein.

2.2 Classical Statistical Approaches

Statistical approaches are consistent with the statistical definition of texture (See Chapter 1). On this level, a texture is considered to be characterized by some set of statistics. These statistics can range from simple first-order statistics like histogram means to sophisticated higher-order statistics such as edge statistics and gray level run length statistics. Methods which can be included in Statistical Approaches are: gray level dependence matrix, generalized spatial dependence matrix, histogram and autocorrelation function analysis, Fourier power spectrum analysis, gray level difference approach, gray level edge element analysis, gray level run length,

gray level profile analysis, and local linear convolution method. For notation convenience, these methods will be hereafter denoted by the following abbreviations respectively: GLDM, GSDM, HAFA, FPSA, GLDA, GLEA, GLRL, GLPA and LLCM.

2.2.1 GLDM: Gray Level Dependence Matrix

Studies on human visual perception of texture suggest that texture discrimination is mostly invoked by the differences in second-order statistics [69]. This is probably the primary motive for the emergence of the GLDM technique for texture analysis. The first step in the GLDM technique is to compute the relative *joint probability* $P(i,j|\vec{\Delta})$ of gray level transition from i to j given the intersample spacing d and the spacing path angle θ, where $\vec{\Delta} = (d,\theta)$ is usually called the *displacement vector*. $P(i,j|\vec{\Delta})$ can also be interpreted as the relative frequency of the co-occurrence of two pixels at a distance of $\vec{\Delta}$ having gray levels i and j respectively. For a given displacement vector $\vec{\Delta}$, $P(i,j|\vec{\Delta})$ can be tabulated as a matrix, justifing the name of *gray level dependence matrix* or *gray level co-occurrence matrix*. If the distinction between opposite directions are ignored, then $P(i,j|\vec{\Delta})$ can be compactly defined as follows:

$$P(i,j|\vec{\Delta}) = \sum_m \sum_n \delta[i-f(m,n)]\delta[j-f(m\pm d\sin\theta, n\pm d\cos\theta)],\ \theta = 0^o, 90^o;$$

$$P(i,j|\vec{\Delta}) = \sum_m \sum_n \delta[i-f(m,n)]\delta[j-f(m+\pm\sqrt{2}d\sin\theta, n-\pm\sqrt{2}d\cos\theta)],\ \theta = 45^o, 135^o;$$

where $f(m,n)$ is the image concerned and $\delta[\bullet]$ is the Kronecker function. As $P(i,j|\vec{\Delta})$ itself is not in a form suitable for texture analysis, **Haralick** [25] proposed 14 features to be calculated from each $P(i,j|\vec{\Delta})$, but usually only a set of 5 out of these features are employed. These 5 features are called *energy, entropy, correlation, local homogeneity* and *inertia*. Their definitions are given in Section 6.5.1 in this thesis.

The effectiveness of GLDM has been extensively illustrated by a number of investigators both theoretically and empirically. The two comparison studies [27,37] have similarly concluded that GLDM is usually superior to other statistical approaches such as FPSA, GLRL and GLDA. To mention but a few examples of GLDM, **Terzopoulos and Zucker** [7] used GLDM features to detect Osteogenesis Imperfecta, which is a highly prevalent, genetically determined disease. They reported that an identification accuracy of 90% can be obtained by means of texture

analysis whereas the accuracy by means of visual inspection by medical experts is only 69%. **Chen** and **Pavlidis** [13] segmented a textured image into uniform texture regions based on the comparison between GLDM measures computed from adjacent windows. **Davis** [17] defined his *polarogram* of GLDM contrast measures for rotation-invariant texture classification, where a polarogram is a polar plot of an orientation-sensitive texture statistic versus orientation. In addition to the extraction of statistical measures, GLDM is also useful for extracting some structural information as shown in [90].

Recently, **He** *et al.* [63] proposed a set of 3 new GLDM features, namely *diagonal moment, high level moment* and *low level moment*, to substitute the classical Haralick's GLDM features.

To conclude this sub-section, a number of points need to be made. The computational requirements in GLDM are usually very high, and even impractical when the number of gray levels and image size are very large. This computational pain can be mitigated by reducing the number of gray levels at the price of possible textural information loss. In practice, one usually does not know *a prior* what displacement vector $\vec{\Delta}$ should be used, and a set of a few displacement vectors are often employed.

2.2.2 GSDM: Generalized Spatial Dependence Matrix

As pointed out by several authors [35,41], a statistical texture approach should take into consideration both the partition of an image into cells and subsequently the assignment of gray levels to these cells. The GLDM technique discussed in the preceding section is concerned mainly with the assignment aspect and as a consequence, further improvement on GLDM seems possible by considering some aspects of the image partitioning process. **Davis** *et al.* [35] attempted to improve the GLDM method by introducing the *Generalized Spatial Dependence Matrix (GSDM)*. Formally, in order to compute a GSDM, one has to specify an image feature prototype P, a spatial predicate S and a prototype attribute A. The feature prototype P is the image feature one will use. Examples are gray level intensity, gray level edges, lines, spots or other image primitives. The spatial predicate S is the constraint imposed on the spatial positions of the pixels under consideration, and A is the feature measure which will be used for indexing the GSDM. For instance, in the GLDM method, a special example of the GSDM method, P is the gray level intensity, S is the displacement vector $\vec{\Delta}$, and A is the gray level of the pixels.

As we did for GLDM, a set of texture features can be similarly calculated from each GSDM and used for texture analysis. **Terzopoulos** and **Zucker** [7] used GSDM as well as GLDM features to identify Osteogenesis Imperfecta. **Dyer** *et al.* [42] used GSDM features to classify textures where the feature prototype P was the local edge maxima. In [23], **Davis** *et al.* evaluated the GSDM method empirically. They computed GSDMs for three different feature prototypes namely intensity, pixel-edges and extended-edges. The effectiveness of these GSDMs in terms of texture classification was compared under three grouping techniques: nearest neighbor classifier, linear discriminant function and Battacharyya distance (B-distance) measure. The results obtained indicate that the GSDM based on pixel-edge gives the best performance, while the intensity GSDM is better than the extended-edge GSDM.

2.2.3 HAFA: Histogram and Autocorrelation Function Analysis

A histogram shows the relative frequency at which a particular intensity occurs in an image. For stationary and ergodic images, a histogram forms a reasonable approximation to the ensemble average as well as to the first-order probability density function of the underlying images. If an image $f(i,j)$ is defined on an MxN grid and quantized to L gray levels from 0 to $L-1$, then its normalized histogram $h(l)$ can be defined as:

$$h(l) = \frac{1}{MxN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \delta[f(i,j)-l], \quad l=0,1,\ldots, L-1$$

The autocorrelation function describes the spatial dependence of a pixel on its neighboring pixels. For a discrete image field $f(i,j)$, an estimate of the corresponding autocorrelation function is usually defined as

$$\rho(k,l) = \sum_{i} \sum_{j} f(i,j) f(i+k, j+l)$$

where (k,l) is the displacement w.r.t. (i,j).

The histogram contains first-order information of the image $f(i,j)$, while the autocorrelation function reflects the second-order statistics to some extent. Autocorrelation functions drop off slowly for coarse textures whereas rapidly for fine textures [6,28]. The appropriateness of the histogram and the autocorrelation function for texture analysis has been demonstrated by several investigators. **Gagalowicz** and **Song** [5,9], for instance, have successfully synthesized natural textures via control of the histogram and the autocorrelation of

the parent texture. **Nagao** *et al.* [47] used histogram measures alone to obtain agricultural land use classification in aerial photographs. Similarly, **Faugeras** *et al.* [81] and **Wang** *et al.* [21] approached the texture analysis problem by using local or global histogram and autocorrelation features.

It should be pointed out, however, that a histogram contains no information about spatial correlations between pixels, and even the joint application of histogram and autocorrelation function is sometimes insufficient for texture analysis as discussed in [14].

2.2.4 FPSA: Fourier Power Spectrum Analysis

An alterative to spatial domain analysis is frequency domain analysis. Among the various transform techniques, Fourier transform analysis is no doubt the most widely used. Since coarse (fine) textures signify relatively slow (quick) intensity variations in the spatial domain and thus rich low (high) frequency components in the frequency domain, Fourier power spectrum analysis is an obvious vehicle for texture analysis. Conventionally, texture features in FPSA are defined to be the *summed energy* measures in some regions on the spectral domain. Some commonly employed features extracted this way include:

1. annular-ring sample geometry, which measures the texture coarseness and is defined as

$$F(\rho_1,\rho_2) = \int_{\rho_1}^{\rho_2} \int_{0}^{2\pi} H(\rho,\theta) d\theta d\rho$$

   where $H(\rho,\theta)$ is the Fourier power spectrum in polar form.

2. wedge sampling geometry, which contains directionality information and is described by

$$F(\theta_1,\theta_2) = \int_{\theta_1}^{\theta_2} \int_{0}^{\infty} H(\rho,\theta) d\rho d\theta$$

3. parallel-slit sampling geometry, which is used less frequently and defined as

$$F(\rho_1,\rho_2,\theta_1,\theta_2) = \int_{\theta_1}^{\theta_2} \int_{\rho_1}^{\rho_2} H(\rho,\theta) d\rho d\theta$$

These conventional features are subsequently criticized for their poor discriminating power [27,37]. This inferiority is not due to the insufficiency of Fourier power spectrum for texture

description but is rather related to the way of defining spectral features and of computing the Fourier spectrum itself as investigated in **Dastous *et al.*** [82] and **Chen** [83].They suggest a reappraisal of FPSA.

Several remarks are now in order regarding the use of FPSA. In FPSA, no consideration has been taken into account on the phase aspect of the underlying texture samples. The neglect seems reasonable as the phase spectrum is not shown to convey much useful information for texture discrimination [12,84,85]. Secondly, the autocorrelation function and the Fourier power spectrum form a Fourier transform pair for stationary images, and their applications in texture analysis are in principle equivalent. And finally, although Fourier power spectrum can reflect some global features like directionality and regularity, it is incapable of representing local features, which are believed to be critical for preattentive texture perception [69]. This problem may be overcome to a certain extent by applying the Fourier transform to local regions [86].

2.2.5 GLDA: Gray Level Difference Approach

Let $f(i,j)$ be a digital image quantized to gray levels 0,1,..., and $L-1$, then the normalized histogram of the absolute difference image $\Delta f(i,j) = |f(i+\Delta i, j+\Delta j) - f(i,j)|$ can be defined as follows:

$$h(l\,|\,\vec{\Delta}) = \frac{\sum_i \sum_j \delta[l - |f(i+\Delta i, j+\Delta j) - f(i,j)|]}{\sum_l \sum_i \sum_j \delta[l - |f(i+\Delta i, j+\Delta j) - f(i,j)|]}$$

where $\vec{\Delta} = (\Delta i, \Delta j)$ is the displacement vector w.r.t. pixel (i,j). $h(l\,|\,\vec{\Delta})$ describes the number of occurrences of two pixels at a distance of $\vec{\Delta}$ differing in intensity by l levels. Several texture features can be extracted from $h(l\,|\,\vec{\Delta})$. These include the mean, the standard deviation, the skewness, the Kurtosis and some other shape descriptors of $h(l\,|\,\vec{\Delta})$.

The appropriateness of GLDA for texture description is easy to see. For coarse textures, the intensity transitions from pixel to pixel are mostly slow or gradual. Consequently, the difference image $\Delta f(i,j)$ will usually have a small dynamic range and the histogram $h(l\,|\,\vec{\Delta})$ will pick up towards $l = 0$. For fine textures, the situation is opposite. In this case, the difference image will have a large dynamic range and the corresponding histogram will be more spread out. The effectiveness of GLDA has been studied both theoretically and empirically [27,37].

In practice, a single displacement vector $\vec{\Delta}$ is rarely adequate. A set of displacement vectors $\vec{\Delta}_1$, $\vec{\Delta}_2$, ..., and $\vec{\Delta}_N$ are usually employed. It is also worthwhile noting that $h(l|\vec{\Delta})$ can be derived from the gray level co-occurrence matrix under the same displacement vector by summing up the entries along axes parallel and symmetrical with respect to the main diagonal of the matrix.

2.2.6 GLEA: Gray Level Edge Element Analysis

A gray level edge can be perceived whenever the gray level continuity is violated. Gray level edges are important clues for human visual perception [78]. The occurrence frequency of gray level edges is strongly related to texture coarseness. Meanwhile, the distribution of edge orientations is indicative of texture directionality. If most of the edge elements in a given texture share a common orientation, the texture will be perceived as being directional.

As early as 1971, **Rosenfeld** and **Thurston** [45] suggested that the edges per unit area is a useful texture measure. To extract edge elements, they applied the quick Roberts gradient operator specified by

$$P(i,j) = |f(i-1,j-1)-f(i+1,j+1)| + |f(i-1,j+1)-f(i+1,j-1)|$$

In [38], **Sutton** and **Hall** extended Rosenfeld and Thurston's idea by making the gradient a function of pixel displacement. Thus, for every possible displacement d and pixel set $(i,j)\in N$, they computed

$$g(d) = \sum_{(i,j)\in N} \Big[|f(i,j)-f(i+d,j)| + |f(i,j)-f(i-d,j)| + |f(i,j)-f(i,j+d)| + |f(i,j)-f(i,j-d)| \Big]$$

They then used the curve $g(d)$ as a texture measure in a pulmonary disease identification experiment and obtained 80% accuracy for identifying normal and abnormal lungs.

Dyer *et al.* [42] calculated their co-occurrence matrices based on edge maxima while **Hong** *et al.* [43] extracted useful texture primitives based on edge element analysis. More recently, **Vilnrotter** *et al.* [87] developed a structural texture analysis approach also based on edge element analysis.

2.2.7 GLRL: Gray Level Run Length

In GLRL, a gray level run is a set of linearly adjacent (usually connected) pixels having the same gray level along a defined direction θ. The length of a run is the total number of pixels in that run. Intuitively, coarse textures are more likely to have long runs whereas fine textures to have short runs. To apply GLRL, one has to first compute the relative frequency $f(i,j|\theta)$ at which a run having length j and gray level i occurs. For a given direction θ, the relative frequency function $f(i,j|\theta)$ can be tabulated as a matrix, the so-called *GLRL matrix*. The element at (i,j) in the matrix is $f(i,j|\theta)$. Being a 2-D array of numbers, the GLRL matrix itself is not apt to direct application for texture analysis. Instead, some measures are usually calculated from it. The most commonly used measures are the following [28,34]:

A. SRE: Short Run Emphasis Inverse Moments

$$SRE(\theta) = \sum_i \sum_j \frac{f(i,j|\theta)}{j^2} \Big/ \sum_i \sum_j f(i,j|\theta)$$

B. LRE: Long Run Emphasis Moments

$$LRE(\theta) = \sum_i \sum_j j^2 f(i,j|\theta) \Big/ \sum_i \sum_j f(i,j|\theta)$$

C. GLN: Gray Level Nonuniformity

$$GTN(\theta) = \sum_i \sum_j f(i,j|\theta)^2 \Big/ \sum_i \sum_j f(i,j|\theta)$$

D. RLN: Run Length Nonuniformity

$$RLN(\theta) = \sum_j \sum_i f(i,j|\theta)^2 \Big/ \sum_i \sum_j f(i,j|\theta)$$

E. FOI: Fraction of Image in Runs

$$FOI(\theta) = \sum_i \sum_j f(i,j|\theta) \Big/ \sum_i \sum_j j f(i,j|\theta)$$

Using these features in conjunction with four directions 0^0,45^0,90^0 and 135^0, **Galloway** [34] demonstrated that about 83% classification accuracy can be obtained in classifying swamp, lake, railroad, orchard, scrub and suburb. On the other hand, the two comparison studies [27] and [37] have consistently concluded that GLRL is usually inferior to other methods such as GLDA and GLDM.

2.2.8 GLPA: Gray Level Profile Analysis

A 1-D gray level profile is the pixel intensity plot versus pixel position. A characteristic of the HVS is the intuition that important texture information is contained in the relative frequency of local extrema of various sizes in gray level profiles. Thus useful texture measures may be extracted from these profiles. **Rosenfeld** and **Troy** [88], for instance, suggested the number of local extrema per unit length as a texture measure where local extrema are defined as

- Local Maximum: $f(i) \geq f(i+j), \quad |j| \leq d$
- Local Minimum: $f(i) \leq f(i+j), \quad |j| \leq d$

with $f(i)$ being the 1-D gray level profile. By defining extrema as pixels having intensity above or below some threshold T, **Mitchell** *et al.* [89] suggested the curve of the number of extrema versus T as a texture feature.

Instead of examining extrema along profiles, **Barba** and **Rosin** [54] performed a so-called *curvilinear integration* of the gray level profile along some half scan lines starting from each pixel and in different directions. The scan line terminates when the integration reaches a preset threshold. Texture measures are then defined as the vector of the displacements along the scan lines of different directions. Conventionally, 1-D gray level profiles are obtained through row or column concatenation. In [59], **Nguyen** and **Quinqueton** proposed the use of *Peano scanning* [62] in constructing gray level profiles within local windows. They used the irregularity measures of the resultant profiles as texture features.

2.2.9 LLCM: Local Linear Convolution Method

Texture is a local image property and some kinds of local operations are intuitively useful for extracting texture measures. The LLCM technique to be discussed in this section is of this

type. In a general LLCM technique, the basic steps are the determination of convolution window or mask sizes, the choice of mask weights and the definition of texture measures. The choice of mask weights is usually the most important step.

Laws [48] is one of the initiators of LLCM. He chose a set of linear convolution masks empirically. These masks have been successfully applied to texture classification and segmentation. In his approach, Laws took two steps. Microstatistic features were first computed using small convolution masks such as 3x3 or 5x5, and macrostatistic features were then calculated over large windows. The most useful features were found to be the sums of the squared micro-features which Laws termed as *texture energy measures*, or the sums of the absolute micro-features which are usually preferable because of cheaper computation and yet maintaining the performance. In [49], **Pietikainen** *et al.* pointed out experimentally that the effectiveness of Laws' convolution masks depends on their general forms such as edge-like, spot-like, line-like etc., rather than on the specific numerical mask weights. In [19], **Ade** modified the Laws' approach in that the convolution masks are adaptively and automatically generated. In each 3x3 window, he formed a 9-component feature vector consisting of the 9 window pixel elements and attached the vector to the central pixel of the window. After estimating the variance-covariance matrix of this feature vector, he subsequently computed the *eigenvectors* of the resultant matrix. To extract texture features, he then used these eigenvectors as convolution masks or *eigenfilters*.

The eigenfilter approach may be criticized for its high computational demand. This is particularly true for large windows as one has to solve an eigenvector problem of high dimension. Instead of looking for eigenvectors, **Unser** [15] proposed to approach the problem of texture measurement by taking some local linear transforms upon each window feature vector. He found that the variances of the transformed images are very effective texture measures.

Another example of LLCM is the General Operator Approach (GOP) of **Granlund** [56]. The GOP involves the application of a set of linear convolution masks. Some of these masks are templates of lines and edges. Granlund suggested the repeated application of convolution masks in order to map the textural differences into gray level differences.

2.3 Structural Approaches

On the structural level, a texture is considered to be composed of some basic subpatterns or textural primitives which are easily identifiable and repetitive over the entire texture field. Such an assumption implies that a texture field can be sufficiently characterized by these subpatterns and their spatial arrangements. Thus a general structural texture analysis approach can involve the following steps:

- Texture Primitive Definition and Extraction;
- Primitive Description and Characterization;
- Primitive Interaction Description and Characterization.

A primitive or basic textural element or *texel* in structural texture analysis is a very important concept and can be defined as a set of pixels (usually connected) satisfying certain predicates. A pixel, for example, together with its gray level, is the simplest primitive. Structural texture analysis approaches found in the open literature differ from each other mainly in the actual contents of the three steps listed above or in the order they are followed. In particular, these approaches can be grouped into three classes, namely Strong Texture Analysis, Weak Texture Analysis and Syntactic Texture Analysis.

2.3.1 Strong Texture Analysis

Strong textures are those which have strong or non-random spatial interactions between primitives. To be adequate, a strong texture analysis approach needs to execute all the three steps mentioned above. However, the order these steps are followed is not unique. If a method starts with primitive extraction and description, followed by primitive interaction description or placement rule characterization, it will be referred to as a *bottom-up* approach. If, on the contrary, it recognizes the texture spatial structure or spatial placement rule before primitive extraction and description, it will be called a *top-down* approach.

In top-down approaches, primitive placement rules are first computed and the spatial grid structure of texture is derived. Primitive extraction is subsequently guided. **Conners** and **Harlow** [51], for instance, specified the spatial grid structure by two *periodicity vectors* which are chosen as the displacement vectors of the gray level co-occurrence matrices with the

smallest moments of inertia. On the other hand, **Matsuyama** *et al.* [58] extracted two similar periodicity vectors based on the energy distribution in the Fourier power spectrum. Recently, **Vilnrotter** *et al.* [87] present another example of top-down strong texture analysis approaches. They first applied an intensity edge detector to the input texture and generated an edge map. From the edge map, they calculated the so-called *Edge Repetition Arrays* (ERAs) which are similar to the gray level co-occurrence matrices. From some selected ERAs, they then extracted the spatial grid structure and placement rules before the corresponding primitives are finally extracted.

In a bottom-up approach, primitive extraction, primitive description and primitive interaction description (i.e., characterization of placement rules) are sequentially performed. **Matsuyama** *et al.* [55], for example, first extracted texture primitives by means of cluster analysis and then extracted the *regularity vectors* which specify the primitive displacement rules. A similar methodology is followed by **Tomita** *et al.* [91]. They define primitives to be image regions of homogeneous intensities. Thus, primitive extraction is equivalent to gray image segmentation which is achieved in their paper via a simple merging technique. After primitive extraction, they perform shape analysis on primitives and calculate primitive attributes such as brightness, area, size, elongatedness and curvature. Based on attribute histograms, they classify primitives into one ground class and several figure classes by a clustering technique. Finally, placement rules are extracted based on some second-order statistics of typical primitives from each class.

In passing, two remarks on strong texture analysis approaches are in order. Usually, primitive extraction is a very difficult process in bottom-up approaches and may be sensitive to various degradations of images as noted in [58]. On the other hand, primitive extraction could be a very trivial task in top-down approaches, but these approaches may fail completely if the underlying spatial structures are very complex.

2.3.2 Weak Texture Analysis

Weak textures are those in which primitives are only weakly coupled or randomly placed and the underlying placement rules convey little textural information. As such, some simple statistics such as histograms and co-occurrence matrices of primitives may be sufficient for weak texture discrimination. Therefore, a weak texture analysis approach may only involve

primitive extraction and description. **Hong** *et al.* [43] is an example of weak texture analysis approaches. They define primitives as dark regions bounded by connected edges on a bright background. Thus, they first applied 8 local edge operators to the input texture sample corresponding to the 8 directions in a 3x3 window. The resultant edge maps are "cleaned" by grouping appropriate edge points into connected boundaries. After primitive extraction (i.e., extraction of dark regions), they compute some primitive attributes and form a set of attribute histograms. Then, the means and standard deviations of these histograms are taken as texture measures for texture discrimination. **Wang** *et al.* [26] present another example of weak texture analysis approaches. In this paper, they define primitives as image pixels having intensities above or below some threshold. They described three different ways of extracting primitives. Statistics of primitive attributes are used as texture features.

In addition to texture classification and discrimination, weak texture analysis has also found its applications in texture segmentation. A simple example is given in **Tsuji** and **Tomita** [50]. They define primitives to be image regions of constant intensities which they called *atomic regions*. Since different textures have distinct primitive attributes, they threshold the attribute histograms to segment the input image into uniform texture regions.

2.3.3 Syntactic Texture Analysis

In [32], **Zucker** proposed to regard realistic or natural textures as the result of some transformation of appropriate, highly structured, perfect texture patterns. According to Zucker's model, an ideal texture is a deep, unobservable perfect pattern in which local primitives (fundamental building blocks) are extended into a global structure such as a regular tessellation. He believes that transformation rules can be defined from a representation of an ideal texture to that of a natural texture. Zucker's model has enjoyed a good implementation in the work by **Lu** and **Fu** [33]. In [33], a texture sample is first divided into non-overlapped windows of size 9x9, and the pixel intensity relations within each window are represented with some fixed tree structures. Based on these tree structures, they derived a set of *production rules* which define the tree grammars for the underlying textures. These grammars can be then used for texture synthesis as well as texture discrimination. In order to model more realistic textures, they subsequently introduced some *distortions* on the production rules. More details on syntactic methods can be found in **Fu** [52].

2.4 Model-Based Approaches

Model-based approaches are relatively recent. Extensive investigation on model-based approaches began around 1980. Models currently used in texture analysis include Autoregressive Model (ARM), Markov Random Field Model (usually Gaussian, GMRFM), Gibbs Random Field Model (GRFM), Mosaic Random Field Model (MRFM), Random Walk Model (RWM), Fractal Surface Model (FSM) and Multichannel Spatial Filtering Model (MSFM). In this section, we shall examine each of these models by briefly stating their basic principles and applications in texture feature extraction and classification.

2.4.1 ARM: Autoregressive Model

Autoregressive (AR) models are a subset of time series models which stem from the automatic control and system analysis disciplines [39]. A general 2D AR model can be defined as

$$f(i,j) = \sum_{(k,l)\in\eta}\sum a_{kl} f(k,l) + w(i,j), \quad (i,j)\in\eta$$

Where $f(i,j)$ is the 2D texture field under study, $w(i,j)$ is a 2D noise field (usually assumed to be a stationary Gaussian process), a_{kl} are the model parameters, and η is usually called the *model supporting region* or a *neighborhood system* w.r.t. pixel (i,j). Clearly, a 2D AR model describes the *linear* dependence of a pixel on its neighboring pixels. The nature of a 2D AR model depends on the structure of η. In particular,

1. if $\{k<i\}$ or $\{ k=i, l<j \}$ for all $(k,l)\in\eta$, then the model is called a *Causal AR Model.*
2. if $\{ k<i \}$ or $\{l<j\}$ for all $(k,l)\in\eta$, then the model is called a *Semi-Causal AR Model*
3. if in η, there exists a (k,l) such that $\{k>i\}$ or $\{l>j\}$ or both is true, then the model is called a *Non-Causal AR Model.*

In practice, model parameters a_{kl} are usually estimated by minimizing the following squared error:

$$e(\vec{\theta}) = \sum_i\sum_j \Big[f(i,j) - \sum_{(k,l)\in\eta}\sum a_{kl} f(k,l) \Big]^2$$

where $\vec{\theta} = [a_{k,l}, (k,l)\in\eta]$ is the model parameter vector. More details on 2D AR models may be

found in [8,29].

Applications of AR models in texture analysis are numerous. To mention just a few examples, **Chellappa** and **Kashyap** [3] studied the applicability of 2D noncausal AR models to texture synthesis. For a parent texture (i.e., a texture to be synthesized), a 2D noncausal AR model is first chosen and model parameters are subsequently estimated. Then based on the model parameter estimates, samples visually similar to the parent textures are synthesized. They evaluated the similarity between the synthesized textures and the corresponding original textures by means of variograms. **Kashyap** and **Khotanzad** [36] used 2D AR models for texture classification. After fitting an AR model to a given texture, three texture features, which are rotation-invariant, are extracted from model parameter estimates. Two of the three features can be physically interpreted as directionality and coarseness indicators. Rotation-invariant texture classification is performed based on the extracted texture features. Applications of 1D AR models to texture analysis have also been demonstrated by several authors [30,31,44].

It should be pointed out that in the above discussion, we have tacitly assumed that the texture field under consideration is stationary and ergodic. Under more realistic circumstances, one may have to employ non-stationary models [1,11].

2.4.2 GMRFM: Gaussian Markov Random Field Model

In addition to AR models, Markov random field models are another class of commonly employed models in texture analysis and other related disciplines. The study and applications of these models have a long history. A texture field $f(i,j)$, defined on an MxN lattice, is said to be a Markov random field or to have a Markovian distribution w.r.t. some neighborhood system η, *iff*

1. **Positivity**: $P\{f(i,j)\} \geq 0$, for all $(i,j) \in MxN$;
2. **Markovianity**: $P\{f(i,j) \mid \text{given all pixels } (k,l) \in MxN, (k,l) \neq (i,j)\} = P\{f(i,j) \mid (k,l) \in \eta\}$;
3. **Homogeneity**: Condition (2) depends only on the shape of η but not on the absolute position of η.

where for a given position (i,j), $f(i,j)$ is regarded as a random variable (RV) and $P\{x\}$ is the probability of event x. For simplicity, $f(i,j)$ is usually assumed to be Gaussian distributed, and

the underlying random field model is named as a Gaussian Markov Random Field Model or GMRFM for short. The explicit mathematical expression for the dependence of pixel (i,j) on its neighboring pixels (k,l) in η depends on the characteristic of $f(i,j)$ and on the structure of the neighborhood system η. For mathematical details on GMRFM, we refer to [29,53].

Chellappa and **Chatterjee** [2] used a class of GMRFMs for texture classification. They assumed a linear dependence of pixel (i,j) on its neighboring pixels $(k,l)\in\eta$, and subsequently arrived at the following mathematical expression for the underlying GMRFM:

$$Y(s) = \sum_{(k,l)\in\eta} \theta_{k,l}[Y(s+(k,l))+Y(s-(k,l))]+\sqrt{\beta}\ e(s)$$

Where $s = (i,j)\in MxN$, $Y(s)$ is the 2D texture field to be modeled, (k,l) is the relative coordinate pair in η w.r.t. pixel (i,j), $e(s)$ is a Gaussian white noise field, and β is the variance of $e(s)$. Note, in the above model, symmetric dependence has been assumed, i.e., pixels $(s+(k,l))$ and $(s-(k,l))$ are assumed to have an identical influence on pixel (i,j). They then estimated the model parameters $\theta_{k,l}$ and β from the input image data. Texture features were extracted from these parameter estimates for classification. A comparison was made between the classification performances by direct use of parameter estimates (i.e., consider them as texture features) and by indirect use of them (i.e., use features derived from them in some way). In both cases, a *minimum distance classifier* was used. **Chellappa** *et al.* [10] followed a similar procedure to investigate the applicability of GMRFMs to texture synthesis and texture data compression.

2.4.3 GRFM: Gibbs Random Field Model

A somehow more general random field model than GMRFM is the Gibbs random field model (GRFM). GRFM was first introduced in statistical mechanics to model molecular interactions in ferromagnetic materials. The model noncausality problem encountered in ARMs and GMRFMs does not exist in GRFMs. With GRFMs, one is able to develop computationally tractable, causal and robust algorithms. In a similar way to ARMs and GMRFMs, GRFMs also attempt to describe the dependence of a pixel on its neighbors. To define a GRFM, some preliminary definitions have to be first given. Let $X(i,j)$ be a 2D random field defined on an MxN lattice L indexed by i and j, and $x(i,j)$ a particular realization of $X(i,j)$. For a particular index pair (i,j), $X(i,j)$ is a random variable whereas $x(i,j)$ is a sample of the random variable,

i.e., a number. A subset of L is called a valid neighborhood system η_{ij} w.r.t. (i,j), if

1. $(i,j) \notin \eta_{ij}$; and
2. if $(k,l) \in \eta_{ij}$, then $(i,j) \in \eta_{kl}$.

For a pair (L,η), we say a set of pixels define a *clique* c if

1. c contains a single pixel in η, or
2. if $\{(i,j) \neq (k,l), (k,l) \in c, (i,j) \in c\}$, then $\{ (i,j) \in \eta_{kl}$ and $(k,l) \in \eta_{ij}\}$.

The set of all cliques defined on a pair (L,η) is denoted by a capital C.

Now, we are at the position to define a GRFM. A random field $X(i,j)$ is said to have a Gibbs distribution *iff* the joint distribution probability function w.r.t. η can be specified as follows:

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

Where

- $U(x) = \sum_{c \in C} V_c(x)$ is called an *energy function*;
- $V_c(x)$ is the *potential function* associated with clique c;
- $Z = \sum_x e^{-U(x)}$ is a normalizing factor in order to make $\int P(X=x)dx = 1$.

Note, we have dropped the (i,j) index in $X(i,j)$ and $x(i,j)$ for simplicity. The only constraint imposed on the potential function $V_c(x)$ is that it should depend only on the pixel values in c but not on the shape and the absolute position of c. As its expression tells us, a GRFM basically characterizes an exponential distribution. But surprisingly, by appropriately choosing $V_c(x)$, a wide variety of distributions can be formulated [22].

In applying GRFMs, the important steps are the selection of a neighborhood system η and the potential functions $V_c(x)$. The applicability of GRFMs to texture analysis has been illustrated by **Derin** and **Elliott** [22], and **Derin** and **Cole** [4]. In both papers, a second order neighborhood system η (a 3x3 neighborhood window) is chosen (For the definition of the order of a neighborhood system, see [22]). Then each valid clique c on η is associated with a parameter by defining the potential function $V_c(x)$ as follows:

$$V_c(x) = \begin{cases} -\xi, \text{ if } \textit{all pixels in c are equal}; \\ \\ \xi, \textit{otherwise}; \end{cases}$$

For single-pixel cliques, $V_c(x)$ is defined as

$$V_c(x) = \alpha_k, \text{ if } x(i,j) - k^{th} \text{ gray level}$$

More details on GRFMs may be found in [4] and [22].

2.4.4 MRFM: Mosaic Random Field Models

As we mentioned earlier, a texture model should take into account both the partition of an image into cells (cell distribution) and the assignment of gray levels to these cells (gray level distribution). A family of stochastic models called Mosaic Random Field Models (MRFM) are of this type. Depending on the geometry of the component cells, MRFMs may be further classified into two categories namely *Planar Tessellation Models* and *Geometrical Coverage Models*.

A. *Planar Tessellation Models*

A general planar tessellation mosaic model can be constructed in two steps: first, the planar region to be modeled is tessellated into cells or mosaics, and then one of the pre-specified L gray levels or colours is independently assigned to each of these cells (i.e., all pixels in the cell have that colour) according to a set of pre-defined assignment probabilities

$$p_1, p_2, \cdots, p_i, \cdots, p_L;$$
$$\sum_i p_i = 1$$

With different planar tessellations, one can obtain various useful mosaic models ranging from deterministic to highly random such as the Chessboard Model, Hexagonal Model, Triangular Model, Poisson Line Model, Occupancy Model and Delaunay Model [79].

B. *Geometrical Coverage Models*

While in planar tessellation models, all mosaics are constrained to have fixed geometrical shapes and to be non-overlapped, geometrical coverage models allow the inclusion of a set of distinctly shaped and possibly overlapped mosaics or cells. A geometrical coverage model is

generated by a random arrangement of a set of figures or "bombs" in a bounded plane. Two probability distributions may be sufficient to determine a geometrical coverage model as described below. A figure is first randomly chosen from a pre-assumed finite set of figures $\{f_i, i=1,2,\ldots,I\}$ according to a fixed set of geometrical shape distribution probabilities

$$p_{f1}, p_{f2}, \cdots, p_{fi}, \cdots, p_{fI};$$
$$\sum_i p_{fi} = 1$$

Then each figure or cell is independently assigned to one of the pre-specified L gray levels or colours based on a set of pre-defined cell colour distribution probabilities

$$p_1, p_2, \ldots, p_l, \ldots, p_L;$$
$$\sum_l p_l = 1$$

Some examples of figures that can be used are line segments, ellipses, circles, rectangles, hexagons and squares etc..

After a mosaic model is assumed, geometrical image properties such as the expected area and width of a component (cell), the expected number of components and the expected total component perimeter, and statistical image properties such as joint gray level probabilities, autocorrelation functions, edge densities can be calculated [41]. Features derived from these properties can then be used for texture analysis. Measures of some perceived texture features such as coarseness, busyness, roughness and patchiness can also be calculated based on mosaic random field models [24].

One shortcoming of MRFMs is the tacit assumption that images are composed of regions of pixels having *constant* intensities whereas realistic images usually contain regions of pixels having *nearly constant* or *similar* intensities. More details on MRFMs may be found in [41,60,79].

2.4.5 RWM: Random Walk Models

A random walk model aims to describe the gray level variations or distributions over an image region by means of some random walks. A random walk in a plane is characterized by a particle moving in unit steps along one of the pre-defined directions. For example, at pixel (i,j)

in a texture field $f(i,j)$, if one defines four moving directions $0°$ $[(i,j)\rightarrow(i,j+1)]$, $90°$ $[(i,j)\rightarrow(i,j,i-1,j)]$, $180°$ $[(i,j)\rightarrow(i,j-1)]$ and $270°$ $[(i,j)\rightarrow(i+1,j)]$, then a random walk w.r.t. (i,j) is described by a particle moving right, up, left and down at unit steps. A random walk is fully specified if the probabilities of leaving a given pixel (i,j) and of moving to one of its neighborhood pixels (k,l) are defined. Pixels in the region in which random walks are performed may be distinguished into two sets: boundary pixel set B and mesh pixel set M. Boundary pixels are those which have at least one neighbor outside the region concerned. Pixels in the region after subtraction of boundary pixels are mesh pixels. Most of RWM-based texture algorithms are then based on the probability distributions of leaving from and moving to M and B. For detailed mathematical treatments on RWMs, one can consult **Wechsler** and **Kidode** [46].

Wechsler and **Citron** [16] discussed a procedure for texture feature extraction and classification based on RWMs. They first divided a given image into suitably sized blocks such as 16x16 and 32x32. Then random walks are performed on these subimages, and the probability distributions of mesh pixels and boundary pixels are computed. Texture features are defined from these distributions. A total average classification accuracy of 89.7% is reported.

For texture segmentation, useful algorithms can also be developed based on random walk models as envisaged in [57].

2.4.6 FSM: Fractal Surface Model

Our universe is highly complex. It contains numerous shapes and forms which can not be described by conventional mathematical models. This representation incapability of standard mathematical models has impelled, in the past two decades, **Benoit Mandelbrot** to develop a new class of models called **Fractal Geometry** or **Fractal Surface Models** [64,65]. A very wide spectrum of physical processes produce surfaces which can be well represented by fractal models. These range from the aggregation of galaxies to the curdling of cheese. Technically, a fractal is defined as a set for which the Hausdorff-Besicovich dimension is strictly larger than the topological dimension, i.e., a set for which the only consistent description of its metric properties requires a *dimension* value larger than our standard, intuitive definition of the set's dimension. This consistent dimension is named as the **Fractal Dimension (FD)**, often denoted by letter D. FD D is indicative of the perceived roughness of a surface. For a flat surface, D is

2 whereas D is near 3 for a highly complex surface. The application of the fractal surface model in image and texture analysis was initiated by **Pentland** [61] in which some justification is presented leading to the assumption that an image intensity surface is a two-dimensional fractal Brownian. For a 2-D fractal Brownian surface, the fractal dimension D can be evaluated directly by using

$$E[\Delta I(\Delta x)] \bullet ||\Delta x||^{D-2} = E[\Delta I(\Delta x = 1)]$$

where $E[\Delta I(\Delta x)]$ is the expected value of the change in the image intensity $I(x)$ over a distance Δx, or by using

$$P(f) = kf^{2D-5}$$

where $P(f)$ is the power spectrum of $I(x)$ and k is a proportional constant. The most important characteristic of FD is its invariance (of being stable) to a wide range of scales.

For texture analysis, FD is often used as a texture feature. For instance, in order to achieve texture discrimination and segmentation, **Pentland** [61] first divides the input image into windows of predefined size (say 8x8), then the FD Ds calculated from these windows are histogramized. By thresholding the D histogram at appropriate valleys, he obtained a reasonable segmentation of the input image. As mentioned earlier, the fractal dimension D is indicative of only one visual texture feature, the roughness. Thus, two texture surfaces can be contrived which possess identical FDs and yet visually are highly separable. This means FD alone may not be an adequate texture descriptor as indeed noted in [66]. To improve the applicability of fractal surface models in texture analysis, **Ait-Kheddache** and **Rajala** [67] subsequently suggested the use of higher-order fractal dimensions (H-OF). The H-OF dimensions are composed of the usual fractal dimension, the correlation dimension and the information dimension.

2.4.7 MSFM: Multichannel Spatial Filtering Model

In model-based approaches, the methods reviewed thus far are all based on the modeling of given image data in order to facilitate texture description and feature extraction. In this sub-section, we shall review texture analysis approaches based on the modeling of the HVS in processing given image data. The underlying model, called the *Multichannel Spatial Filtering Model*, is rooted on the hypothesis that the processing of afferent pictorial information in the

HVS (the visual cortex in particular) involves a set of parallel and quasi-independent mechanisms or *channels* [68], each of which may be envisaged to perform a two-dimensional spatial filtering operation.

Since the establishment of the MSFM in the vision research community, there has been a great deal of interest in applying the model to texture analysis. The earliest application is probably due to **Faugeras** [70]. The work is based on the structural model of a single channel proposed in [71]. In the i^{th} channel, the input image $P(x,y)$ is first spatially filtered by a 2-D linear filter with transfer function $H_i(f_1,f_2)$, giving a filtered image $S_i(x,y)$. This image, after being weighted by $a_i(x,y)$, undergoes a non-linear operation (the p^{th} power of the absolute value). Finally, the output from the non-linear operator passes through a spatial adder to give a scalar channel output t_i. This single channel model assumes the transfer function $H_i(f_1,f_2)$ can be represented as the product of the radial frequency response $H_{ir}(f)$ and the spatial orientation response $H_{ia}(\theta)$ in polar coordinates. The description of the filter is given by

$$H_i(f,\theta) = H_{ir}(f)H_{ia}(\theta)$$

$$H_{ir}(f) = [1+\frac{(f-f_{0i})^2}{w_i}]^{-\frac{1}{2}}$$

$$H_{ia}(\theta) = e^{-\frac{(\theta-\theta_{0i})^2}{2b}} + e^{-\frac{(\theta-\theta_{0i}-\pi)^2}{2b}}$$

where w_i and b define the radial and angular bandwidth, and (f_{0i},θ_{0i}) is the central frequency of the i^{th} channel. For texture analysis, **Faugeras** used the channel output t_i as a texture feature.

Similar work has been done recently by **Wright** and **Jernigan** [72,73], and **Vicente** and **Abella** [18]. For example, in [72], they assume that the visual response characteristics in the spatial frequency and orientation dimensions are independent of each other and subsequently define the polar frequency response $F(r,\theta)$ of a channel to be the product of the radial frequency response $F(r)$ and the angular response $F(\theta)$. Mathematically, the channel model is described by the following:

$$F_i(r,\theta) = F_i(r)F_i(\theta)$$

$$F_i(r) = A_i\sigma_i\pi^{1/4}[e^{-\pi\sigma_i r)^2} - D_i e^{-(\Delta_i\pi\sigma_i r)^2}]$$

$$F_i(\theta) = e^{-\frac{0.5\theta^{*2}}{\sigma_\theta^2}}$$

where $i = 1,2,3,4$ corresponding to the four mechanisms or visual channels N,T,S,U proposed by **Wilson** and **Bergen** respectively [74]. Filter parameters are those measured empirically in [74]. For texture feature extraction, **Wright** and **Jernigan** [72] filtered an input image independently with each individual radial frequency filter as well as each individual orientation filter. They then defined the logarithm of the energy output from each filter as a texture measure. A similar methodology and channel model has been followed by **Coggins** and **Jain** in texture segmentation as well as texture classification [75].

Another example of multichannel spatial filtering approaches to texture analysis is given by **Ikonomopoulos** and **Unser** [76]. Instead of taking filter models based on physiological and psychophysical studies, they define a set of so-called *directional filters* empirically. To define n directional filters, they partition the entire frequency plane into $2n$ wedged regions. They then define the frequency response $G_i(u,v)$ of the i^{th} filter as the ideal wedge-shaped response $G_{Ii}(u,v)$ weighted by a 2-D Gaussian function $W(u,v)$ as described by the following:

$$G_i(u,v) = G_{Ii}(u,v)W(u,v)$$

$$G_{Ii}(u,v) = \begin{cases} 1, & \text{if } \theta_i \le \tan^{-1}(u/v) < \theta_{i+1} \\ 0, & otherwise \end{cases}$$

$$\theta_i = \frac{i\pi}{n}$$

The MFSM based approaches reviewed so far all perform two-dimensional or spatial filtering in the frequency domain. The alternative of performing spatial filtering via spatial convolution is equally applicable. For example, with the belief in mind that feature extraction in the HVS involves a set of directional filters, **Wermser** and **Leidtke** [77] perform texture analysis via directional filtering in the spatial domain. Directional convolution masks are defined empirically. These masks have four directions (0^0, 45^0, 90^0 and 135^0) and three different window sizes: 5x5, 9x9 and 19x19.

For more on the MSFM in texture analysis, see Chapters 6 and 9 of this thesis.

2.5 Remarks

In this chapter, we have attempted to review the literature on the major techniques for texture feature extraction and classification. Because of space limitation, this review is necessarily incomplete in that some more references could have been mentioned. To remedy this incompleteness, we have described the basic principles of each major technique and examined the references which are thought to be representative of the underlying technique. Three major groups of techniques, namely Classical Statistical Approaches, Structural Approaches and Model-Based Approaches, are reviewed.

Statistical approaches are those which were extensively studied in the past. They are generally applicable as statistics are always extractable from any kind of texture. Nevertheless, if a given texture is composed of position-invariant primitives arranged according to well-defined placement rules, then statistical approaches are obviously not ideal choices as this texture can be more compactly described by a structural approach. Among all statistical approaches, the GLDM technique is the most popular and has been the benchmark against which a new technique should be compared. The major drawback of GLDM, as we mentioned, is its computational complexity and the choice of appropriate displacement vectors.

Structural approaches, on the other hand, are less applicable. They require textures to be composed of primitives which are easily identifiable and whose interactions are inferable in some cases. The application of structural approaches has been very limited as textures in reality rarely fit the requirements necessary for the implementation of a structural approach, and for stochastic textures, structural approaches would be completely useless (See Chapter 5). In addition, primitive and placement rule extraction is usually very complex and time-consuming. Some structural approaches, i.e., strong texture analysis approaches and syntactic texture analysis approaches, are capable of texture synthesis as well as texture analysis but the synthesized textures are in general monotonous [92].

Model-based approaches are based either on modeling given image data or on modeling the HVS in processing given image data. In the former, most of the models were originally developed in the texture synthesis field whereas in the latter, models are drawn directly upon the findings in spatial vision research. Methods in the former category are in general computationally expensive and their applications may be hampered by the difficulty to choose

appropriate neighborhood systems or to determine the order of models. After all, results of model-based techniques are impressive, and further investigations on the MSFM technique in particular seem to be worthwhile [6].

In the following four chapters, we shall present a number of novel algorithms for texture feature extraction and classification. These algorithms attempt either to unify, generalize and improve some of the existing algorithms or to open a completely new area for texture analysis. In developing these algorithms, we must comply with the criterion of easy implementation and high performance.

2.6 References

[1] **A.M.Tekalp** *et al.*, *Fast Recursive Estimation of the Parameters of a Space-Varying AR Model*, IEEE Trans. on Acoustic, Speech and Signal Processing, vol.ASSP-33, 4/1985, pp.469-472.

[2] **R.Chellappa** and **S.Chatterjee**, *Classification of Textures Using Gaussian Markov Random Fields*, IEEE Trans. on Acoustic, Speech and Signal Processing, vol. ASSP-33, 8/85, pp.959-963.

[3] **R.Chellappa** and **R.L.Kashyap**, *Texture Synthesis Using 2-D Noncausal AR Models*, IEEE Trans. on Acoustic, Speech and Signal Processing, vol.ASSP-33,2/85, pp.194-203.

[4] **H.Derin** and **W.S.Cole**, *Segmentation of Textured Images Using Gibbs Random Fields*, Comput. Vision, Graphics and Image Processing, vol.35, 7/1986, pp.72-98.

[5] **A.Gagalowicz** and **Song De Ma**, *Sequential Synthesis of Natural Textures*, Comput. Vision, Graphics and Image Proc., vol.30, 1985,pp.289-315.

[6] **L.V.Gool** *et al.*, *Survey: Texture Analysis Anno 1983*, Comput. Vision, Graphics and Image Proc., vol.29, 1985, pp.336-357.

[7] **D.Terzopoulos** and **S.W.Zucker**, *Detection of Osteogenesis Imperfecta by Automated Texture Analysis*, Comput. Graphics and Image Processing, vol.20, pp.229-243, 1982.

[8] **K.Deguchi** and **I.Morishita**, *Two-Dimensional Auto-Regressive Model for the Representation of Random Image Fields*, Proc. of 6th Inter. Conf. on Pattern Recognition, Munich, 1982.

[9] **Song De Ma** and **A.Gagalowicz**, *Natural Texture Synthesis with the Control of the Autocorrelation and Histogram Parameters*, Proc. of 3rd Scandinavian Conf. on Image Analysis, Copenhagen, Denmark, July 12-14, 1983, pp.79-84.

[10] **R.Chellappa** *et al.*, *Texture Synthesis and Compression Using Gaussian-Markov Random Field Models*, IEEE Trans. Sys. Man. Cyber., vol.SMC-15, 1985, pp.298-303.

[11] **B.R.Hunt**, *Non-Stationary Statistical Image Models (and Their Applications to Image Data Compression)*, Comput. Graphics and Image Proc., vol.12, pp.173-186, 1980.

[12] **J.O.Eklundh**, *On the Use of Fourier Phase Features for Texture Discrimination*, Comput. Graphics and Image Proc., vol.9, pp.199-201, 1979.

[13] **P.C.Chen** and **T.Pavlidis**, *Segmentation by Texture Using a Co-occurrence Matrix and a Split-and-Merge Algorithm*, Comput. Graphics and Image Proc., vol.10, pp172-182, 1979.

[14] **W.K.Pratt** *et al.*, *Visual Discrimination of Stochastic Texture Fields*, IEEE Trans. on Sys. Man and Cyber., vol.SMC-8, 1978, pp.796-804.

[15] **M.Unser**, *Local Linear Transforms for Texture Measurements*, Signal Processing, vol.11, pp.61-79, 7/1986.

[16] **H.Wechsler** and **T.Citron**, *Feature Extraction for Texture Classification*, Pattern Recognition, vol.12, pp.301-311, 1980.

[17] **L.S.Davis**, *Polarograms: A New Tool for Image Texture Analysis*, Pattern Recognition, vol.13, pp.129-223, 1981.

[18] **A.Vicente** and **G.Abella**, *A Visual Perception Approach to Texture Description*, Proc. of MELECON'85 Mediterranean Electrotechnical Conf., Madrid, Spain, 8-10 Oct.1985, pp.273-278.

[19] **F.Ade**, *Characterization of Texture by Eigenfilters*, Signal Processing, 5/1983, pp.451-457.

[20] **H. Wechsler**, *Texture Analysis: A Survey*, Signal Processing, vol.2, pp.271-282, 1980.

[21] **R.Wang** *et al.*, *Texture Analysis Based on Local Standard deviation of Intensity*, Proc. CVPR'86: IEEE Computer Society Conf. on Comput. Vision and Pattern Recognition, FL, USA, 22-26 June 1986, pp.482-488.

[22] **H.Derin** and **H.Elliott**, *Modelling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-9, pp.39-55, Jan. 1987.

[23] **L.S.Davis** *et al.*, *An Empirical Evaluation of Generalized Co-occurrence Matrices*, IEEE Trans. Pattern Anal. Mach. Intell., vol.PAMI-3, pp.214-221, 1981.

[24] **B.Schachter**, *Model-Based Texture Measures*, IEEE Trans. Pattern Anal. Mach. Intell., vol.PAMI-2, pp.169-171, 1980.

[25] **R.M.Haralick** *et al.*, *Texture Features For Image Classification* IEEE Trans. on Systems, Man and Cybern., vol.SMC-3, 1973.

[26] **S.Wang** *et al.*, *Relative Effectiveness of Selected Texture Primitive Statistics for Texture Discrimination*, IEEE Trans. Syst., Man. Cybern., vol.SMC-11, pp.360-370, May 1981.

[27] **J.S.Weszka** *et al.*, *A Comparative Study of Texture Measures for Terrain Classification* , IEEE Trans. Syst., Man, Cybern., vol.SMC-6, pp.269-285, Apr.1976.

[28] **R.M.Haralick**, *Statistical and Structural Approaches to Texture*, Proc. of IEEE, vol.67, pp.786-804, 1979.

[29] **A.K.Jain**, *Advances in Mathematical Models for Image Processing*, Proc. of IEEE, vol. 69, pp.502-528, 1981.

[30] **T.N.Tan**, *Dynamic Programming Approach to Texture Edge Identification Using Linear Estimation Technique*, MSc thesis, Electrical Engineering Department Library, Imperial College, 1986.

[31] **B.H.McCormick** *et al.*, *Time Series Model for Texture Synthesis*, Int. Jour. of Comput. Inf. Sci., vol.3, pp.329-343, 1974.

[32] **S.W.Zucker**, *Toward a Model of Texture*, Comput. Graphics and Image Processing, vol.5, pp.190-202, 1976.

[33] **S.Y.Lu** and **K.S.Fu**, *A Syntactic Approach to Texture Analysis*, Comput. Graphics Image Processing, vol. 7, pp.303-330, 1978.

[34] **M.Galloway**, *Texture Analysis Using Gray Level Run Lengths*, Comput. Graphics Image Processing, vol.4, pp.172-199, 1974.

[35] **L.Davis** *et al.*, *Texture Analysis Using Generalized Co-occurrence Matrices*, presented at Pattern Recog. and Image Proc.Conf. (Chicago, IL), May 31-June 2, 1978.

[36] **R.L.Kashyap** and **A.Khotanzad**, *A Model-Based Method for Rotation Invariant Texture Classification*, IEEE Trans. Pattern Anal. Machine Intell., vol.PAMI-8, pp.472-482, Jul. 1986.

[37] **R.W.Conners** *et al.*, *A Theoretical Comparison of Texture Algorithms*, IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-2, pp.204-222, 1980.

[38] **R.Sutton** and **E.Hall**, *Texture Measures for Automatic Classification of Pulmonary Disease*, IEEE Trans. Comput., vol.C-21, pp.667-676, 1972.

[39] **G.E.P.Box** and **G.M.Jenkins**, *Time Series Analysis: forecasting and control*, revised edition, San Francisco: Holden-Day, 1976.

[40] **K.Fukunaga**, *Introduction to Statistical Pattern Recognition*, New York: Academic Press, 1972.

[41] **N.Ahuja** and **A.Rosenfeld**, *Mosaic Models for Textures*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-3, no.1,pp.1-11, 1981.

[42] **C.R.Dyer** *et al.*, *Texture Classification Using Gray Level Co-occurrence Based on Edge Maxima*, IEEE Trans. Sys.Man. Cyber., vol. SMC-10, pp.158-163, 1980.

[43] **T.H.Hong** *et al.*, *Texture Primitive Extraction Using an Edge-Based Approach*, IEEE Trans. Sys. Man. Cyber., vol.SMC-10, pp.659-675, 1980.

[44] **P.De.Souza**, *Texture Recognition via Autoregression*, Pattern Recognition, vol.15, 1982, pp.471-475.

[45] **A.Rosenfeld** and **M.Thurston**, *Edge and Curve Detection for Visual Scene Analysis*, IEEE Trans. Comput., vol.C-20, pp.562-569, 1971.

[46] **A.Wechsler** and **M.Kidode**, *A Random Walk Procedure for Texture Discrimination*, IEEE Trans. Pattern Anal. Mach.Intell., vol.PAMI-1, pp.272-280, 1979.

[47] **M.Nagao** *et al.*, *Agricultural Land Use Classification of Aerial Photographs by Histogram Similarity Method*, Proc. of 3rd Int. Conf. on Pattern Recognition, Coronado, Calif. Nov.8-11, 1976, pp.669-672.

[48] **K.I.Laws**, *Textured Image Segmentation*, PhD Dissertation, University of Southern California, California, 1980.

[49] **M.Pietikainer** *et al.*, *Texture Classification Using Averages of Local Pattern Matches*, in Proc. 6th Int. Conf. on Pattern Recognition, Munich, 1982, pp.301-303.

[50] **S.Tsuji** and **F.Tomita**, *A Structural Analyzer for a Class of Textures*, Comput. Graphics, Image Processing, vol. 2, 1973, pp.216-231.

[51] **R.W.Conners** and **C.A.Harlow**, *Towards a Structural Texture Analyzer Based on Statistical Method*, Comput. Graphics, Image Processing, vol.12, 1980, pp.224-256.

[52] **K.S.Fu**, *Syntactic Pattern Recognition and Applications*, Prentice-Hall, Endlewood Cliffs, N.J., 1982.

[53] **G.R.Cross** and **A.K.Jain**, *Markov Random Field Texture Models*, Proc. of Inter. Conf. on Pattern Recognition and Image Processing, Dallas, Texas, USA, Aug.3-5, 1981, pp.597-602.

[54] **D.Barba** and **J.Ronsin**, *New Method in Texture Analysis in the Context of Image Segmentation*, Sig. Proc. II: Theories and Appl., Proc. of EUSIPCO"83, 2nd European Sig. Proc. Conf., pp.283-6, Erlanger, Germany, 12-16/9/1983.

[55] **T.Matsuyama** *et al.*, *A Structural Description of Regularly Arranged Textures*, Proc. of 5th Int. Conf. on Pattern Recognition, Miami Beach, Florida, Dec.1-4, 1980, pp.1115-1118.

[56] **G.H.Granlund**, *Description of Texture Using the General Operator Approach*, *ibid*, pp.776-779.

[57] **H.Wechsler**, *Taxonomy and Segmentation of Textured Images*, *ibid*, pp.532-534.

[58] **T.Matsuyama** *et al.*, *A Structural Analysis of Natural Textures by Fourier Transform*, Proc. of 6th Int. Conf. on PR, Munich, 1982, pp.289-292.

[59] **P.T.Nguyen** and **J.Quinqueton**, *Space-Filling Curves and Texture Analysis*, *ibid*, pp.282-285.

[60] **B.J.Schachter** *et al.*, *Random Mosaic Models for Textures*, IEEE Trans. on Syst. Man Cybern, vol.SMC-8, pp.694-702, 1978.

[61] **A.P.Pentland**, *Fractal-Based Description of Natural Scenes*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-6, pp.661-674, 1984.

[62] **H.C.Kennedy**, *Selected Works of Giuseppe Peano*, London: George Allen & Unwin Ltd, pp.143-149, 1974.

[63] **Dong-Chen He** *et al.*, *Texture Feature Extraction*, Pattern Recognition Letter, vol.6, 1987, pp.269-273.

[64] **B. B. Mandelbrot**, *Fractals: Form, Chance, and Dimension*, San Francisco, CA: Freeman 1977.

[65] **B. B. Mandelbrot**, *The Fractal Geometry of Nature*, San Francisco, CA: Freeman 1983.

[66] **G. G. Medioni** and **Y. Yasumoto**, *A Note on Using the Fractal Dimension for Segmentation*, IEEE Computer Vision Workshop, Annapolis MD, 1984, pp.25-30.

[67] **A. Ait-Kheddache** and **S. A. Rajala**, *Texture Classification Based on Higher-Order Fractals*, Proc of the Inter. Conf. on Acoustics, Speech and Signal Processing, New York, USA, 1988, pp.1112-1115.

[68] **K.K.DeValois**, *Spatial Vision*, Annual Review of Physiology, 1984.

[69] **B.Julesz** *et al*, *Textons, the Fundamental Elements in Preattentive Vision and Perception of Textures*, The Bell Syst Tech J, vol.62, 1983, pp.1619-1645.

[70] **O.D.Faugeras**, *Texture Analysis and Classification Using A Human Vision Model*, Proc of the Inter. Conf. on Pattern Recognition, 1978, pp.549-552.

[71] **H.Mostafavi** and **D.S.Sakrison**, *Structure and Properties of A Single Channel in The Human Visual System*, Vision Research, vol.16, 1976, pp.957-968.

[72] **G.A.Wright** and **M.E.Jernigan**, *Spatial Frequency and Orientation Correlates of Textures in Human Vision*, Proc. of Inter. Conf. on Cybern. and Soc., Tucson, Arizona, Nov.12-15, 1985, pp.697-700.

[73] **G.A.Wright** and **M.E.Jernigan**, *Texture Discrimination from Spatial Frequency Channels*, Proc. of Inter. Conf. on Syst. Man and Cybern., Atlanta, GA, USA, Oct.14-17, 1986, pp.519-524.

[74] **H.R.Wilson** and **J.R.Bergen**, *A Four Mechanism Model for Threshold Spatial Vision*, Vision Research, vol.19, 1979, pp.19-32.

[75] **J.M.Coggins** and **A.K.Jain**, *A Spatial Filtering Approach to Texture Analysis*, Pattern Recognition Letter, vol.3, 1985, pp.195-203.

[76] **A.Ikonomopoulos** and **M.Unser**, *A Directional Filtering Approach to Texture Discrimination*, Proc. of Inter. Conf. on Pattern Recognition, Montreal, Canada, July 30-Aug. 2 1984, pp.80-82.

[77] **D.Wermser** and **C.E.Leidtke**, *Texture Analysis Using A Model of Visual System*, Proc. of Inter. Conf. on Pattern Recognition, 1982, pp.1078-1080.

[78] **D.Marr**, *Vision*, W.H.Freeman, 1982.

[79] **N.Ahuja** and **B.J.Schachter**, *Pattern Models*, New York: John Wiley & Sons, 1983.

[80] **R.O.Duda** and **P.E.Hart**, *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[81] **O.D.Faugeras** and **W.K.Pratt**, *Decorrelation Methods of Texture Feature Extraction*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-2, 1980, pp.323-332.

[82] **F.Dastous** and **M.E.Jernigan**, *Texture Discrimination Based on Detailed Measures of the Power Spectrum*, Proc. of the 7th Inter. Conf. on Pattern Recognition, Montreal, Canada, July 30-Aug.2, 1984, pp.83-86.

[83] **C.H.Chen**, *A Study of Texture Classification Using Spectral Features*, Proc. of 6th Inter. Conf. on PR, Munich, 1982, pp.1064-1067.

[84] **A.E.J.Walford** and **M.E.Jernigan**, *The Phase Spectrum and Its Contribution to Human Texture Perception*, Proc. of Inter. Conf. on Cybern. and Soc., Tucson, Arizona, Nov.12-15, 1985, pp.701-704.

[85] **F.Dastous** and **M.E.Jernigan**, *Phase Information in Texture Feature Extraction*, Proc. of Inter. Conf. on Cybern. and Soc., Atlanta, Georgia, Oct.26-28, 1981, pp.182-186.

[86] **R.Bajcsy**, *Computer Identification of Textured Visual Scenes*, PhD dissertation, Stanford University, Stanford, California.

[87] **F.M.Vilnrotter** *et al.*, *Structural Analysis of Natural Textures*, IEEE Trans. on Pattern Anal. Mach. Intell., vol. PAMI-8, 1986, pp.76-90.

[88] **A.Rosenfeld** and **E.Troy**, *Visual Texture Analysis*, TR-116, Computer Science Center, University of Maryland, June 1970.

[89] **O.Mitchell** *et al.*, *A Max-Min Measure for Image Texture Analysis*, IEEE Trans. on Comput., vol.C-26, 1977, pp.408-414.

[90] **S.W.Zucker** and **D.Terzopoulos**, *Finding Structure in Co-occurrence Matrices for Texture Analysis*, Comput. Graphics and Image Proc., vol.12, pp.286-308, 1980.

[91] **F.Tomita** *et al.*, *Description of Textures by a Structural Analyzer*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-4, 1982, pp.183-191.

[92] **P.Volet** and **M.Kunt**, *A New Method for the Synthesis and Efficient Coding of Natural Structural Textures*, Proc. of SPIE, vol.594, 1985, pp.225-232.

3. TEXTURE DESCRIPTION BY LOCAL VECTOR MAPPING

3.1 Introduction

As texture is a local regional image characteristic, some types of local operations are intuitively useful for texture feature extraction and description. Our intuition has found solid support in the *texton theory* postulated by **Julesz** [1]. This theory asserts that preattentive texture perception and discrimination is based on some local conspicuous image properties or features which are best computed by means of local operations [2]. In the past few years, a large group of approaches towards texture analysis have been suggested and developed based on local operations [3-13,18]. Some of these approaches have been briefly described in the preceding chapter. Most of these approaches are based on linear local operations even though the usefulness of some non-linear local operations such as local extrema and standard deviation computation for texture description is obvious.

In this chapter, the introduction of a conceptually new scheme, called the **local vector mapping (LVM)** scheme, enables us to offer a consistent and convenient interpretation, a unification and a generalization of these local-operation-based approaches. In the following sections, we shall describe this new scheme and its applications in texture feature extraction and classification.

3.2 Local Vector Mapping

A local vector mapping operation or simply a LVM involves the definition of a local neighborhood, the formation of a vector, and the mapping of the vector. A local neighborhood is an arbitrarily-shaped, connected local region and may contain any number of pixels. Vector formation is the process of translating a 2-D region of pixels into a 1-D sequence or vector of pixels. And finally, a vector mapping is a linear or non-linear vector transformation.

3.2.1 Local Neighborhood

Let i and j be the index variables for a grid lattice IxJ with $i \in (1,2,3,\ldots,I)$ and $j \in (1,2,3,\ldots J)$, and let a digital image be denoted by $f(i,j)$ with $(i,j) \in IxJ$ and $f(i,j) \in (1,2,3,\ldots,Q)$. A sub-region of the field $f(i,j)$ or a sub-set of pixels in $f(i,j)$ will be symbolized by η, i.e.,

$$\eta \in \{(i,j): 1 \le i \le I,\ 1 \le j \le J\}$$

Formally, we call a sub-set η a *valid local neighborhood* $\eta_B(i,j)$ associated with pixel (i,j) *iff*

1. $(i,j) \in \eta_B(i,j)$; and
2. *B is a simple closed curve.*

where B is the boundary of the local image region concerned. Condition (1) states that pixel (i,j) is a member of its own local neighborhood, while Condition (2) ensures that a valid local neighborhood contains a set of neighboring pixels, and is a connected region in that no "holes" are present within the region. In passing, it is worthwhile to mention the difference between the definition of a valid local neighborhood given here and that used in some texture models (see, for instance, in Gibbs random field models described in Section 2.4.3), where a pixel is precluded from its own local neighborhood. If we define the centroid (m,n) of a local neighborhood $\eta_B(i,j)$ as

$$\begin{aligned} m &= Int\Big[\frac{\sum_{(k,l)} k}{K}\Big]; \quad (k,l) \in \eta_B(i,j) \\ n &= Int\Big[\frac{\sum_{(k,l)} l}{K}\Big]; \quad (k,l) \in \eta_B(i,j) \end{aligned} \tag{3.1}$$

where $Int[\bullet]$ denotes taking the nearest integer and K is the total number of pixels in $\eta_B(i,j)$, i.e.,

$$K = \#\{(k,l): (k,l) \in \eta_B(i,j)\}$$

then a local neighborhood $\eta_B(i,j)$ is called a *symmetrical local neighborhood* w.r.t. the underlying pixel (i,j) if $\eta_B(i,j)$ satisfies:

$$\sum_{(k,l)} (k-m) = 0 \text{ and } \sum_{(k,l)} (l-n) = 0; \ (k,l) \in \eta_B(i,j) \tag{3.2}$$

If K is an odd number, then the centroid (m,n), the center and the pixel (i,j) of a symmetrical $\eta_B(i,j)$ will be identical. Although $\eta_B(i,j)$ can be, in principle, arbitrarily shaped and can contain any number of pixels, it is presumed in the subsequent discussions that $\eta_B(i,j)$ is symmetrical and contains a total of K pixels with K being odd unless otherwise stated.

Fig.3.1 shows some examples of valid local neighborhoods. With reference to texture feature extraction, care should be taken when choosing an appropriate local neighborhood

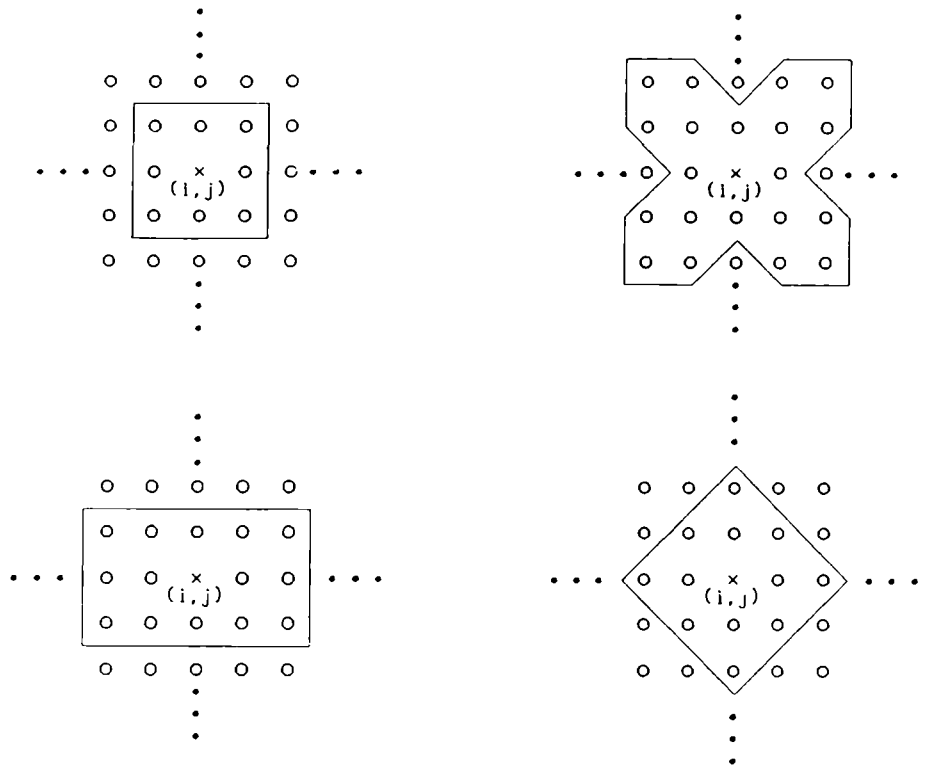


Figure 3.1. Examples of Valid Local Neighborhoods.

$\eta_B(i,j)$. As we demonstrated in Fig.1.3, the perceivable textural information conveyed in a $\eta_B(i,j)$ of some size may be independent of the absolute position (i,j) of $\eta_B(i,j)$. In particular, if the texture $f(i,j)$ under consideration is purely deterministic or strictly regular, then an adequate description of a $\eta_B(i,j)$ of some size will suffice for the characterization of the entire texture field $f(i,j)$. For natural textures, if the identifiable primitives are of relatively large size, then large $\eta_B(i,j)$s should be used. On the other hand, if the given texture image is stochastically uniform and homogeneous, and the pixels are weakly correlated, then relatively small $\eta_B(i,j)$s can be defined. For instance, for a uniform and homogeneous (also stationary) Gaussian random field of independently and identically distributed random variables, definition and characterization of a local neighborhood $\eta_B(i,j)$ with a single pixel (i,j) will be adequate.

3.2.2 Local Vector Formation

Local neighborhood specification associates each pixel (i,j) with a valid local neighborhood $\eta_B(i,j)$. The $K-1$ pixels in $\eta_B(i,j)$, excluding (i,j), will be called the $K-1$ *neighbors* of pixel (i,j).

If the K pixels in $\eta_B(i,j)$ are ordered sequentially, one eventually obtains a 1-D pixel sequence or a vector with K elements. Such a vector is termed as a *local vector* at pixel (i,j) w.r.t the neighborhood system $\eta_B(i,j)$, and the corresponding process of translating the pixels in $\eta_B(i,j)$ into such a vector is referred to as *local vector formation*. This local vector will be attached to pixel (i,j), and denoted by

$$\vec{V}(i,j) = [v_1(i,j)\ v_2(i,j)\ \cdots\ v_k(i,j)\ \cdots\ v_K(i,j)]^T$$

with $v_k(i,j)$ being the k^{th} element of $\vec{V}(i,j)$. To formally define the local vector formation process, let us represent the process symbolically by

$$\Psi:\ \eta_B(i,j) \rightarrow \vec{V}(i,j)$$

or simply write

$$\Psi[\eta_B(i,j)] = \vec{V}(i,j)$$

We call a translation or mapping Ψ from a 2-D local neighborhood $\eta_B(i,j)$ to a 1-D local vector $\vec{V}(i,j)$ a *valid local vector formation procedure* if Ψ satisfies the following:

1. **Uniqueness:** If $\Psi_1[\eta_B(i,j)] = \vec{V}(i,j)$ and $\Psi_2[\eta_B(i,j)] = \vec{V}(i,j)$, then $\Psi_1 = \Psi_2$;
2. **Completeness:** If there is a pixel A in $\eta_B(i,j)$, there is a corresponding element in $\vec{V}(i,j)$ or vice versa;
3. **Simplex:** If the pixels in $\eta_B(i,j)$ are linked with line segments in the same order as they are scanned in the process of local vector formation, then the resultant continuous curve is a simple one in the sense that it has no crossings.

Some thoughts will reveal that after the formation process Ψ is implemented at every pixel in $f(i,j)$, the original scalar image $f(i,j)$ is transformed into a *vector image* $\vec{F}(i,j)$ with the pixel at (i,j) being a local vector $\vec{V}(i,j)$, or into a multivariate image sequence $\{f_k(i,j):\ k=1,2,\ldots,K \mid f_k(i,j) = v_k(i,j)\}$. At this point, one may already realize that the methods of forming a 1-D local vector from a 2-D local neighborhood $\eta_B(i,j)$ are numerous. This is indeed the case and may be regarded as one of the advantages of the LVM scheme. Nevertheless, we maintain that a "good and desirable" local vector formation procedure Ψ should be such that

the image topology and the spatial pixel correlations in $\eta_B(i,j)$ be preserved and maintained in the resultant vector $\vec{V}(i,j)$ as much as possible. Fig.3.2 illustrates some examples of valid local vector formation schemes.

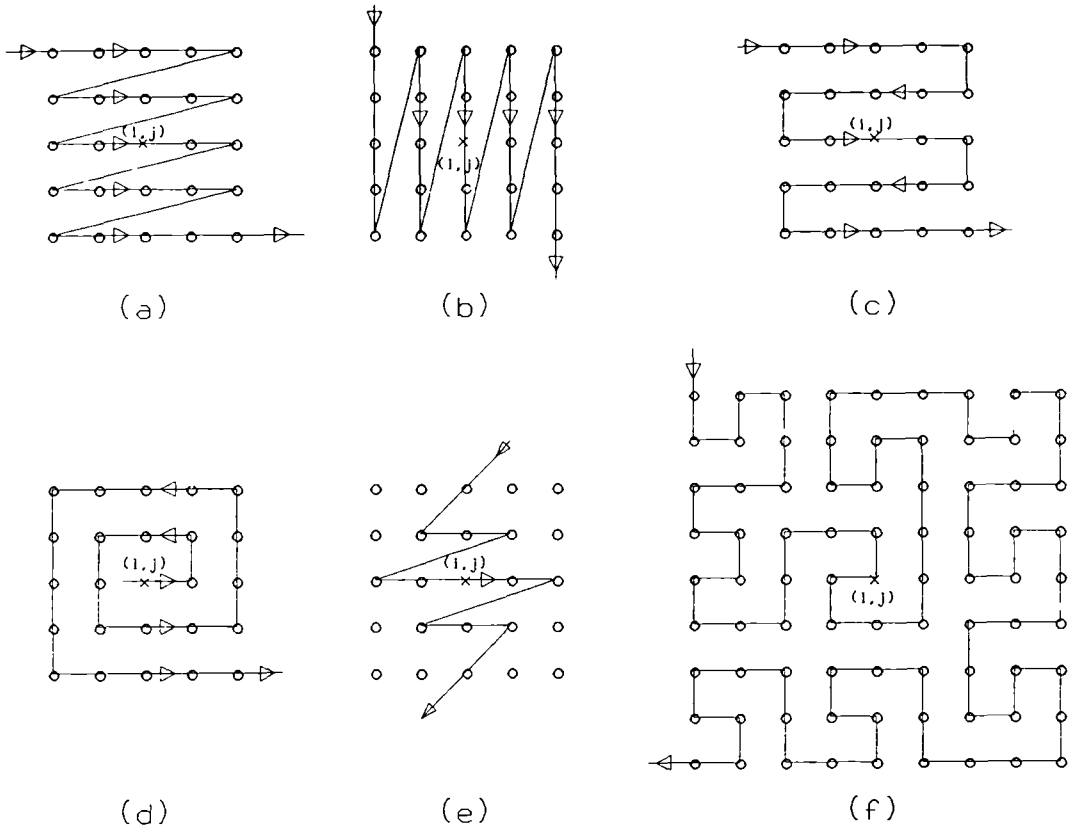


Figure 3.2. Examples of Valid Local Vector Formation Procedures. (a) Row concatenation; (b) Column concatenation; (c)-(e) Some variants of (a) and (b) and (f) Peano scanning.

Fig.3.2(a) and (b) are of the conventional raster scan type and of either row concatenation or column concatenation of the pixels in $\eta_B(i,j)$. Fig.3.2(c) to (e) are some variants of Fig.3.2(a) and (b). Evidence has shown that usual scanning with parallel lines like those in Fig.3.2(a) to (e) does not adequately preserve the topology of the underlying image, and the pixels within a small neighborhood of a given image may be mapped into many disconsecutive points in the formed 1-D local vector [14]. This undesirable weakness can be alleviated to some extent by a special scanning scheme called the *space-filling curve* or *Peano scanning* [15]. Fig.3.2(f) shows the Peano scanning scheme for the purpose of local vector formation in a local neighborhood of size 9x9. The relevant properties of Peano scanning to the formation of local vectors are

summarized in [19]. For more details on Peano scanning and its applications in image processing, we refer to [12, 14-17].

The differences between distinct local vector formation schemes are also exemplified in Fig.3.3.

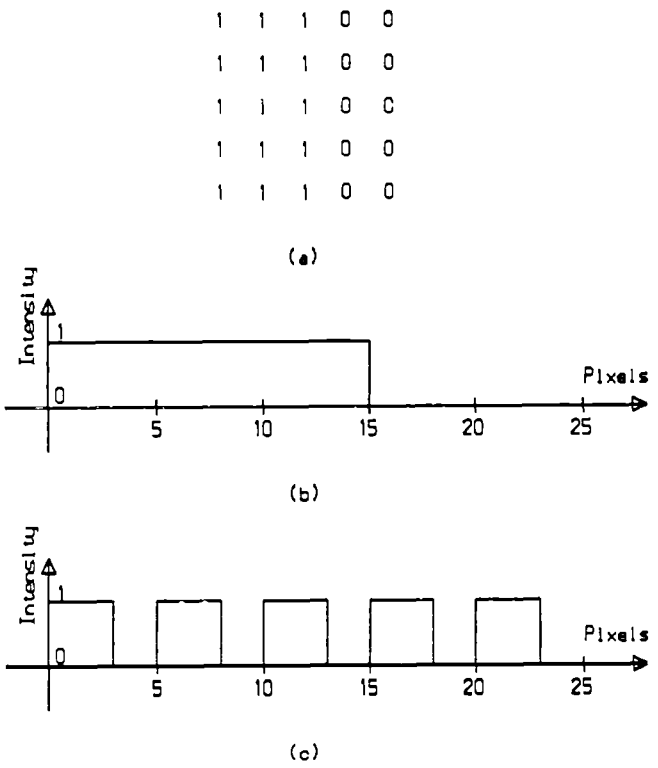


Figure 3.3. Illustration of the Differences between Distinct Local Vector Formation Schemes. (a) A 5x5 local neighborhood; (b) The local vector obtained via column concatenation and (c) The local vector obtained via row concatenation.

Fig.3.3(a) shows a local neighborhood $\eta_B(i,j)$ of size 5x5, Fig.3.3(b) gives the local vector $\vec{V}(i,j)$ formed by means of column concatenation and plotted as a 1-D wave, and Fig.3.3(c) shows the local vector obtained with row concatenation. While Fig.3.3(b) is a rather simple unit function, Fig.3.3(c) contains a few pulses. This figure clearly demonstrates that different local vector formation schemes may lead to substantial discrepancies in the simplicity of the representation and characterization of the resultant local vector.

The explicit relationship between the elements of $\vec{V}(i,j)$ and the pixels in $\eta_B(i,j)$ depends apparently on the underlying mapping Ψ. For instance, if $\eta_B(i,j)$ is chosen to be a 3x3 square window and Ψ is of row concatenation, then the corresponding $\vec{V}(i,j)$ is composed of 9 elements which are related to the pixels in $\eta_B(i,j)$ by the following:

$$v_1(i,j) = f(i-1,j-1)$$

$$v_2(i,j) = f(i-1,j)$$

$$v_3(i,j) = f(i-1,j+1)$$

$$v_4(i,j) = f(i,j-1)$$

$$v_5(i,j) = f(i,j)$$

$$v_6(i,j) = f(i,j+1)$$

$$v_7(i,j) = f(i+1,j-1)$$

$$v_8(i,j) = f(i+1,j)$$

$$v_9(i,j) = f(i+1,j+1)$$

3.2.3 Vector Mapping

Assuming the local neighborhood $\eta_B(i,j)$ and the local vector formation scheme Ψ are appropriately defined and chosen, then local vector $\vec{V}(i,j)$ should be sufficiently characterized in order to describe the textural behavior of the given image $f(i,j)$ and to compute useful texture measures at pixel (i,j). If each element $v_k(i,j)$, $k=1,2,\ldots,K$ is regarded as a random variable (RV), then $\vec{V}(i,j)$ is a random vector. This random vector can be adequately described by its $K^{th}-order$ joint probability density function (pdf):

$$P(\vec{V}(i,j)) = P(v_1(i,j), v_2(i,j), \ldots, v_k(i,j), \ldots, v_K(i,j)) \tag{3.3}$$

If the original image $f(i,j)$ is assumed to be stationary and ergodic (so is $v_k(i,j)$, $k=1,2,\ldots,K$), the normalized histogram of $v_k(i,j)$ is a good approximation to the first-order pdf of $\vec{V}(i,j)$. Similarly, the second-order, or pair-to-pair joint pdfs of $\vec{V}(i,j)$ can be approximated by the so-called gray level co-occurrence matrices. In principle, higher-order pdfs may be similarly estimated. However, the estimation of the pdfs requires dramatically increasing computation and memory as the order of pdfs slowly increases. In practice, it is a forbidden task to estimate the pdfs of $\vec{V}(i,j)$ of order higher than 2 unless a parametric mathematical model is adopted.

To get out of this computational dilemma, one could think of two alternatives to the statistical description of $\vec{V}(i,j)$ outlined above. One may, through some operations on $\vec{V}(i,j)$, reduce the dimension of $\vec{V}(i,j)$ from K to K' ($< K$) without much loss of textural information, i.e., pack the textural information contained in $\vec{V}(i,j)$ into less random variables; or one would like to maintain the dimension K but transfer the information contained in higher-order pdfs to lower-order pdfs (say the first-order pdfs), e.g., in the ideal case, after some operation, the characterization of the K first-order pdfs or even the histograms of the K transformed RVs would be equivalent to the specification of the $K^{th}-order$ pdf of $\vec{V}(i,j)$. Certainly, the two alternatives can be combined as long as the important textural information in $\vec{V}(i,j)$ does not experience severe loss. With respect to texture feature extraction and classification/discrimination, these operations should be ideally such that the textural differences between distinct textures before the operations are retained or preferably "enhanced" after the operations. Now, let us formulate the above ideas mathematically.

A set of T operations on $\vec{V}(i,j)$, whether linear or non-linear, can be collectively and succinctly written as

$$\vec{G}(i,j) = \vec{H}[\vec{V}(i,j)] \tag{3.4a}$$

or

$$\begin{Bmatrix} g_1(i,j) \\ \cdot \\ \cdot \\ \cdot \\ g_T(i,j) \end{Bmatrix} = \begin{Bmatrix} h_1[v_1(i,j)\ v_2(i,j)\ \cdots\ v_K(i,j)] \\ \cdot \\ \cdot \\ \cdot \\ h_T[v_1(i,j)\ v_2(i,j)\ \cdots\ v_K(i,j)] \end{Bmatrix} \tag{3.4b}$$

where

- $T \leq K$;
- $\vec{V}(i,j) = [v_1(i,j)\ v_2(i,j)\ \cdots\ v_k(i,j)\ \cdots\ v_K(i,j)]^T$;
- $\vec{G}(i,j) = [g_1(i,j)\ g_2(i,j)\ \cdots\ g_t(i,j)\ \cdots\ g_T(i,j)]^T$;
- $\vec{H}[\bullet] = \left[h_1[\bullet]\ h_2[\bullet]\ \cdots\ h_t[\bullet]\ \cdots\ h_T[\bullet]\right]^T$.

If $T = K$ and a one-to-one correspondence exists between random vectors $\vec{G}(i,j)$ and $\vec{V}(i,j)$, then their pdfs are related by the familiar formula [20]:

$$p_G(\vec{G}(i,j)) = \frac{p_V(\vec{V}(i,j))}{|J|} = \frac{p_V(\vec{H}^{-1}(\vec{G}(i,j)))}{|J|}$$

where $|J|$ is the absolute value of the *Jacobian:*

$$J = \begin{vmatrix} \frac{\partial h_1}{\partial v_1} & \cdots & \frac{\partial h_K}{\partial v_K} \\ & \cdots\cdots & \\ & \cdots\cdots & \\ \frac{\partial h_K}{\partial v_1} & \cdots & \frac{\partial h_K}{\partial v_K} \end{vmatrix}$$

The operations defined by (3.4), together with the underlying local neighborhood $\eta_B(i,j)$ and local vector formation scheme Ψ, will hereafter be formally referred to as a set of **Local Vector Mappings** or **LVMs** for a shorthand notation. If $h_t[\bullet]$ is linear w.r.t. $\vec{V}(i,j)$, the corresponding mapping is called a *linear local vector mapping (LLVM)*; If it is non-linear, the specified mapping is called a *non-linear local vector mapping (NLVM).*

The expression of (3.4) leads to at least the following four interpretations of the T local vector mappings:

1. If $v_k(i,j)$ and $g_t(i,j)$ are taken as random variables, then $\vec{H}[\bullet]$ defines a random vector transformation from random vector $\vec{V}(i,j)$ of dimension K to random vector $\vec{G}(i,j)$ of dimension T;

2. If $\vec{F}(i,j)$ with basic elements $\vec{V}(i,j)$ is regarded as a K-dimensional multivariate image sequence $\{F_k(i,j): k=1,2,...K\}$, or a vector image as we have called it, then $\vec{H}[\bullet]$ defines an image sequence transformation which maps the K-dimensional vector image $\vec{F}(i,j)$ into a T-dimensional vector image $\vec{G}(i,j)$, whose element images will be called **texture feature images** in this thesis;

3. If $\vec{V}(i,j)$ is envisaged as a point in a R^K space, then $\vec{H}[\bullet]$ specifies a space mapping operation, though not necessarily one-to-one, which maps a point $\{v_1(i,j), v_2(i,j), \ldots, v_K(i,j)\}$ in the R^K space into a point $\{g_1(i,j), g_2(i,j), \ldots, g_T(i,j)\}$ in a R^T space; and finally

4. If $\vec{H}[\bullet]$ is considered to be a collection of linear or/and non-linear systems, then (3.4) defines a set of linear or/and non-linear filtering operations on $\vec{V}(i,j)$, where each

$h_t[\bullet]$, $t=1,2,\ldots,T$ is a particular linear or non-linear filter.

If $\vec{H}[\bullet]$ is entirely linear, i.e., all component filters $h_t[\bullet]$, $t=1,2,\ldots,T$, are linear in nature, then (3.4) can be rewritten as:

$$\begin{Bmatrix} g_1(i,j) \\ g_2(i,j) \\ \cdot \\ \cdot \\ \cdot \\ g_T(i,j) \end{Bmatrix} = [h_1^T\ x\vec{V}(i,j)\ h_2^T\ x\vec{V}(i,j)\ \cdots\ h_t^T\ x\vec{V}(i,j)\ \cdots\ h_T^T\ x\vec{V}(i,j)]^T \tag{3.5a}$$

or

$$\begin{Bmatrix} g_1(i,j) \\ \cdot \\ \cdot \\ \cdot \\ g_T(i,j) \end{Bmatrix} = \begin{Bmatrix} h_1(1)\ h_1(2)\ \cdots\ h_1(K) \\ \cdot \\ \cdot \\ \cdot \\ h_T(1)h_T(2)\ \cdots\ h_T(K) \end{Bmatrix} \begin{Bmatrix} v_1(i,j) \\ \cdot \\ \cdot \\ \cdot \\ v_K(i,j) \end{Bmatrix} \tag{3.5b}$$

i.e.,

$$g_t(i,j) = \sum_{l=1}^{K} h_t(l)\bullet v_l(i,j),\ t=1,2,\ldots,T \tag{3.5c}$$

where

$$\vec{h}_t=[h_t(1)\ h_t(2)\ \cdots\ h_t(k)\ \cdots\ h_t(K)]^T,\ t=1,2,\ldots,T$$

$h_t(k)$ for a fixed k is a constant or filter coefficient, and x in (3.5a) denotes inner product of vectors. Obviously, (3.5) specifies T **(FIR)** filters. Each FIR filter $h_t(k)$ has duration K. If the covariance matrix of $\vec{V}(i,j)$ is $\vec{C}_v(i,j)$, some simple matrix manipulations will tell us that the covariance matrix of $\vec{G}(i,j)$ defined by (3.5) is

$$\vec{C}_g(i,j) = E\left\{[(\vec{G}(i,j)-E[\vec{G}(i,j)]]\bullet[(\vec{G}(i,j)-E[\vec{G}(i,j)]]^T\right\} = \vec{H}\bullet\vec{C}_v(i,j)\bullet\vec{H}^T \tag{3.6}$$

where

$$\vec{H} = [h_1^T\ \ h_2^T\ \ \cdots\ h_T^T\ ]^T$$

$E[y]$ represents the expectation of y, and the following rule of matrices is used

$$(AB)^T = B^T A^T$$

3.2.4 LLCM as a Special Case of LVM

We claim that many of the LLCM (local linear convolution method) techniques discussed in Section 2.2.9 can be included as a subset of the LVM scheme presented in this chapter. To make this statement more tangible, let us concentrate on how Laws' approach [6] (the most popular LLCM technique) and Ade's approach [8] may be considered and specified under the general LVM scheme.

After an extensive trial-and-error searching, Laws discovered a number of 2-D convolution masks useful for texture feature measurement. The four most important 5x5 masks, according to his data set, are given in Fig.3.4.

-1	-4	-6	-4	-1
-2	-8	-12	-8	-2
0	0	0	0	0
2	8	12	8	2
1	4	6	4	1

(a)

1	-4	6	-4	1
-4	16	-24	16	-4
6	-24	36	-24	6
-4	16	-24	16	-4
1	-4	6	-4	1

(b)

-1	0	2	0	-1
-2	0	4	0	-2
0	0	0	0	0
2	0	-4	0	2
1	0	-2	0	1

(c)

-1	0	2	0	-1
-4	0	8	0	-4
-6	0	12	0	-6
-4	0	8	0	-4
-1	0	2	0	-1

(d)

Figure 3.4. Four Laws' 5x5 Masks Used in Texture Analysis. (a) Horizontal edge; (b) High frequency spot; (c) V-shape and (d) Vertical line.

These masks can be easily described under the LVM scheme. For instance, the horizontal edge mask shown in Fig.3.4(a) can be specified by

$$\begin{cases} \eta_B(i,j) = \textit{a 5x5 region centered at } (i,j); \\ \Psi = \textit{row concatenation}; \\ \vec{h}_t = [-1\ -4\ -6\ -4\ -1\ -2\ -8\ -12\ -8\ -2\ 0\ 0\ 0\ 0\ 0\ 2\ 8\ 12\ 8\ 2\ 1\ 4\ 6\ 4\ 1]^T \end{cases}$$

or equivalently

$$\begin{cases} \eta_B(i,j) = \textit{a 5x5 region centered at } (i,j); \\ \Psi = \textit{column concatenation}; \\ \vec{h}_t = [-1\ -2\ 0\ 2\ 1\ -4\ -8\ 0\ 8\ 4\ -6\ -12\ 0\ 12\ 6\ -4\ -8\ 0\ 8\ 4\ -1\ -2\ 0\ 2\ 1]^T \end{cases}$$

For feature extraction, Laws proceeded with a local masking operation which can be compactly described using our LVM terms as

$$g_t(i,j) = \vec{h}_t x \vec{V}(i,j)$$

where $\vec{V}(i,j) = \Psi[\eta_B(i,j)]$. He then computed and defined the sum of the squared $g_t(i,j)$ in small regions as texture features, which he referred to as *texture energy measures*. The representation and description for the other masks shown in Fig.3.4 is straightforward.

While in Laws's approach, the masks are independent of the textures concerned and of the masking region $\eta_B(i,j)$, Ade proposed to apply *dynamic* masks, which are texture-dependent and position-variant. Under the LVM scheme, Ade's approach can be conveniently described by

$$\begin{cases} \eta_B(i,j) = \textit{a 3x3 region centered at } (i,j); \\ \Psi = \textit{row concatenation}; \\ g_t(i,j) = \vec{h}_t x \vec{V}(i,j) \end{cases}$$

where the texture feature image $g_t(i,j)$ is called a *principal component image* by him, and $\vec{h}_t$ is an eigenvector of the covariance matrix $\vec{C}_v(i,j)$ of the local vector $\vec{V}(i,j)$.

Some remarks are now in order before concluding this section. Firstly, although one may argue that some useful operations can be directly defined on a 2-D $\eta_B(i,j)$ and thus the mapping Ψ from $\eta_B(i,j)$ to a 1-D vector $\vec{V}(i,j)$ seems unnecessary, the introduction of the LVM scheme offers us a consistent, convenient and succinct representation and explanation of local-operation-based approaches to texture analysis as we just discussed. In addition, 1-D problems are certainly preferable to 2-D ones; secondly, the LVM scheme may guide one to define useful local operations for texture analysis by changing the three degrees of freedom $\eta_B(i,j)$, Ψ and $h_t[\vec{V}(i,j)]$, while most of the local masks previously used are determined heuristically after a

time-consuming trial-and-error searching process; finally, we maintain that it will take a long period to exhaust the possibilities of defining useful local operations for texture analysis under the general LVM scheme.

3.3 Computing LVMs: Some Examples

To illustrate what has been discussed in the last section, we shall define in this section some local vector mappings which have proven to be effective for texture description.

3.3.1 Linear Local Vector Mapping (LLVM)

For a set of linear mappings, i.e., the LLVMs defined by (3.5), it can be shown that the optimal mapping or transform is the well-known *Karhunen-Loeve Transform (KLT)* [4]. The KLT is optimal in that it generates uncorrelated random variables or diagonalizes the covariance matrix $\vec{C}v(i,j)$, and is capable of packing the maximum amount of textural information into (or causing the minimum mean square error of representation with) a given number of mapped random variables. From the standpoint of easy implementation, however, the KLT is not a good choice as it is usually rather time-consuming. Fortunately, some computationally cheaper transforms have been shown to be equivalent to the KLT in an asymptotical fashion [21]. One of these transforms is the *discrete cosine transform (DCT)*. So, let us first discuss the DCT of a local vector $\vec{V}(i,j)$.

A. Local Discrete Cosine Mapping/Transform

Consider a local vector $\vec{V}(i,j)$ as a 1-D digital signal of points $v_1(i,j),v_2(i,j),\ldots,v_k(i,j),\ldots,v_K(i,j)$, then its DCT may be defined as

$$g_t(i,j) = \frac{2c(t)}{K}\sum_{k=1}^{K} v_k(i,j)COS\frac{(2k-1)t\pi}{2K},\ t=0,1,2,\ldots,T \tag{3.7}$$

where

$$c(t) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } t=0 \\ 1, & otherwise \end{cases}$$

Since (3.7) defines $T+1$ linear operations, it can be rewritten in a matrix form as

$$\vec{G}(i,j) = \vec{H} \bullet \vec{V}(i,j) \tag{3.8}$$

where

$$\vec{G}(i,j) = [g_0(i,j)\ g_1(i,j)\ \cdots\ g_t(i,j)\ \cdots\ g_T(i,j)]^T$$

$$\vec{V}(i,j) = [v_1(i,j)\ v_2(i,j)\ \cdots\ v_k(i,j)\ \cdots\ v_K(i,j)]^T$$

and $\vec{H}$ is a $(T+1)xK$ matrix whose element $h(t,k)$ is given by

$$h(t,k) = \frac{2c(t)}{K} COS \frac{(2k-1)t\pi}{2K} \tag{3.9}$$

Recalling (3.4) and (3.5), one knows that (3.8) actually defines $T+1$ local vector mapping operations with the t^{th} feature image $g_t(i,j)$ being specified by

$$\begin{aligned} g_t(i,j) &= h_t[\vec{V}(i,j)] = \vec{h}_t^T\ x\vec{V}(i,j) \\ &= [h_t(1)\ h_t(2)\ \cdots\ h_t(k)\ \cdots\ h_t(K)]x\vec{V}(i,j) \end{aligned} \tag{3.10}$$

where

$$h_t(k) = h(t,k)$$

The operation specified by (3.10) will be called a *local discrete cosine mapping* or *transform (LDCT)*.

Fig.3.5 shows some of the feature images $g_t(i,j)$ defined by (3.10). The original image $f(i,j)$ is the Calf texture taken from **Brodatz** [22] and is shown in Fig.3.5(a). Fig.3.6 presents the histograms of the images shown in Fig.3.5.

Note, in the above discussions, no assumptions have been made about the local neighborhood $\eta_B(i,j)$ and the local vector formation scheme Ψ.

B. Local Polynomial Mapping/Transform

Another set of linear operations which are found to be effective for extracting powerful texture features are defined by

$$\vec{G}(i,j) = \vec{H}[\vec{V}(i,j)] = [\vec{h}_1^T\ x\vec{V}(i,j)\ \vec{h}_2^T\ x\vec{V}(i,j)\ \cdots\ \vec{h}_6^T\ x\vec{V}(i,j)]^T \tag{3.11}$$

where

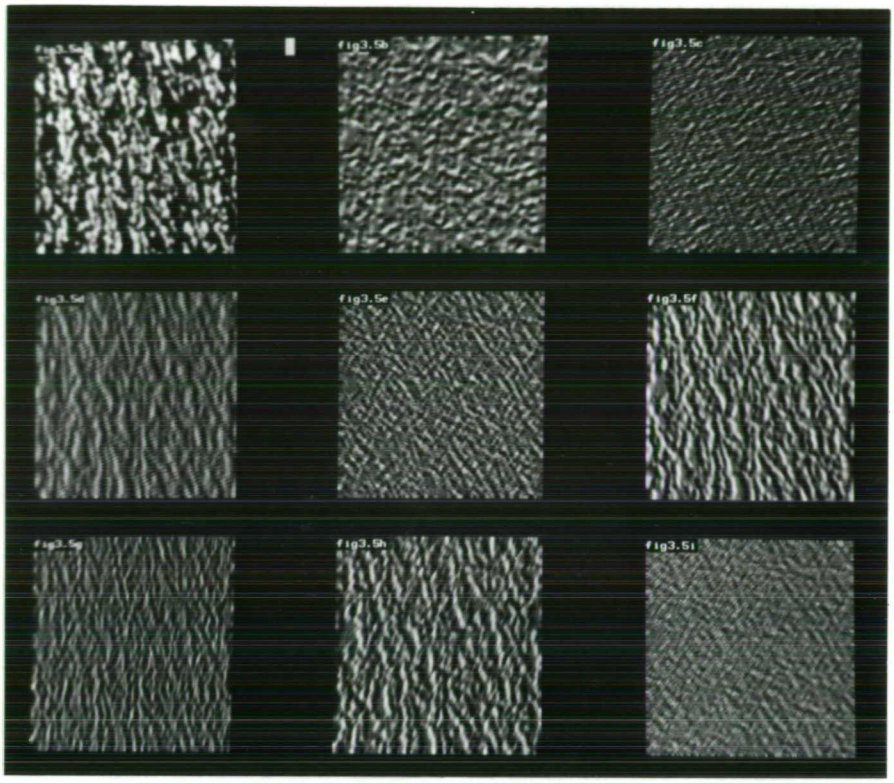


Figure 3.5. Examples of LDCTed Images. (a) Original Calf texture; (b) $g_1(i,j)$; (c) $g_2(i,j)$; (d) $g_3(i,j)$; (e) $g_4(i,j)$; (f) $g_5(i,j)$; (g) $g_6(i,j)$; (h) $g_7(i,j)$ and (i) $g_8(i,j)$.

$$\vec{G}(i,j) = [g_1(i,j)\ g_2(i,j)\ \cdots\ g_6(i,j)]^T$$

$$h_1 = \frac{1}{9}[\,-1\ 2\ -1\ 2\ 5\ 2\ -1\ 2\ -1\,]^T$$

$$h_2 = \frac{1}{6}[\,-1\ 0\ 1\ -1\ 0\ 1\ -1\ 0\ 1\,]^T$$

$$h_3 = \frac{1}{6}[\,1\ 1\ 1\ 0\ 0\ 0\ -1\ -1\ -1\,]^T$$

$$h_4 = \frac{1}{4}[\,-1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ -1\,]^T$$

$$h_5 = \frac{1}{6}[\,1\ -2\ 1\ 1\ -2\ 1\ 1\ -2\ 1\,]^T$$

$$h_6 = \frac{1}{6}[\,1\ 1\ 1\ -2\ -2\ -2\ 1\ 1\ 1\,]^T$$

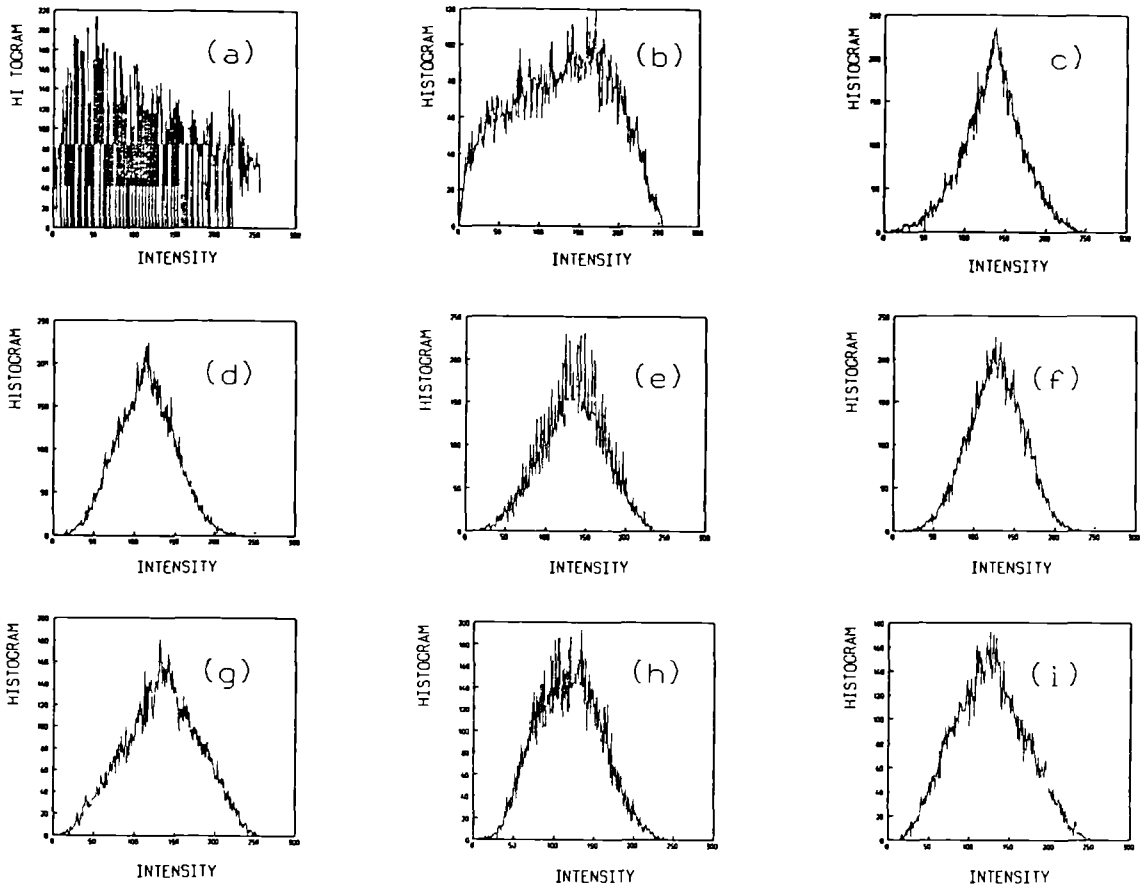


Figure 3.6. Histograms of Images Shown in Fig.3.5(a) to (i) Respectively.

It is obvious that for (3.11) to be valid, the local vector $\vec{V}(i,j)$ involved must be of dimension 9. If the underlying local neighborhood $\eta_B(i,j)$ is assumed to be a 3x3 square region, and the mapping Ψ is of row concatenation, then, it can be shown that the local vector mappings defined in (3.11) are equivalent to fitting the discrete local surface specified by $\eta_B(i,j)$ by a continuous second-order polynomial function

$$\hat{F}(i,j) = a_1 + a_2 i + a_3 j + a_4 ij + a_5 i^2 + a_6 j^2 \tag{3.12}$$

under the linear least squares criterion of approximation (See Chapter 4). $a_t, t=1,2,\ldots,6$ in (3.12) are the fitting coefficients corresponding to the images $g_t(i,j), t=1,2,\ldots,6$ in (3.11) respectively. For this reason, the operations defined in (3.11) will be referred to as a set of *local polynomial mappings* or *transforms (LPTs)*.

Some images computed by (3.11) are illustrated in Fig.3.7.

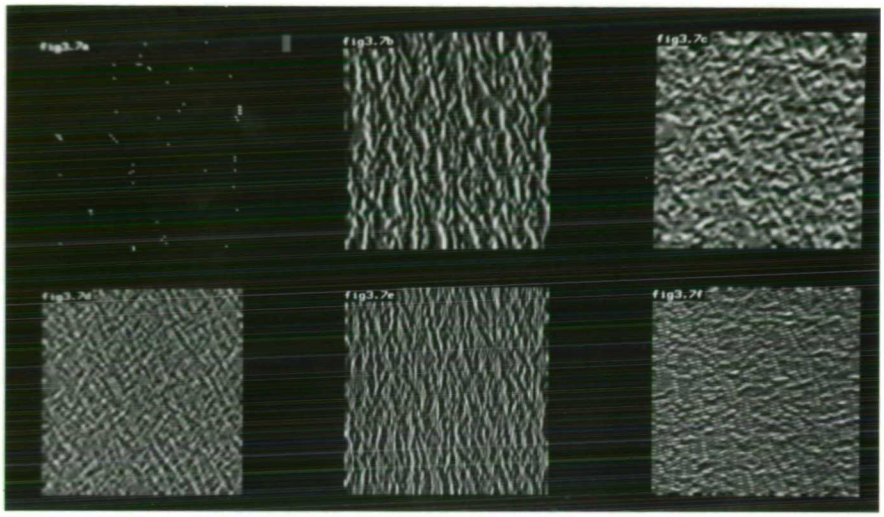


Figure 3.7. Examples of LPTed Images. (a) $g_1(i,j)$; (b) $g_2(i,j)$; (c) $g_3(i,j)$; (d) $g_4(i,j)$; (e) $g_5(i,j)$ and (f) $g_6(i,j)$.

The original image $f(i,j)$ is the Calf texture as used in Fig.3.5. Fig.3.8 shows the histograms of the images in Fig.3.7.

3.3.2 Non-Linear Local Vector Mapping (NLVM)

Unlike LLVMs, an optimal non-linear mapping similar to the KLT does not exist. From the point of view of texture feature extraction, we shall define in this sub-section some NLVMs which are intuitively useful.

A. Local Variance Mapping

The local variance mapping describes the smoothness of the curve composed of the elements of the underlying local vector $\vec{V}(i,j)$, and is defined to be

$$g(i,j) = h_{var}[\vec{V}(i,j)] = \frac{1}{K}\sum_{k=1}^{K}[v_k(i,j)-\mu]^2$$

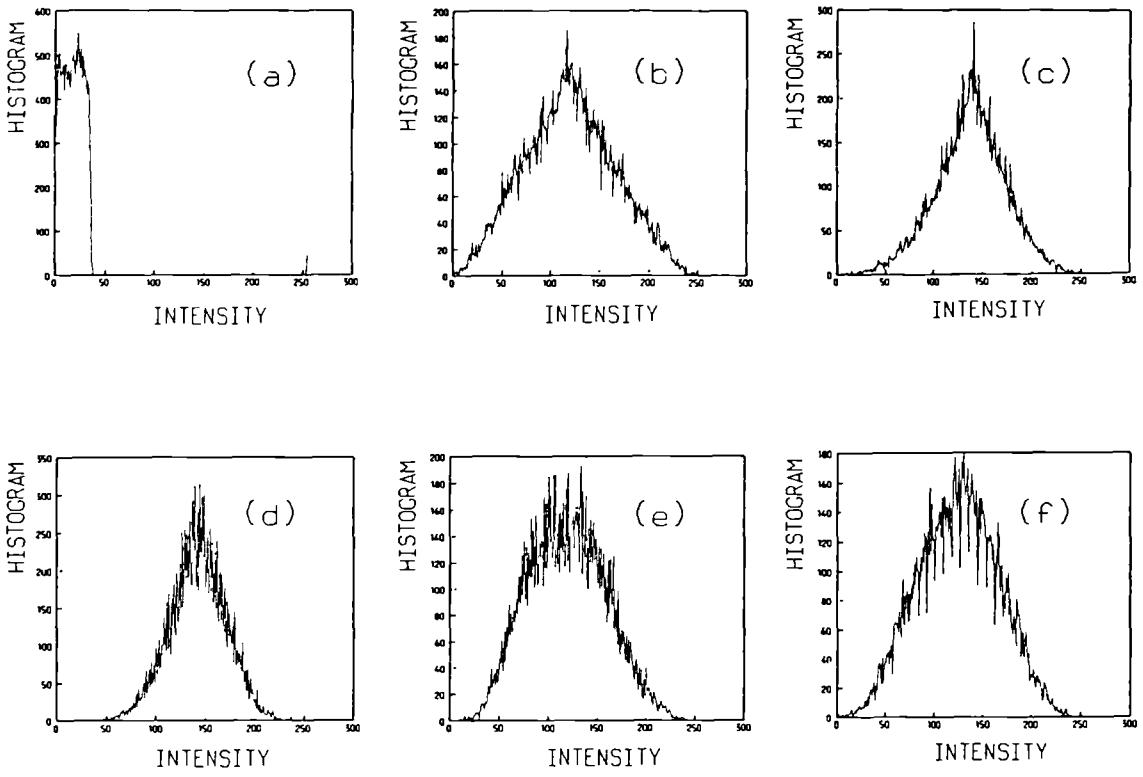


Figure 3.8. Histograms of Images Shown in Fig.3.7(a) to (f) Respectively.

where μ is the average value of $v_k(i,j)$, $k-1,2,\ldots,K$.

B. Local Maximum Mapping

$$g(i,j) = h_{\max}[\vec{V}(i,j)] = \max_k[v_k(i,j)]$$

C. Local Minimum Mapping

$$g(i,j) = h_{\min}[\vec{V}(i,j)] = \min_k[v_k(i,j)]$$

D. Local Extrema Difference Mapping

The local extrema difference mapping describes the difference between the maximum and the minimum of $\vec{V}(i,j)$, in other words, it characterizes the local dynamic range of gray levels in the input image $f(i,j)$. It is defined as

$$g(i,j) = h_{led}[\vec{V}(i,j)] = h_{\max}[\vec{V}(i,j)] - h_{\min}[\vec{V}(i,j)]$$

E. Local Generalized Run Length Mapping

If we assume that the local neighborhood $\eta_B(i,j)$ is symmetrical and contains an odd number of K pixels with pixel (i,j) being the middle element of the local vector $\vec{V}(i,j)$, then the local generalized run length mapping is defined by

$$g(i,j) = h_{grl}[\vec{V}(i,j)] = \sum_{k=1}^{K} U[\Delta - |v_k(i,j) - v_{\frac{K+1}{2}}(i,j)|]$$

where Δ is some threshold and $U[\bullet]$ is the unit step function defined by

$$U(t) = \begin{cases} 0, & t \le 0; \\ 1, & t > 0; \end{cases}$$

F. Local Horizontal Transition Mapping

The local horizontal transition mapping to be defined aims to characterize the horizontal correlations between pixels in $\eta_B(i,j)$. If the local neighborhood $\eta_B(i,j)$ is an NxN square region and the local vector formation scheme Ψ is of row concatenation, then the mathematical expression of the mapping is given by

$$g(i,j) = h_{lht}[\vec{V}(i,j)] = \sum_{m=1}^{N} \sum_{n=(m-1)N+2}^{mN} [1 - \delta[v_n(i,j) - v_{n-1}(i,j)]]$$

where $\delta[\bullet]$ is the delta function.

G. Local Vertical Transition Mapping

Similarly, a mapping called the local vertical transition mapping can be defined to describe the vertical intensity correlations between the pixels in $\eta_B(i,j)$. If $\eta_B(i,j)$ is an NxN square region and Ψ is of column concatenation, then the mathematical description of the local vertical transition mapping is identical to that of the local horizontal transition mapping in form, that is

$$g(i,j) = h_{lvt}[\vec{V}(i,j)] = \sum_{m=1}^{N} \sum_{n=(m-1)N+2}^{mN} [1 - \delta[v_n(i,j) - v_{n-1}(i,j)]]$$

Fig.3.9 shows these NLVMed images for the Calf texture shown in Fig.3.5(a).

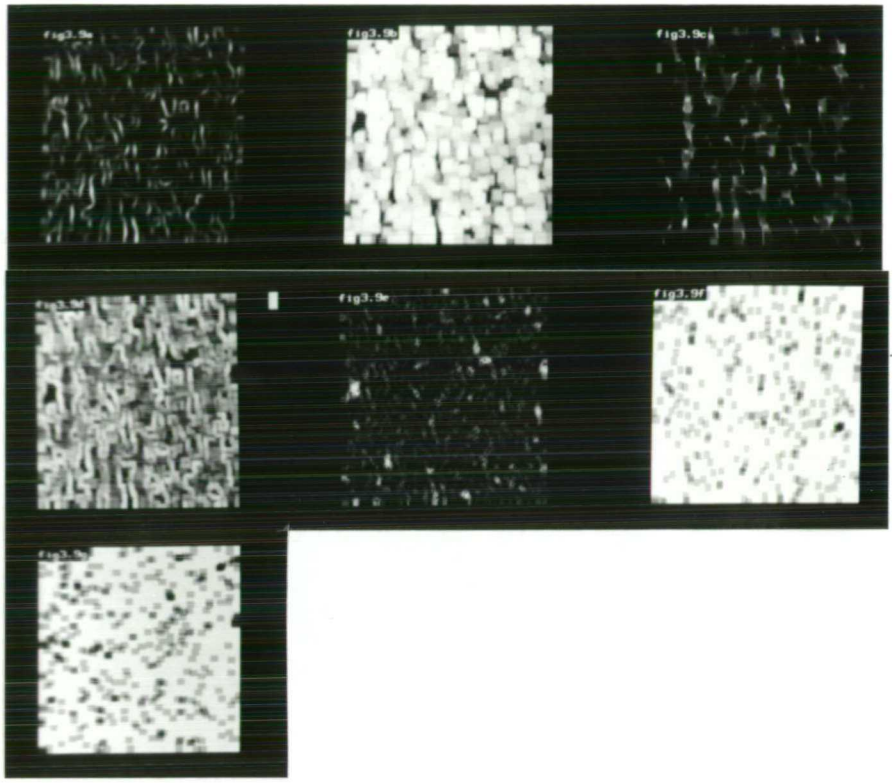


Figure 3.9. Examples of NLVMed Images. (a) $h_{var}[\vec{V}(i,j)]$; (b) $h_{\max}[\vec{V}(i,j)]$; (c) $h_{\min}[\vec{V}(i,j)]$; (d) $h_{led}[\vec{V}(i,j)]$; (e) $h_{grl}[\vec{V}(i,j)]$; (f) $h_{lhi}[\vec{V}(i,j)]$ and (g) $h_{lvt}[\vec{V}(i,j)]$.

To end this section, two points are worthwhile mentioning. First of all, the identical expressions of the local horizontal and vertical transition mappings imply that in order to uniquely determine a LVM, the local neighborhood $\eta_B(i,j)$, the vector formation process Ψ, and the actual mapping operation $h_t[\bullet]$ have to be simultaneously specified. The same mapping operation $h_t[\bullet]$ could have different physical interpretations under distinct $\eta_B(i,j)$ and Ψ. Secondly, we have given in this section only some simple LVMs for the purpose of illustrating the general framework of the LVM scheme. Other useful local operations could be equally well

included.

3.4 LVM Texture Features

The ultimate goal of LVM operations is to extract useful texture measures from a given image $f(i,j)$. In this section, we shall define some texture features based on the LVM operations defined in the last section. The relations between the features defined and some of the visually perceived texture features are also established in this section.

3.4.1 Definitions of LVM Texture Features

If a given set of T LVMs on a local vector $\vec{V}(i,j)$ $h_t[\bullet]$, $t=1,2,...T$ generate T independent feature images $g_t(i,j)$, $t=1,2,...,T$, then the first-order pdfs of $g_t(i,j)$ would be sufficient to characterize $\vec{V}(i,j)$ as in this case the joint pdfs of $\vec{G}(i,j)$ can be specified by the appropriate products of individual 1-D pdfs of $g_t(i,j)$. Under normal distributions, only the means and the standard deviations of $g_t(i,j)$ are required. Bearing these assumptions in mind, we simply use the first four moments namely mean, variance, skewness and Kurtosis of the feature images $g_t(i,j)$, $t=1,2,...,T$ as texture features. So, mathematically, for each LVM $h[\bullet]$ and the corresponding feature image $g(i,j)$ of size IxJ, we define

- Texture Feature TF_1:

$$TF_1 = \frac{1}{IxJ}\sum_i\sum_j g(i,j) = \frac{1}{IxJ}\sum_i\sum_j h[\vec{V}(i,j)] \tag{3.13}$$

- Texture Feature TF_2:

$$TF_2 = \frac{1}{IxJ}\sum_i\sum_j [g(i,j)-TF_1]^2 = \frac{1}{IxJ}\sum_i\sum_j [h[Vij]-TF_1]^2 \tag{3.14}$$

- Texture Feature TF_3:

$$TF_3 = \frac{1}{IxJ}\sum_i\sum_j \frac{[g(i,j)-TF_1]^3}{TF_2} = \frac{1}{IxJ}\sum_i\sum_j \frac{[h[\vec{V}(i,j)]-TF_1]^3}{TF_2} \tag{3.15}$$

- Texture Feature TF_4:

$$TF_4 = \frac{1}{IxJ}\sum_i\sum_j \frac{[g(i,j)-TF_1]^4}{TF_2} = \frac{1}{IxJ}\sum_i\sum_j \frac{[h[\vec{V}(i,j)]-TF_1]^4}{TF_2} \qquad (3.16)$$

where TF_3 and TF_4 , i.e., the skewness and Kurtosis, have been normalized w.r.t the variance TF_2.

In the following few paragraphs, we shall define the LVM texture features which will be employed in the subsequent texture classification experiments.

A. LDCT Texture Features

For the local discrete cosine mappings defined in (3.8), we shall only use the first 9 feature images $g_0(i,j)$, $g_1(i,j)$, $\cdots$, and $g_8(i,j)$. These images will be denoted by $LDCT.a_0(i,j)$, $LDCT.a_1(i,j)$, $\cdots$, and $LDCT.a_8(i,j)$ respectively. For each feature image, TF_1, TF_2, TF_3 and TF_4 are calculated. Thus, for a given image $f(i,j)$, a total of 4x9=36 LDCT texture features are extracted. However, $LDCT.a_0(i,j)$ is the result of a local averaging operation on $f(i,j)$, and since all textures in our database have been preprocessed by means of histogram equalization, $LDCT.a_0(i,j)$ is not expected to carry any useful information for texture discrimination. Thus, for a given image $f(i,j)$, only 4x8=32 LDCT texture features are actually extracted and used.

B. LPT Texture Features

Corresponding to LPT, there are 6 feature images $g_1(i,j)$, $g_2(i,j)$, ..., $g_6(i,j)$, which will be redenoted by $LPT.a_{00}(i,j)$, $LPT.a_{10}(i,j)$, $LPT.a_{01}(i,j)$, $LPT.a_{11}(i,j)$, $LPT.a_{20}(i,j)$, and $LPT.a_{02}(i,j)$ respectively. From Fig.3.7, one can see that $LPT.a_{00}(i,j)$ is not expected to be useful for texture characterization, so it is omitted. After calculations of TF_1, TF_2, TF_3 and TF_4 are performed for the other five feature images, one extracts 4x5=20 LPT texture features from a given image $f(i,j)$.

C. NLVM Texture Features

In Section 3.3.2, we defined 7 NLVMs. The corresponding 7 feature images will be represented by $NLVM.var(i,j)$, $NLVM.max(i,j)$, $NLVM.min(i,j)$, $NLVM.led(i,j)$, $NLVM.grl(i,j)$, $NLVM.lht(i,j)$ and $NLVM.lvt(i,j)$ respectively. By computing TF_1, TF_2, TF_3 and TF_4 for each of these 7 feature images, one gets 28 NLVM texture features.

Now, we have defined 32 LDCT texture features, 20 LPT texture features and 28 NLVM texture features, which makes a total of 80 LVM features to be tested in texture classification in the next section. Tables 3.1 to 3.4 show the LDCT TF_3s, the LPT TF_3s, the NLVM TF_1s and TF_2s extracted from each texture class in the database respectively. Other features are tabulated in [19]. Generation of these features will be explained in the next section.

Table 3.1. The TF_3 Features from The Local Discrete Cosine Transform/Mapping

FEATURE IMAGE / TEXTURE	LDCT.a1	LDCT.a2	LDCT.a3	LDCT.a4	LDCT.a5	LDCT.a6	LDCT.a7	LDCT.a8
Beach Sand	-4.500518	2.136578	-1.767426	-1.622386	-2.035195	0.5486963	2.113003	0.1531227
Calf	-5.194580	1.457447	-1.864354	-0.6446491	-2.738254	2.005965	2.137539	0.1146491
Cotton Canvas	-3.759499	3.169134	-3.486961	1.886599	-9.028043	5.050972	15 07853	-2.105487
Grass Lawn	-1.490027	2.161608	-1.731635	-1.403849	-4.896209	-1.145861	3.002179	-0 2796280
Herring-bone	1.296175	-1.259725	-3.319711	0.1575390	-6 343473	-1.890940	4.375171	0.1115666
Pressed Cork	-4.593649	1.292551	-0 4743148	-2.701194	-2.720966	1.652104	0 6148248	1 704481
Oriented Straw	1.350974	0.06900839	6.647482	-0 6816247	-9.909585	4.471372	4 452701	1.729018
Wood Grain	-2 244811	0 3242628	3.040824	0.3667743	6 863795	0 7998689	-3.856697	0 03786577
d53	-7.650908	-3.686948	-6.646344	-0.5902335	9 218411	9.005033	6.537112	-1.263892
hand-made Paper	-5.936265	0.5953072	0.04661481	-0.1453972	0 6549756	-0 7278007	0 6531643	-0.0259815

3.4.2 LVM Texture Features versus Visual Texture Features

A desirable property of computational texture features is their ability to indicate some of the visual aspects of the textures concerned as human beings are the ultimate observers and judges of images. As investigated by **Tamura** *et al.* [23], the six common features which the human being uses to distinguish textures are coarseness (versus fineness), contrast (high contrast versus low contrast), directionality, line-likeness (line-like versus blob-like), regularity (regular versus irregular) and roughness (rough versus smooth). These features may be called *visual texture features*. Experimental results reveal that in almost all situations, coarseness, directionality and regularity are the three most important visual features [23]. In the following paragraphs, let us examine how the LVM features just defined may be correlated with some of

Table 3.2. The TF_3 Features from The Local Polynomial Transform/Mapping

TEXTURE \ FEATURE IMAGE	LPT.a10	LPT.a01	LPT.a11	LPT.a20	LPT.a02
Beach Sand	3.959440	-3.012475	-0.4896876	0.8230534	2.908304
Calf	2.675762	-5.578301	-1.897057	3.008951	0.01859014
Cotton Canvas	19.60697	-3.594029	-2.774701	7.576499	6.453653
Grass Lawn	4.746506	-1.023621	-0.4084101	-1.718774	1.415159
Herring-bone	8.615847	-1.911830	0.2358666	-2.836392	-0.2325375
Pressed Cork	1.734858	-3.451416	1.401982	2.478164	2.235519
Oriented Straw	9.929277	4.949886	1.782777	6.707087	-0.5621271
Wood Grain	-8.023869	-0.330609	-0.4545313	1.199797	-0.02613597
d53	11.82073	-2.236443	-4.841856	13.50759	-2.847527
hand-made Paper	2.225188	-3.242252	0.05549838	-1.091693	0.3314562

the visual texture features.

A. LLVM Texture Features versus Visual Texture Feature

Being coarse (fine) implies low (high) intensity variations or poor (rich) high frequency components. Thus, mean values of LDCT texture feature images, i.e., LDCT TF_1s, are directly related to the coarseness of a texture. On the contrary, such an explicit relation does not exist between LLVM texture features and directionality or regularity. Nevertheless, careful studies on the LLVM features will uncover their implicit relationship.

If a LVM is considered as a random vector transformation operation, then each random variable in a LLVMed image or feature image $g_t(i,j)$ is a linear combination of a set of random variables in the original image field $f(i,j)$, or equivalently, $g_t(i,j)$ is the sum of a set of random variables. For a local neighborhood $\eta_B(i,j)$ of reasonable size, $g_t(i,j)$ is the sum of a number of random variables. By recalling the *Central Limit Theorem* [20], one understands that this implies the feature image $g_t(i,j)$ tends to be Gaussian-distributed (see Fig.3.6 and Fig.3.8). However, the theorem requires that the random variables to be summed are independent of each other,

Table 3.3. The TF_1 Features from The Defined Non-linear Local Vector Mappings

TEXTURE \ FEATURE IMAGE	NLVM.VAR	NLVM.MAX	NLVM.MIN	NLVM.LED	NLVM.GRL	NLVM.LHT	NLVM.LVT
Beach Sand	2699.457	214.7226	45.97448	168.7481	13.57499	19.59134	19.52943
Calf	3182.861	222.7890	38.02905	184.7600	11.79422	19.61668	19.52633
Cotton Canvas	4487.195	237.8120	20.34191	217.4700	8.984949	19.71319	19.75466
Grass Lawn	3113.094	219.1759	35.73843	183.4375	12.11638	19.54329	19.49922
Herring-bone	3532.968	223.0630	28.79439	194.2689	10.79511	19.58138	19.72304
Pressed Cork	2858.445	215.7315	41.12552	174.6061	13.01927	19.38422	19.44351
Oriented Straw	2305.877	208.0632	53.18740	154.8761	14.63438	19.60886	19.58517
Wood Grain	3455.900	214.7103	39.19879	175.5117	12.41332	19.66642	19.05532
d53	4230.196	232.5326	23.94367	208.5891	9.558065	19.79991	19.72865
hand-made Paper	3023.267	218.5969	38.77994	179.8170	12.49970	19.56849	19.72843

from which we may conclude that the more random the original image $f(i,j)$, the more likely $g_t(i,j)$ tends to be Gaussian. Since the directionality and regularity in the original image means some kinds of correlations between the neighboring pixels or random variables, regular textures or textures with strong directionality will have LLVMed images which are far from Gaussian. The dominant feature of a Gaussian distribution is its bell-like symmetry. A true Gaussian random variable has zero skewness, i.e., $TF_3 - 0$. One can see, from Tables 3.1 and 3.2, random textures Hand-made Paper, Beach Sand, Calf, Grass Lawn and Pressed Cork all have relatively small LDCT and LPT TF_3s whereas the two regular textures Cotton Canvas and d53 all have large LDCT and LPT TF_3s.

B. NLVM Texture Features versus Visual Texture Features

Unlike LLVM texture features, the NLVM texture features defined have some explicit correspondences to some visual texture features such as coarseness, directionality and regularity. This is not difficult to understand if one recalls the definitions of these NLVM

Table 3.4. The TF_2 Features from The Defined Non-linear Local Vector Mappings

FEATURE IMAGE / TEXTURE	NLVM.VAR	NLVM.MAX	NLVM.MIN	NLVM.LED	NLVM.GRL	NLVM.LHT	NLVM.LVT
Beach Sand	2402689	1491.734	1622.181	1795.408	44.85094	0.4667729	0.6499433
Calf	2592598	1235.743	1198.102	1441.099	26.45067	0.3935319	0.5062370
Cotton Canvas	1722635	249.5420	344.0557	486.1276	8.454125	0.2804432	0.2481523
Grass Lawn	2281604	1219.302	1210.993	1381.522	30.97143	0.5065299	0.5521626
Herring-bone	2045027	846.4141	768.8043	984.4811	18.41392	0.5011709	0.3003469
Pressed Cork	2480576	1447.440	1746.241	1793.568	37.89499	0.6683496	0.6963632
Oriented Straw	2094966	2053.600	1915.284	1827.079	44.61764	0.5022287	0.5285182
Wood Grain	4652943	1297.671	1434.175	1912.858	28.96321	0.7836777	1.814296
d53	2261556	368.4061	469.1037	598.0498	10.51280	0.1864431	0.2566914
hand-made Paper	2489926	1512.903	1392.840	1580.181	32.12897	0.4881074	0.2824541

operations. In particular, some of the NLVM features are obviously related to coarseness since most of the defined NLVM operations take into account the intensity variations in a local region such as local variance mapping, local extrema difference mapping and local generalized run length mapping. For a coarse texture, one will have small local variances, small local extrema differences and long generalized run lengths. This means NLVM TF_1s are good indicators of texture coarseness.

The local horizontal and vertical transition mappings, on the other hand, are good indicators of directionality since the former measures the horizontal pixel correlation and the latter the vertical pixel correlation. Thus, textures with horizontal directionality will be expected to have small *NLVM lht* TF_1s and textures with vertical directionality to have small *NLVM lvt* TF_1s. Wood Grain, for instance, is the texture with the strongest vertical directionality among the textures in the database (See Fig.3.10) and hence is expected to have the smallest NLVM.lvt TF_1. This is indeed so as shown in Table 3.3.

For a regular texture, properties from various local regions of reasonable size, will be similar. Putting this another way, one says that regular textures have small local property variations, while irregular ones have large variations. A conclusion which can be drawn from this is that NLVM TF_2s are good measures of texture regularity. This conclusion is justified in Table 3.4. Among all textures in the database (See Fig.3.10), Cotton Canvas and d53 are the two regular textures and indeed have relatively small NLVM TF_2s as shown in Row 4 and 10 of Table 3.4.

In summary, we state that

1. LDCT TF_1s and some of NLVM TF_1s are directly related to coarseness;
2. LLVM TF_3s are good indicators of directionality and/or regularity;
3. TF_1s of local horizontal and vertical transition mappings are strongly correlated with horizontal and vertical directionality; and
4. NLVM TF_2s reflect texture regularity.

3.5 Texture Classification by LVM: Experimental Results

In this section, we shall test the effectiveness of the defined LVM features in the context of texture classification, and summarize the experimental results obtained. The results will be given in the form of *Classification Confusion Matrix* or *Classification Accuracy Matrix.*

3.5.1 Texture Feature Vector Generation

The database chosen in this chapter consists of 10 natural textures taken from **Brodatz** [22] and is shown in Fig.3.10. All samples are digitized into 128x128 square grids and represented by 8 bits. They are also preprocessed by means of histogram equalization in order to compensate for the lighting differences and to remove texture discrimination based on first-order statistics.

For the purpose of prototype feature vector generation (i.e., classifier training), 75 texture samples for each texture class are derived from the original texture set. These 75 samples consist of 49 32x32 samples, 25 64x64 samples and 1 128x128 sample. For each texture sample, all useful texture features are calculated as discussed in the preceding section. In other words,

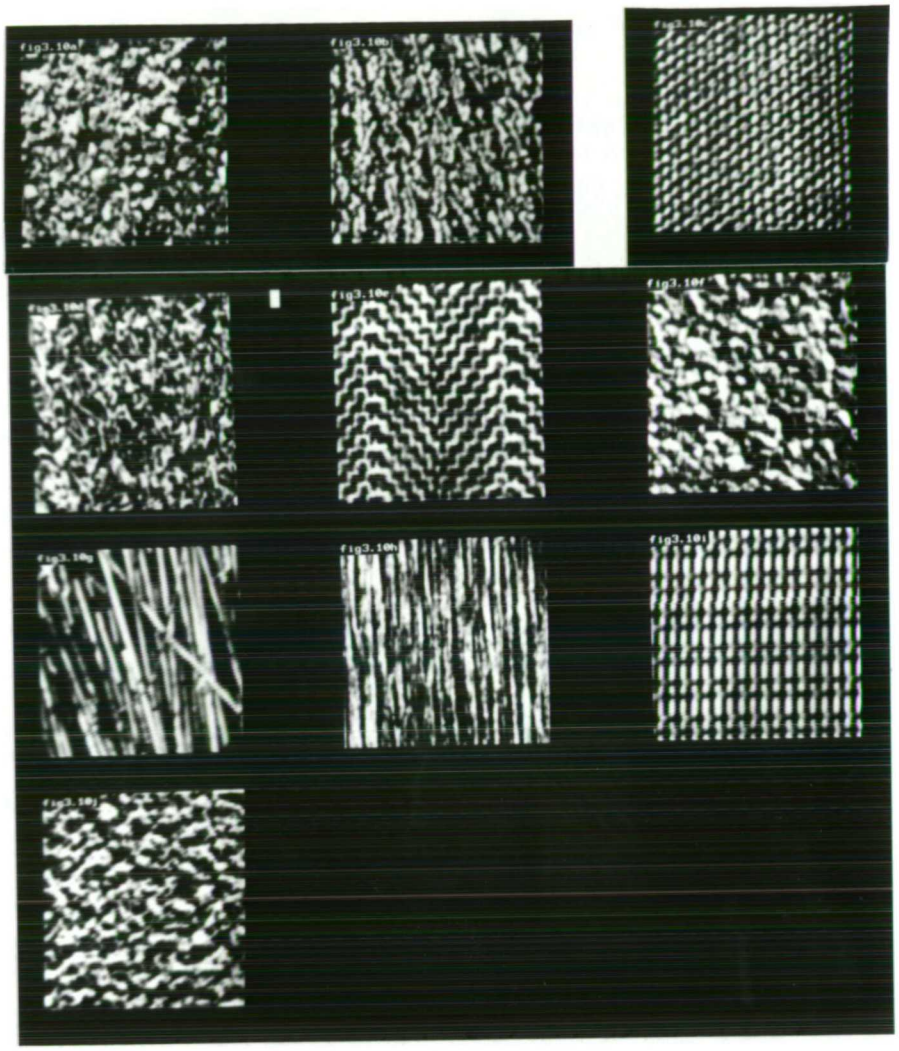


Figure 3.10. The 10 Texture Classes Used in LVM Texture Classification. (a) Beach Sand; (b) Calf; (c) Cotton Canvas; (d) Grass Lawn; (e) Herringbone; (f) Pressed Cork; (g) Oriental Straw; (h) Wood Grain; (i) d53 and (j) Hand-made Paper.

all TF_1s, TF_2s, TF_3s and TF_4s of LPT, LDCT and NLVM feature images are computed generating a corresponding feature vector with 80 (=4x5 + 4x8 + 4x7) components. The average value of 75 such texture feature vectors is assumed to be the feature vector for the

corresponding texture class. Some results of these calculations are tabulated in Tables 3.1 to 3.4. More details may be found in [19].

The unknown texture set is composed of 90 64x64 texture patterns 10% of which (i.e., 9 samples) are taken from each class. In the following, we list the input data used in the implementation of the LVM scheme for texture feature extraction and classification:

The 10 Texture Classes in The Database:

Beach Sand, Calf, Cotton Canvas, Grass Lawn, Herringbone, Pressed Cork, Oriented Straw, Wood Grain, d53 and Hand-made Paper as shown in Fig.3.10;

For LLVMs (LPT and LDCT):

Local Neighborhood $\eta_B(i,j)$ = a 3x3 square region;

Vector Formation Scheme Ψ = row concatenation;

For NLVMs:

Local Neighborhood $\eta_B(i,j)$ = a 5x5 square region (except for $h_{grl}[\bullet]$)

Vector Formation Scheme Ψ = row concatenation (except for $h_{lvt}[\bullet]$);

Local Neighborhood $\eta_B(i,j)$ for $h_{grl}[\bullet]$ = a 7x7 square region;

Vector Formation Scheme Ψ for $h_{lvt}[\bullet]$ = column concatenation;

Threshold Δ for $h_{grl}[\bullet]$ = 20;

3.5.2 The Classifiers

Although various pattern classifiers are available [24-25], two simple classifiers have been chosen in our experiments since our interests lie mainly in the investigation of the relative effectiveness of the defined LVM features in texture classification rather than the performance of different classifiers. The first classifier is the usual minimum Euclidean distance classifier which hereafter is called the *Minimum Aggregate Difference Square Classifier* denoted by **MADSC** and defined by

$$\begin{aligned} & d(k) = [\vec{F} - \vec{F}_k]^T \bullet [\vec{F} - \vec{F}_k] \\ & \textit{Classify texture } \vec{F} \textit{ into class } i \textit{ iff} \\ & d(i) = \min_{1 \le k \le 10} [\, d(k) \,] \end{aligned} \tag{3.17}$$

The second type of classifier used is referred to as *Minimum Aggregate Difference Absolute Classifier* denoted by **MADAC** which is sometimes also called the minimum city block distance classifier and is specified as follows

$$d(k) = \sum_{j=1}^{N} |f(j) - f_i(j)|$$
$$\textit{Classify texture } \vec{F} \textit{ into class i iff} \qquad (3.18)$$
$$d(i) = \min_{1 \le k \le 10} [\, d(k) \,]$$

where in (3.17) and (3.18)

- N = the total number of texture features;
- $f(j)$ = the j^{th} feature derived from the unknown texture concerned;
- $f_i(j)$ = the j^{th} feature of the known texture class i;
- $\vec{F} = [f(1)\, f(2) \cdots f(j) \cdots f(N)]^T$ is the feature vector extracted from the unknown texture sample; and
- $\vec{F}_i = [f_i(1)\, f_i(2) \cdots f_i(j) \cdots f_i(N)]^T$ is the feature vector of the known texture class i.

3.5.3 Texture Classification By LLVM Features

A. LPT Features

LPT texture features have been tested in texture classification both individually and collectively. Table 3.5 shows the classification accuracies by using LPT texture features. For example, at the third row and the second column, it reads 72.22% which is the classification accuracy by only using the variance TF_2 of the feature image $LPT.a_{11}(i,j)$. Since only one feature is used, that is N=1, MADSC and MADAC give identical results. It is encouraging to note that some LPT texture features are very powerful on their own in terms of their high classification accuracies. LPT TF_2s, and some LPT TF_4s are examples.

Table 3.6 presents the classification confusion matrix by using the 5 LPT TF_2s. In the table, the notation x/y means the number of texture class assignments by MADSC is x, and is y by MADAC, or the classification accuracy by MADSC is x and is y by MADAC as in "Accuracy = x/y ". In addition, all blank boxes are 0/0. Note, with 5 LPT TF_2s, only 3 samples

Table 3.5. The Classification Accuracies by Using Individual LPT Texture Features

moment / feature / image	MEAN TF1	VARIANCE TF2	SKEWNESS TF3	KURTOSIS TF4
LPT.a10	8.89%	61.11%	55.56%	60.00%
LPT.a01	10.00%	65.56%	31 11%	61.11%
LPT.a11	15.56%	72.22%	42.22%	58.89%
LPT.a20	8.89%	53.33%	42.22%	51.11%
LPT.a02	14.44%	61.11%	36.67%	55.56%

out of 90 are misclassified under MADSC, i.e., 3 Cotton Canvas samples are mislabeled with class Herringbone, while no classification error occurred under MADAC. More classification results by LPT texture features may be found in [19]. Based on these results, the following conclusions can be drawn:

1. In all cases, MADAC gives better results than MADSC in terms of classification accuracy;
2. LPT TF_1s are not much useful for texture classification as under both MADSC and MADAC, the classification accuracy is very low(less than 15%);
3. LPT TF_2s are the most powerful texture features compared with LPT TF_1s, TF_3s and TF_4s. Classification results by LPT TF_2s under both MADSC and MADAC are indeed very impressive;
4. LPT TF_3s and LPT TF_4s give comparable and also very good classification results.

It should also be pointed out that with the joint applications of LPT TF_2s and TF_3s or LPT TF_2s and TF_4s, the classification accuracies have not been seen to be improved and are even

Table 3.6. The Classification Confusion Matrix by Using The 5 LPT TF_2 Features

TEXTURE \ TEXT RE	Wood Grain	hand-made Paper	d53	Cotton Canvas	Straw	Beach Sand	Herring-bone	Calf	Grass Lawn	Pressed-cork
Wood Grain	9/9				0/0				0/0	
Paper		9/9								
d53			9/9							
Cotton Canvas		0/0		6/9			3/0			
Straw					9/9			0/0		
Beach Sand						9/9				
Herring-bone							9/9			
Calf								9/9		
Grass Lawn		0/0				0/0			9/9	
Pressed-cork										9/9

worse in some cases [19]. The reasons for this "inconsistency", we guess, are the following. Both MADSC and MADAC assume that all texture features are equally important and of the same order in magnitude. In addition, TF_2s, TF_3s and TF_4s defined (see (3.13) to (3.16)) are not mutually independent.

B. LDCT Features

For comparison, classification experiments by LDCT texture features have also been conducted. The same procedure has been followed as in LPT texture classification. Table 3.7 is the classification confusion matrix by using LDCT TF_2s. Under both MADSC and MADAC, the classification accuracy is 99.99%. The classification error is due to the misclassification of one Herringbone sample into class Beach Sand. More results by LDCT features are included in [19]. These results indicate that:

1. On the whole, LDCT features give only slightly better results than LPT features;
2. Conclusions and comments made on texture classification by LPT features are also applicable to LDCT features. This is not surprising since LPT and LDCT are both linear

Table 3.7. The Classification Confusion Matrix by Using The LDCT TF_2 Features

TEXTURE \ TEXT RE	Wood Grain	hand-made Paper	d53	Cotton Canvas	Straw	Beach Sand	Herring-bone	Calf	Grass Lawn	Pressed-cork
Wood Grain	9/9									
Paper		9/9								
d53			9/9							
Cotton Canvas				9/9						
Straw					9/9					
Beach Sand						9/9				
Herring-bone						1/1	8/8			
Calf								9/9		
Grass Lawn									9/9	
Pressed-cork										9/9

LVMs; and

3. Since the extraction of LDCT features are much more time-consuming than that of LPT features, LPT features may be preferred in practice.

3.5.4 Texture Classification by NLVM Features

To test the effectiveness of the defined NLVMs, classification experiments by NLVM features have also been performed. Table 3.8 shows the classification accuracy matrix by using individual NLVM texture features. *NLVM.grl* TF_1 seems to be most powerful among all 28 NLVM features as it gives the highest accuracy of 66.67%. The next most powerful one is *NLVM.led* TF_1 which gives 51.11%. Unlike LLVM texture features, NLVM TF_1s, TF_2s, TF_3s and TF_4s give comparable classification results with NLVM TF_1s being slightly better.

Table 3.9 summarizes the classification results by using three NLVM texture features: TF_1s of *NLVM.grl*, *NLVM.lht* and *NLVM.lvt*. The classification accuracies under both MADSC and MADAC are 75.56%. Concerned with the results shown in this table, two points are worthwhile making. First, the majority of the misclassifications occurred between those

Table 3.8. The Classification Accuracies by Using Individual NLVM Texture Features

Moment / Feature / Image	MEAN TF1	VARIANCE TF2	SKEWNESS TF3	KURTOSIS TF4
NLVM.VAR	45.56%	24.44%	37.78%	10.00%
NLVM.MAX	24.44%	43.33%	30.00%	36.67%
NLVM.MIN	32.22%	40.00%	27.78%	45.56%
NLVM.LED	51.11%	41.11%	15.56%	27.78%
NLVM.GRL	66.67%	47.78%	28.89%	34.44%
NLVM.LHT	40.00%	32.22%	22.22%	27.78%
NLVM.LVT	42.22%	30.00%	16.67%	16.67%

textures which are visually similar such as Hand-made Paper, Pressed Cork, Calf, Grass Lawn and Beach Sand. Secondly, misclassifications are unlikely linked to regular textures such as d53, Cotton Canvas and Herringbone. In fact, For these three texture classes, 100% accuracies are achieved.

In addition, classification experiments by using TF_1*s* and TF_2*s* of *NLVM.grl*, *NLVM.lht* and *NLVM.lvt* (6 features), and by using TF_1*s* and TF_2*s* of all NLVMs (14 features) have also been conducted using MADAC [19]. It is found that while the number of features used increases from 3, 6 to 14, the classification accuracy decreases from 75.56%, 61.11% to 26.67% [19]. This "inconsistency" has been observed in the classification experiments with LPT features. To overcome this weakness, the following classifier is employed:

$$d(k) \quad \sum_{j=1}^{N} \frac{|f(j)-f_k(j)|}{|v(j)-v_k(j)|}$$

$$\textit{Classify texture } \bar{F} \textit{ into class } i \textit{ iff} \qquad (3.19)$$

$$d(i) = \min_{1\le k\le 10}[d(k)]$$

where v is the TF_2 corresponding to feature f. Other symbols have the same meanings as in

Table 3.9. The Classification Results by Using The TF_1 Features of *NLVM.grl*, *NLVM.lht* and *NLVM.lvt*

TEXT RE / TEX URE	Wood Grain	hand-made Paper	d53	Cotton Canvas	Straw	Beach Sand	Herring-bone	Calf	Grass Lawn	Pressed-cork
Wood Grain	7/7							1/1		1/1
Paper		4/4						1/1	2/2	2/2
d53			9/9							
Cotton Canvas				9/9						
Straw					6/6	2/2				1/1
Beach Sand		1/1			2/2	5/5				1/1
Herring-bone							9/9			
Calf								7/7	2/2	
Grass Lawn		1/1						2/2	6/6	
Pressed-cork		1/1				2/2				6/6

(3.17) and (3.18). The classifier specified by (3.19) will be termed the *Modified MADAC* or **MMADAC** in short. The justification for MMADAC is that the feature images which have high absolute valued $TF_1 s$, $TF_3 s$ or $TF_4 s$ also have very large $TF_2 s$ (see Tables 3.1 to 3.4), Thus, with (3.19), the features $f(j)$ to be used are hoped to be normalized to a similar order in magnitude. With MMADAC, Table 3.10 shows the confusion matrix by 6 NLVM texture features ($TF_1 s$ and $TF_2 s$ of *NLVM.grl*, *NLVM.lht* and *NLVM.lvt*). The classification accuracy is 90.00% (versus 61.11% in MADAC as shown in Table 3.9). Further improvement in classification accuracy is achieved by using 14 NLVM texture features ($TF_1 s$ and $TF_2 s$ of all NLVMs) as illustrated in Table 3.11. The accuracy is as high as 95.56% (versus 26.67% in MADAC). Moreover, the majority of misclassifications in this case occur between the three similar textures: Grass Lawn, Calf and Pressed Cork.

Experimental results with NLVM features show that:

1. On the whole, NLVM texture features give slightly poorer classification results than

Table 3.10. The Classification Confusion Matrix by Using The TF_1 and TF_2 Features of *NLVM.grl*, *NLVM.lht* and *NLVM.lvt*

TEXTURE / TEXTURE	Wood Grain	hand-made Paper	d53	Cotton Canvas	Straw	Beach Sand	Herring-bone	Calf	Grass Lawn	Pressed-cork
Wood Grain	9									
Paper		9								
d53			9							
Cotton Canvas				9						
Straw		1			6	1			1	1
Beach Sand						7		1		1
Herring-bone				1			8			
Calf								9		
Grass Lawn						1			8	
Pressed-cork									1	8

LLVM features;

2. Misclassifications occur in a symmetrical way in that classification errors are most likely related to those visually "confusing" textures;
3. NLVM texture features are more appropriate for describing regular textures since misclassifications rarely occur between regular textures; and
4. Unlike LLVM features, TF_1s, TF_2s, TF_3s and TF_4s in NLVMs perform comparably.

3.6 Conclusions

Many approaches have been developed and used to extract some useful texture features to characterize a given texture field or/and to discriminate a large number of texture types. Since textural property is not point-oriented but local-region-oriented, local-operator-based approaches are often effective for texture analysis especially in the context of texture classification. By the introduction of the *Local Vector Mapping* scheme in this chapter, we offer a unification and generalization of some of the existing local-operation-based texture

Table 3.11. The Classification Confusion Matrix by Using The TF_1 and TF_2 Features of All The Defined NLVMs

TEXTURE / TEXTURE	Wood Grain	hand-made Paper	d53	Cotton Canvas	Straw	Beach Sand	Herring-bone	Calf	Grass Lawn	Pressed-cork
Wood Grain	9									
Paper		9								
d53			9							
Cotton Canvas				9						
Straw		1			8					
Beach Sand						9				
Herring-bone							9			
Calf								8	1	
Grass Lawn								1	8	
Pressed-cork									1	8

approaches. While the local operators previously used are mostly linear, the LVM scheme also includes non-linear local operations. By the formation and mapping operations of local vectors at each pixel, we conduct a series of image transformation operations, i.e., transform the original image into one or a few texture feature images or LVMed images. We find that the first-order statistics of the LVMed images are good texture descriptors for the original texture field.

Experimental evaluations for the effectiveness of some concrete examples of LVMs such as LPT, LDCT and some defined NLVMs have been performed in terms of texture classification. The results obtained enable us to draw the following conclusions on the LVM scheme presented in this chapter for texture feature extraction and classification: the LVM scheme is

- **Effective:** In both LLVM and NLVM texture classification, a classification accuracy higher than 90% is achieved by using only a few texture features. Especially in the case of LLVM, TF_2s alone give an accuracy as high as 95% or higher (even 100% in fact);

- **General:** Any types of textures (deterministic, structural or stochastic) may be classified. In addition, any approaches to texture based on local operations like local masking can be considered as special cases of the LVM scheme;
- **Compact:** The number of texture features used is small;
- **Flexible:** In addition to LPT, LDCT and the NLVMs defined in this chapter, many other useful operations such as the discrete sine transform, discrete Hadamard transform and the KLT can be included. Indeed, it may take a very long period to exhaust all possible LVMs for texture analysis. Moreover, we may improve the performance simply by increasing the dimension of local vectors (i.e., by defining large local neighborhoods), adopting better vector formation schemes or by using higher-order statistics of feature images;
- **Consistent with human visual perception:** Classification errors occur mainly between visually similar (confusing) textures. In addition, LVM texture features used provide good indications of important visual texture features like coarseness, regularity and directionality; and
- **Suitable for parallel (real-time) implementation:** As a LVM is a pointwise operation, the implementation of the LVM scheme can be greatly accelerated by means of parallel processing.

3.7 References

[1] **B.Julesz**, *Texton Gradient: The Texton Theory Revisited*, Biol. Cybern., vol.54, 1986, pp.245-251.

[2] **B.Julesz** and **B.Krose**, *Features and Spatial Filters*, Nature, vol.333, 1988, pp.302-303.

[3] **N.Lins**, *Refinement of Spectral Methods for Use in Texture Analysis*, SPIE, vol.596, 1985, pp.105-111.

[4] **M.Unser**, *Local Linear Transforms for Texture Measurements*, Signal Processing, vol.11, pp.61-79, 7/1986.

[5] **R.M.Haralick**, *Statistical and Structural Approaches to Texture*, Proc. IEEE, vol.67, pp.786-804, 1979.

[6] **K.I.Laws**, *Textured Image Segmentation*, PhD dissertation, University of Southern California, California, 1980.

[7] **M.Pietikainen** *et al.*, *Texture Classification Using Averages of Local Pattern Matches*, in Proc. 6th Int. Conf. on Pattern Recognition, Munich, 1982, pp.301-303.

[8] **F.Ade**, *Characterization of Texture by Eigenfilters*, Signal Processing, 5/1983, pp.451-457.

[9] **G.E.Lowitz**, *Mapping the Local Information Content of A Spatial Image*, Pattern Recognition, vol.17, pp.545-550, 1984.

[10] **R.Wang** *et al.*, *Texture Analysis Based on Local Standard deviation of Intensity*, Proc. CVPR"86: IEEE Computer Society Conf. on Comput. Vision and Pattern Recognition, FL, USA, 22-26 June 1986, pp.482-488.

[11] **O.R.Mitchell** and **S.G.Charlton**, *Image Segmentation Using a Local Extrema Texture Measures*, Pattern Recognition, vol.10, No.3, pp.205-2110, 1978.

[12] **P.T.Nguyen** and **J.Quinqueton**, *Space-Filling Curves and Texture Analysis*, Proc. of 6ICPR, Munich, 1982, pp.282-285.

[13] **T.N.Tan** and **A.G.Constantinides**, *Texture Classification by Local Surface Fitting*, Proc. of BPRA 4th Inter. Conf. on Pattern Recognition, Springer-Verlag, 1988, pp.48-58.

[14] **J.J.Koenderink** and **A.J.Van Doorn**, *New Type of Raster Scan Preserves the Topology of the Image*, Proc. of IEEE, vol.67, pp.1465-1466, 1979.

[15] **H.C.Kennedy**, *Selected Works of Giuseppe Peano*, pp.143-149, London: George Allen & Unwin Ltd., 1974.

[16] **T.Bially**, *Space-Filling Curves: Their Generation and Their Applications to Bandwidth Reduction*, IEEE Trans. on Information Theory, vol.IT-15, 1969, pp.658-664.

[17] **J.Quinqueton** and **M.Berthod**, *A Locally Adaptive Peano Scanning Algorithm*, IEEE Trans. on Patter Anal. Mach. Intell., vol.PAMI-3, pp.403-412, 1980.

[18] **L.Van Gool** *et al.*, *Survey: Texture Analysis Anno 1983*, Computer Vision, Graphics and Image Processing, vol.29, 1985, pp.336-357.

[19] **T.N.Tan**, *LVM Approach to Texture Analysis: Texture Classification*, TR-02-TNT, SPABSS/EE/ICST, May 1987.

[20] **A.Papoulis**, *Probability, Random Variables and Stochastic Process*, 2nd Edition, Mcgraw-Hill International Book Company, 1984.

[21] **M.Unser**, *On the Approximation of the Karhunen-Loeve Transform of Stationary Processes*, Signal Processing, vol.7, No.3, pp.231-249, 1984.

[22] **P.Brodatz**, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, 1966.

[23] **H.Tamura** *et. al.*, *Texture Features Corresponding to Visual Perception*, IEEE Trans. on Sys. Man and Cyber., vol.SMC-8, pp.460-472, 1978.

[24] **K.Fukunaga**, *Introduction to Statistical Pattern Recognition*, New York: Academic Press, 1972.

[25] **R.O.Duda** and **P.E.Hart**, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

4. LOCAL SURFACE FITTING FOR TEXTURE DESCRIPTION

In the last chapter, a general scheme for texture feature extraction, called the LVM (local vector mapping) scheme, is introduced where the description of a complete LVM necessitates local neighborhood specification $\eta_B(i,j)$, local vector formation $\Psi[\vec{V}(i,j)]$, and vector mapping $h_t[\vec{V}(i,j)]$. In this chapter, we shall investigate how some 2-D local operations useful for texture feature extraction may be directly defined on the chosen local neighborhood $\eta_B(i,j)$. The relationship between the defined 2-D local operations and the LVM scheme discussed in the preceding chapter will also be disclosed in this chapter.

4.1 Introduction

To define 2-D operations directly on $\eta_B(i,j)$, one may first have to be confronted with the question: what can these operations be? Recalling that different textures certainly exhibit distinct surface characteristics, digital image surface approximation and description can be expected to be a good vehicle for texture analysis. As a matter of fact, image surface approximation has been applied in a number of fields such as image interpolation, image restoration, image segmentation, and image feature extraction [1]. In most of these applications, surface approximation or fitting is performed globally, i.e., on the entire image surface. This type of global operation may be referred to as *global surface fitting*. As texture is a *local* image characteristic, and as global surface fitting is apparently incapable of exploiting local image features, we concur with **Eden** *et al.* [2] in the belief that a local continuous image representation or a local image surface fitting may be useful for texture feature extraction. Nevertheless, no formal procedure has been formulated in the open literature regarding the application of local image surface fitting in texture analysis, and thus our conjecture has not been affirmed. In the following sections, we shall present a general procedure for texture feature extraction based on local image surface fitting, and exemplify the procedure using polynomial approximation.

4.2 Local Surface Fitting (LSF)

Let k and l be the index variables for a grid lattice KxL with $k \in (1,2,...,K)$ and $l \in (1,2,...,L)$, and let a digital image be denoted by $F(k,l)$ with $(k,l) \in KxL$ and $F(k,l) \in (1,2,...,Q)$. A connected

sub-image or region with area A and boundary B will be represented by η, where A is defined to be the total number of pixels in η, i.e.,

$$A = \#\{(k,l): (k,l)\in\eta\}$$

where # denotes the number of elements in a set. If a unity quality is assigned to all pixels in η, the centroid (i,j) of the region η may be defined as follows

$$\begin{aligned} i &= Int\Big[\frac{\sum_{(k,l)\in\eta} k}{A}\Big] \\ j &= Int\Big[\frac{\sum_{(k,l)\in\eta} l}{A}\Big] \end{aligned} \tag{4.1}$$

where $Int[\bullet]$ denotes taking the nearest integer. In order to re-index the pixels in η, we define the following

$$\begin{aligned} &r = k-i; \\ &c = l-j; \\ &f(r,c) = F(r+i,c+j) = F(k,l) \end{aligned} \tag{4.2}$$

where $(i,j)\in KxL$ is the centroid of η, (k,l) the index pair for the original image $F(k,l)$, and (r,c) the relative coordinates of (k,l) in η w.r.t (i,j). A region specified by (4.1) and (4.2) will be called an *indexed local neighborhood*, and will be explicitly denoted by $\eta_B(i,j)$ as it is uniquely determined by its centroid and its boundary. From (4.2), it is clear that the centroid (i,j) has been actually defined as the origin (0,0) of the indexed local neighborhood $\eta_B(i,j)$. If the boundary B of $\eta_B(i,j)$ is symmetrical w.r.t. (i,j), similar expressions to (3.2) can be obtained:

$$\sum_{(r,c)} r=0 \quad \text{and} \quad \sum_{(r,c)} c=0; \qquad (r,c) \in \eta_B(i,j) \tag{4.3}$$

Now, if we let $z = f(r,c)$, then (r,c,z) may be visualized as a point in a R^3 space. In other words, $\eta_B(i,j)$ can be envisaged as defining a discrete surface described by $z = f(r,c)$ and bounded by B. Thus, any operations on $\eta_B(i,j)$ can be equivalently thought as the operations on the digital surface $z = f(r,c)$. However, being discrete is a severe obstacle for many mathematical manipulations. Therefore, we want to find a continuous surface which can approximate the digital surface $z = f(r,c)$ as closely as desired. Let such a continuous surface be symbolized by $z = \hat{f}(r,c,\vec{a}_T(i,j))$, which is uniquely characterized by a parameter vector $\vec{a}_T(i,j)$:

$$\vec{a}_T(i,j) = [a_1(i,j)\ a_2(i,j)\ \cdots\ a_T(i,j)]^T \tag{4.4}$$

For notational convenience, in the following discussions, we will omit the obvious dependence of the parameter vector $\vec{a}_T(i,j)$ on the centroid (i,j) of $\eta_B(i,j)$.

To find such a $z = \hat{f}(r,c,\vec{a}_T(i,j))$, the pivotal step is to estimate the parameter vector $\vec{a}_T$ from the given image data in $\eta_B(i,j)$. On doing so, let us define the approximation error to be

$$\epsilon(\vec{a}_T) = \sum_{(r,c)} \left| \hat{f}(r,c,\vec{a}_T) - f(r,c) \right|^2 \tag{4.5}$$

To obtain such an $\vec{a}_T$ that minimizes $\epsilon(\vec{a}_T)$, we take partial derivatives of (4.5) w.r.t. $\vec{a}_T$ and set them to zero giving

$$\frac{\partial \epsilon(\vec{a}_T)}{\partial \vec{a}_T} = 0 \tag{4.6a}$$

equivalently,

$$\frac{\partial \epsilon(\vec{a}_T)}{\partial a_t} = 0, \quad t=1,2,\ldots,T \tag{4.6b}$$

or

$$\sum_{(r,c)\in\eta_B(i,j)} [\hat{f}(r,c,\vec{a}_T) - f(r,c)] \bullet \frac{\partial \hat{f}}{\partial a_t} = 0, \quad t=1,2,\ldots,T \tag{4.6c}$$

In general, one has to solve T nonlinear equations specified by (4.6c) in order to get the estimates of the parameters a_t, $t = 1,2,\ldots,T$. Usually, this is computationally rather involved. Nevertheless, mathematically tractable solutions can be obtained if one carefully chooses the fitting function $\hat{f}(r,c,\vec{a}_T)$. One of such choices is given when $\hat{f}(r,c,\vec{a}_T)$ is specified as follows

$$\hat{f}(r,c,\vec{a}_T) = \vec{a}_T^T \bullet \vec{f}(r,c) \tag{4.7}$$

where $\vec{a}_T = [a_1\ a_2\ \cdots\ a_T]^T$ is the parameter or coefficient vector, $\vec{f}(r,c) = [f_1(r,c)\ f_2(r,c)\ \cdots\ f_T(r,c)]^T$ is a known function vector (i.e., basis functions), and the black dot $\bullet$ denotes inner product of vectors. By substituting (4.7) into (4.6c) and carrying out some simple mathematical operations, one gets

$$a_t \sum_{(r,c)} f_t^2(r,c) + \sum_{(r,c)} \sum_{s=1,s\neq t}^{T} a_s f_s(r,c) f_t(r,c) = \sum_{(r,c)} f(r,c) f_t(r,c); \quad (r,c) \in \eta_B(i,j) \tag{4.8}$$

From (4.8), the coefficient a_t can be easily obtained:

$$a_t = \frac{\sum_{(r,c)} f_t(r,c)\left[f(r,c) - \sum_{s=1,s\neq t}^{T} a_s f_s(r,c)\right]}{\sum_{(r,c)} f_t^2(r,c)}; \quad (r,c) \in \eta_B(i,j) \tag{4.9}$$

Regarding (4.9), several remarks are in order. The coefficient a_t calculated by (4.9) depends on the absolute position of $\eta_B(i,j)$ in the original image $F(k,l)$, in other words, a_t is a function of (i,j), which has not been explicitly expressed for the sake of simplicity. Secondly, in order to get the coefficients, one needs to solve T linear equations, implying that coefficient a_t is a linear combination of pixel values $f(r,c)$ in $\eta_B(i,j)$ as all $f_s(r,c)$, $s=1,2,...T$ are deterministic. Finally, if $f_s(r,c)$ satisfies

$$\sum_{(r,c)} f_s(r,c) \bullet f_t(r,c) = E_t \delta(t-s); \quad (r,c) \in \eta_B(i,j) \quad \text{and} \quad s,t = 1,2,...,T \tag{4.10}$$

i.e., if the basis functions are orthogonal to each other on $\eta_B(i,j)$, then (4.9) can be simplified as:

$$a_t = \frac{1}{E_t} \bullet \sum_{(r,c)} f_t(r,c) f(r,c), \quad (r,c) \in \eta_B(i,j) \quad \text{and} \quad t = 1,2,...,T \tag{4.11}$$

where E_t is some constant. The operations specified by (4.5), (4.6c) or (4.9) will be called **Local Surface Fitting** operations or simply **LSF** operations. It should be pointed out that from the practical point of view, (4.7) is often assumed.

Now, if the entire image $F(k,l)$ is scanned with $\eta_B(i,j)$ and the operation described above, a data field $a_t(i,j)$ for each coefficient is consequently produced. This equivalently says that after the LSF operation is performed at each pixel (i,j), the original image $F(k,l)$ will be "transformed" or "mapped" into T coefficient fields $a_t(i,j)$, $t=1,2,...,T$. Some thought will unveil that for a fixed (i,j), the textural information contained in the local neighborhood $\eta_B(i,j)$ is distributed upon the T coefficients a_t, $t = 1,2,...,T$ and the approximation error $\epsilon(\vec{a}_T)$. If the fitting is assumed to be good enough and hence the error $\epsilon(\vec{a}_T)$ is small enough, the information carried by $\epsilon(\vec{a}_T)$ is insignificant and thus the T coefficients suffice for characterizing $\eta_B(i,j)$. For

this reason, the T coefficient fields $a_t(i,j)$, $t = 1,2,...,T$ will be hereafter referred to as *texture feature images*.

Before leaving this section, it is worthwhile mentioning two points. Firstly, when $\eta_B(i,j)$ covers the entire image $F(k,l)$, LSF is then the usual image approximation/interpolation operation [3]; and secondly, when $\eta_B(i,j)$ is the original image $F(k,l)$, and (4.10) holds true, then LSF becomes some conventional orthogonal transforms such as DFT [4].

4.3 A Case Study: Local Polynomial Fitting

For a better understanding of the general procedure described in the preceding section, let us now concentrate on a concrete example, local surface fitting using polynomial basis functions. Note, in the following discussions, (4.7) is taken as granted.

For simplicity, let $\eta_B(i,j)$ be a 3x3 local region or square window with area $A = 9$. We define the known function vector $\vec{f}(r,c)$ as polynomial functions. For example, if T is chosen to be 6, we may define $\vec{f}(r,c) = [f_1(r,c)\ f_2(r,c)\ \cdots\ f_6(r,c)]^T$ as

$$\begin{aligned} f_1(r,c) &= 1 \\ f_2(r,c) &= r \\ f_3(r,c) &= c \\ f_4(r,c) &= rc \\ f_5(r,c) &= r^2 \\ f_6(r,c) &= c^2 \end{aligned} \tag{4.12}$$

Let the parameter vector be denoted by

$$\vec{a}_T = [a_{00}\ a_{10}\ a_{01}\ a_{11}\ a_{20}\ a_{02}]^T$$

then, (4.7) becomes

$$\begin{aligned} \hat{f}(r,c,\vec{a}_T) &= \vec{a}_T{}^T \bullet \vec{f}(r,c) \\ &= a_{00} + a_{10}r + a_{01}c + a_{11}rc + a_{20}r^2 + a_{02}c^2 \end{aligned} \tag{4.13}$$

which states that the discrete surface $\eta_B(i,j)$ is fitted by a second-order polynomial function. We call the LSF specified by (4.13) *local polynomial fitting (LPF)*. To compute the fitting coefficients, let us substitute (4.12) into (4.9), and solve the resultant 6 linear equations obtaining:

$$
\begin{aligned}
a_{00} &= \frac{1}{9}[-f(-1,-1)+2f(0,-1)-f(1,-1)-f(-1,1) \\
&\quad +2f(-1,0)+5f(0,0)+2f(1,0)-f(1,1)+2f(0,1)] \\
a_{10} &= \frac{1}{6}[-f(-1,-1)-f(1,-1)-f(0,-1)+f(0,1)+f(-1,1)+f(1,1)] \\
a_{01} &= \frac{1}{6}[f(-1,-1)+f(-1,0)+f(-1,1)-f(1,1)-f(1,0)-f(1,-1)]
\end{aligned}
\tag{4.14a}
$$

and

$$
\begin{aligned}
a_{11} &= \frac{1}{4}[-f(-1,-1)+f(1,-1)+f(-1,1)-f(1,1)] \\
a_{20} &= \frac{1}{6}[f(-1,-1)-2f(-1,0)+f(1,-1)+f(0,-1) \\
&\quad -2f(0,0)+f(0,1)+f(-1,1)-2f(1,0)+f(1,1)] \\
a_{02} &= \frac{1}{6}[f(-1,-1)+f(-1,0)+f(1,-1)-2f(0,-1) \\
&\quad -2f(0,0)-2f(0,1)+f(-1,1)+f(1,0)+f(1,1)]
\end{aligned}
\tag{4.14b}
$$

Recalling (4.2) and the dependence of $\vec{a}_T$ on (i,j), we write

$$
\begin{aligned}
a_{00}(i,j) &= \frac{1}{9}[-F(i-1,j-1)+2F(i,j-1)-F(i+1,j-1)-F(i-1,j+1) \\
&\quad +2F(i-1,j)+5F(i,j)+2F(i+1,j)-F(i+1,j+1)+2F(i,j+1)] \\
a_{10}(i,j) &= \frac{1}{6}[-F(i-1,j-1)-F(i+1,j-1)-F(i,j-1) \\
&\quad +F(i,j+1)+F(i-1,j+1)+F(i+1,j+1)] \\
a_{01}(i,j) &= \frac{1}{6}[F(i-1,j-1)+F(i-1,j)+F(i-1,j+1) \\
&\quad -F(i+1,j+1)-F(i+1,j)-F(i+1,j-1)]
\end{aligned}
\tag{4.15a}
$$

and

$$
\begin{aligned}
a_{11}(i,j) &= \frac{1}{4}[-F(i-1,j-1)+F(i+1,j-1)+F(i-1,j+1)-F(i+1,j+1)] \\
a_{20}(i,j) &= \frac{1}{6}[F(i-1,j-1)-2F(i-1,j)+F(i+1,j-1)+F(i,j-1) \\
&\quad -2F(i,j)+F(i,j+1)+F(i-1,j+1)-2F(i+1,j)+F(i+1,j+1)] \\
a_{02}(i,j) &= \frac{1}{6}[F(i-1,j-1)+F(i-1,j)+F(i+1,j-1)-2F(i,j-1) \\
&\quad -2F(i,j)-2F(i,j+1)+F(i-1,j+1)+F(i+1,j)+F(i+1,j+1)]
\end{aligned}
\tag{4.15b}
$$

(4.15a) and (4.15b) clearly define 6 image transforms or 6 FIR filters. After these operations are performed at all pixels, one obtains 6 texture feature images: $a_{00}(i,j)$, $a_{10}(i,j)$, $a_{01}(i,j)$,..., $a_{02}(i,j)$.

For large $\eta_B(i,j)$s and higher-order polynomials, the implementation of the general procedure is straightforward.

4.4 Texture Feature Extraction Based on LSF

It was claimed in the last chapter that many local operations can be interpreted under the general LVM scheme elaborated in that chapter. Since the LSF scheme introduced in this chapter is also of local operation type in nature, it can be expected to be expressible as a LVM problem. To confirm this, let us focus our attention on the local polynomial fitting specified by (4.13) and (4.15). Let the 9 pixels in $\eta_B(i,j)$ be represented by a 1-D vector $\vec{V}(i,j)$ as:

$$\begin{aligned}\vec{V}(i,j) = [&F(i-1,j-1)\ F(i-1,j)\ F(i-1,j+1)\ F(i,j-1)\\ &F(i,j)\ F(i,j+1)\ F(i+1,j-1)\ F(i+1,j)\ F(i+1,j+1)]^T\end{aligned} \tag{4.16}$$

If $\vec{V}(i,j)$ is considered as a local vector associated with pixel (i,j), then it can be written in the terminology of the LVM scheme as

$$\vec{V}(i,j) = \Psi[\eta_B(i,j)] \tag{4.17}$$

where Ψ = *row concatenation*. Further let us define

$$\vec{a}_T(i,j) = [a_{00}(i,j)\ a_{10}(i,j)\ a_{01}(i,j)\ a_{11}(i,j)\ a_{20}(i,j)\ a_{02}(i,j)]^T \tag{4.18}$$

then (4.15) can be compactly written as

$$\vec{a}_T(i,j) = [h_1^T\ x\vec{V}(i,j)\ h_2^T\ x\vec{V}(i,j)\ \cdots\ h_6^T\ x\vec{V}(i,j)]^T \tag{4.19}$$

where h_t, $t = 1,2,\ldots,6$ is given in (3.11). Now, it is clear that the LSF operation specified by (4.13) and (4.15) can be equivalently described as the following LVM problem:

$$\begin{aligned}&\eta_B(i,j) = a\ 3x3\ window;\\ &\Psi - row\ concatenation;\\ &g_t(i,j) = h_t^T\ x\vec{V}(i,j);\quad t - 1,2,\ldots,6\end{aligned} \tag{4.20}$$

where $g_t(i,j)$ is the t^{th} element of $\vec{a}_T(i,j)$ defined in (4.18). This LVM problem was referred to as *local polynomial mapping* in the last chapter. The implications from the above description equivalence are two fold. First, a LSF operation can be conveniently interpreted as a LVM problem; and secondly, some LVM problems can have more intuitive interpretations (such as surface fitting) on the 2-D domain.

Since LSF operations can be formulated as some LVM operations, texture feature extraction based on LSF can be similarly handled as in the LVM scheme. For instance, the

moments from the coefficient images or texture feature images $a_t(i,j)$, $t = 1,2,...,T$ may be computed and defined as texture measures. To avoid duplication, details will not be presented.

4.5 Conclusions

In this chapter, we attempted to define useful operations for texture feature extraction directly on 2-D local neighborhoods. The operation is based on fitting a local digital image surface by continuous functions, and is called local surface fitting (LSF). The output of a LSF operation is a set of coefficient images from which texture measures can be extracted. After the description of the general LSF scheme, a concrete example (local polynomial fitting) is given. Since it is demonstrated that a LSF operation can be described as a LVM problem, details on texture feature extraction based on LSF have been omitted.

4.6 References

[1] **M.Kocher** and **R.Leonardi**, *Adaptive Region Growing Technique Using Polynomial Function for Image Approximation*, Signal Processing, vol.11, 1986, pp.47-60.

[2] **M.Eden** *et al.*, *Polynomial Representation of Picture*, Signal Processing, vol.10, 1986, pp.385-393.

[3] **P.Lancaster** and **K.Salkauskas**, *Curve and Surface Fitting: an introduction*, Academic Press, Orlando, FL, 1986.

[4] **A.K.Jain**, *A Sinusoidal Family of Unitary Transforms*, IEEE Trans. Pattern Anal. Mach. Intell., vol.PAMI-1, 1979, pp.356-365.

5. TEXTURE FEATURE EXTRACTION BASED ON PRIMITIVE ANALYSIS

5.1 Introduction

It is commonly accepted that statistical texture approaches are in general effective and efficient for microtextures and stochastic textures whereas structural approaches are usually well suited for macrotextures and structural textures [1,2]. As the approaches described in Chapters 3 and 4 are all based on some kinds of local operations, and make no efforts to identify texture primitives, they can be classified as statistical approaches. As such, they are not particularly effective and may be inefficient for macrotextures and structural textures. It is the concern of the current chapter to present an easily implementable algorithm for macrotexture and structural texture analysis, and to verify the common assertion that structural approaches are not effective for stochastic textures.

Considering the high complexity of most of the structural approaches, the approach to be presented belongs to weak texture analysis. Thus particular emphasis is placed on primitive extraction for which three different methods are described. The first is based on simple thresholding preceded by Gaussian low-pass filtering. The threshold is determined empirically; The second differs from the first in that the filtering operation is substituted by an image segmentation operation. The third method also applies image segmentation before intensity thresholding but the thresholds are determined automatically based on a moment-preserving criterion. The overall algorithm for texture feature extraction proposed in this chapter is composed of primitive extraction, primitive refinement, primitive attribute calculation and texture feature computation. The effectiveness of the extracted features for texture description is illustrated via texture classification. The relative performances of the three methods for primitive extraction are also compared.

In the next section, the three methods for primitive extraction will be described in detail. Primitive refinement will also be covered in this section. Section 5.3 deals with primitive description and texture feature computation whereas Section 5.4 describes the procedure for texture classification. Classification results are also given in this section. Finally, Section 5.5 concludes this chapter.

5.2 Primitive Extraction and Primitive Refinement

5.2.1 Primitive Extraction

To facilitate the presentation of the methods for primitive extraction, the concept of texture primitive should be first understood. In this chapter, a primitive is defined as a simply (rather than multiply) connected image region of homogeneous intensities. In particular, two types of primitives namely *dark primitives* and *bright primitives* will be defined and used here whereas only one type of primitive (either dark or bright) has been defined and applied in previous work by others [3,4]. A dark primitive is defined as an image region, the constituent pixels of which have intensities below a given threshold, while a bright primitive is an image region, the pixels of which have intensities above a pre-specified threshold. In the following, we shall present the three methods for primitive extraction.

METHOD A: GLPF & FPT

This technique is basically the one proposed in [4] but is extended here in the sense that both dark and bright primitives are defined, extracted and applied. It is included here also as a benchmark for comparison. In [4], the authors presented three different techniques for primitive extraction namely 25% percentile thresholding (FPT: Fixed Percentile Thresholding), iterative histogram modification and Superslice. They subsequently reported that the simplest of these, the FPT, yields primitives that are quite effective in texture discrimination. This motivates the inclusion of the FPT operation here. To understand the FPT, let the input image be $f(i,j)$ of size NxN, and let T_d and T_b represent the thresholds associated with dark and bright primitives respectively. Furthermore, let N_d be the number of pixels having intensity below T_d, and N_b the number of pixels having intensity above T_b. Then, in the FPT, T_d and T_b are chosen to be such that the following holds:

$$\begin{aligned} \frac{N_d}{NxN} x100\% &= 25\% \\ \frac{N_b}{NxN} x100\% &= 25\% \end{aligned} \tag{5.1}$$

In order to reduce noise effects and to extract connected primitives, the FPT is preceded by low-pass filtering. We choose a Gaussian low-pass filter (GLPF) for its optimization of the *uncertainty principle* [5].

The weakness of this method is easy to see. The extent of low-pass filtering is clearly image-dependent. Excessive filtering will cause primitives to merge with each other (especially when primitives are of small size) whereas inadequate filtering may result in noisy and disconnected primitives.

METHOD B: RSST & FPT

GLPF & FPT may be preferred for its simplicity as it consists of low-pass filtering and intensity thresholding. However, as we discussed in the preceding paragraph, it may fail if either or both of the following is true:

1. The input image is very noisy;
2. The underlying primitives are very small in size.

In order to overcome these problems, a segmentation technique would be required which is robust in the presence of noise and capable of extracting connected image regions of any size (high segmentation resolution). A segmentation technique called the *Recursive Shortest Spanning Tree* (RSST) segmentation technique recently developed by **Morris** *et al* [6] has proven to satisfy all these requirements. The technique can segment an image into any number of regions. Since this segmentation technique is insensitive to noise, GLPF is not required. Thus, RSST segmentation is directly followed by the FPT described in (5.1). Although the number of segmentation regions may be image-dependent, e.g., images containing small primitives should be partitioned into more regions than images having large primitives, we found experimentally that a number of 200 segmentation regions is in general pertinent.

METHOD C: RSST & MIT

In both method A and B, the two thresholds T_d and T_b related to dark and bright primitives are determined empirically according to equation (5.1). In this method, we apply a technique which automatically calculates the thresholding percentiles and the corresponding thresholds. The technique is called *Moment Invariant Thresholding* (MIT) developed in [7]. After the RSST segmentation, the segmented image is then gray-scaled or thresholded by means of the MIT technique to three gray levels z_1, z_2 and z_3 with $z_1 < z_2 < z_3$. These three "representative" gray levels and the corresponding two thresholds are determined in such a way that the image

moments are kept constant before and after gray-scaling. We then define image regions with level z_1 as dark primitives and image regions with z_3 as bright primitives. For details on the MIT technique, we refer to [7].

This completes the presentation of the three methods for primitive extraction.

5.2.2 Primitive Refinement

Some of the primitives obtained by means of Methods A, B or C are found to be not isolated or noisy. The resultant primitive maps also contain isolated pixels or short line segments not belonging to any specific primitives. Meanwhile, border regions are only part of the underlying true primitives and their inclusion will certainly falsify primitive attribute distributions. We thus proceed with primitive refinement after primitive extraction. This operation involves isolated pixel deletion, short line segment deletion, singe-pixel-connection deletion, extrema primitive (like very large and very small primitives) deletion and border primitive deletion. Primitives which survive after primitive refinement may be called *valid* primitives. For each valid primitive, we then perform simple shape analysis as discussed in the next section.

For visual illustration, Fig.5.1 is given. The three images in the first row represent the original texture field (D55) taken from [8], its GLPFed version, and its segmented version respectively. The second row shows the dark primitive maps (after primitive refinement) computed by Method A, B and C respectively. Similarly, the third row presents the three bright primitive maps.

5.3 Texture Feature Extraction

Once we have extracted the texture primitives, the next step is to measure the properties or attributes which may characterize them (primitive characterization). We have chosen a set of five shape attributes (area, perimeter, compactness, eccentricity and orientation) together with primitive color and primitive contrast.

The area of a primitive is defined simply as the total number of constituent pixels in the primitive. The area attributes are directly related to the coarseness and fineness of a texture field as coarse textures will have very large primitives thus large areas, while fine textures will

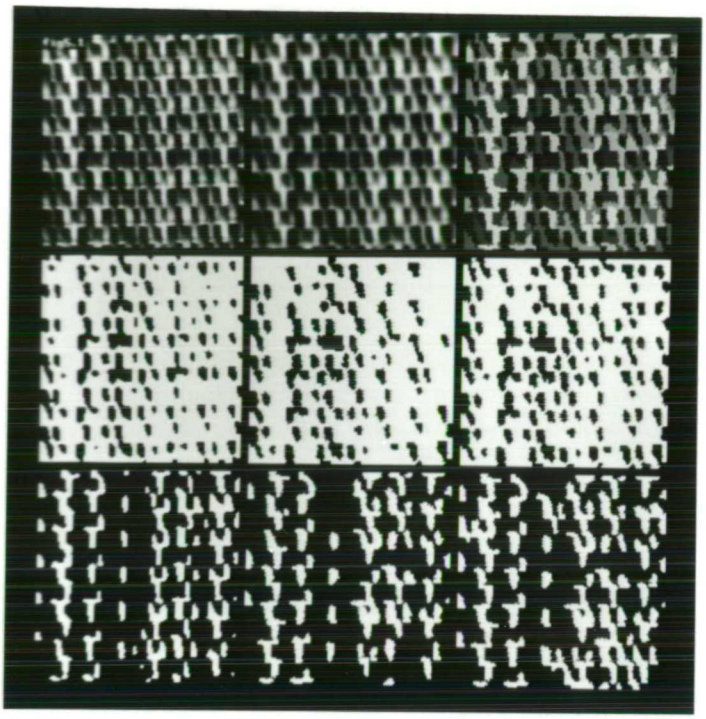


Figure 5.1. Illustration of Primitive Extraction and Refinement.

contain small primitives thus small areas. The perimeter of a primitive is the total number of boundary pixels in the primitive whereas the compactness is defined to be the ratio of the square of the perimeter to the area. To facilitate the definition of eccentricity and orientation, let us denote the centroid of a primitive $p(i,j)$ by (i_0,j_0). For all pixels in $p(i,j)$, let us calculate the three moments of $p(i,j)$ defined by

$$\begin{cases} M_{ii} = \sum(j-j_0)^2 \\ M_{jj} = \sum(i-i_0)^2 \\ M_{ij} = \sum(i-i_0)(j-j_0) \end{cases} \tag{5.2}$$

Then the orientation of a primitive $p(i,j)$ is defined as the angle that the major axis of the inertia of the primitive makes with the vertical direction given by

$$\phi(p) = \frac{1}{2}\tan^{-1}\left(\frac{2M_{ij}}{M_{jj}-M_{ii}}\right) \tag{5.3}$$

and the eccentricity is defined as the ratio of the major to the minor axis specified by

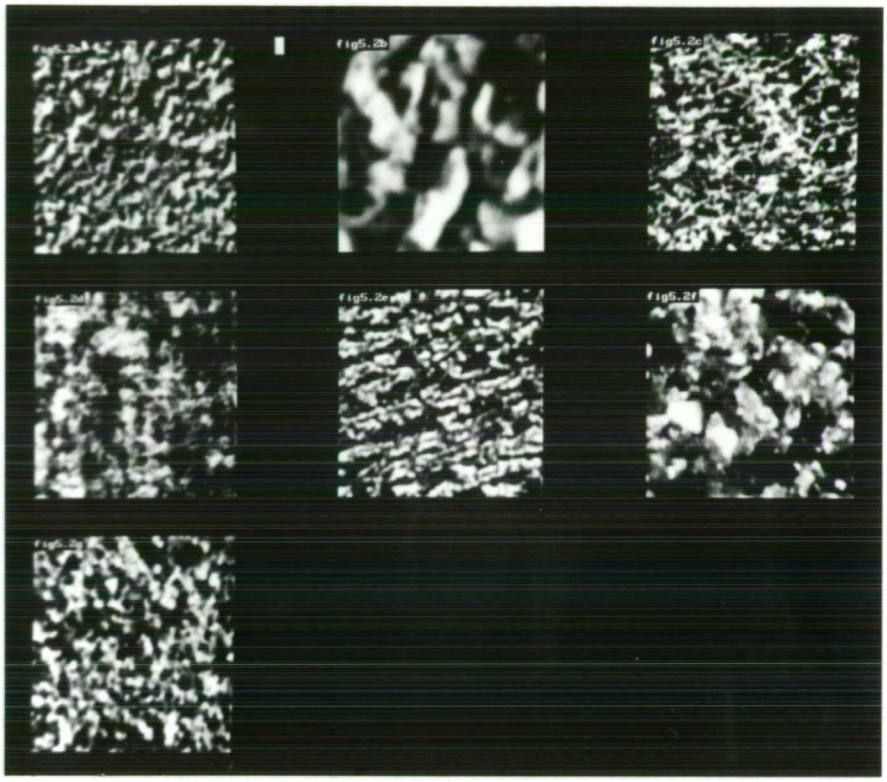
$$E(p) = \left(\frac{M_{ii}\cos^2\phi(p)+M_{jj}\sin^2\phi(p)-M_{ij}sin2\phi(p)}{M_{ii}\sin^2\phi(p)+M_{jj}\cos^2\phi(p)+M_{ij}sin2\phi(p)}\right)^{1/4} \tag{5.4}$$

Finally, the color of a primitive is simply the average intensity of the pixels in the primitive, while the contrast is defined as the intensity variance of the primitive.

The attribute values of a given primitive are assumed to describe that particular primitive whereas the distributions of the attribute values for all primitives serve to characterize the input texture field. This is clearly a statistical description. For the purpose of texture feature extraction, we may use primitive attribute distributions as texture features. Although higher-order distributions of primitive attributes (like second-order distributions) can be used, only first-order distributions are used here as higher-order distributions do not seem to improve texture discrimination significantly [4]. The histograms of primitive attribute values are good approximations of the underlying first-order distributions. Instead of directly using attribute histograms as texture features, we further approximate each histogram by its mean and variance. In other words, associated with each primitive attribute, we define two texture features: the mean and the variance (or the standard deviation) of the attribute distribution. Since we have two types of primitives (dark and bright), and each primitive is described by 7 attributes as discussed above, we defined, for each texture sample, $2x7x2 = 28$ texture features. These features will be tested in the context of texture classification in the next section.

5.4 Texture Classification and Experimental Results

To test and compare the effectiveness of the three methods discussed for texture primitive and feature extraction, the extracted texture features are applied in texture classification. 14 texture classes have been selected from the texture album of **Brodatz** [8]. They are Pressed Cork (D4), Expanded Mica (D5), Grass Lawn (D9), Woolen Cloth (D19), Pressed Calf Leather (D24), Beach Sand (D28 and D29), Oriental Straw Cloth (D53), Straw Matting (D55), Handmade Paper (D57), Oriental Rattan (D65), Wood Grain (D68), Raffia (D84) and Pigskin (D92). Samples from these classes are displayed in Fig.5.2. For classifier



(To be continued on the next page)

training, 16 128x128 images (without overlap) are derived from a 512x512 image for each texture class. For classification, 10 samples from each class are used giving a total number of 140 unknown samples to be classified. These unknown samples have not been used in classifier training but have some overlaps with those used in the training. To prevent texture discrimination on the basis of first-order statistics such as average intensity, we preprocess all samples by means of histogram equalization. The classifier classifies an unknown texture sample with feature vector $\hat{F}$ into texture class i *iff* the following distance function is a minimum at i:

$$d(\hat{F},i) = \sum_{k=1}^{N} \frac{[f_k - f_{ik}]^2}{\sigma_{ik}^2} \qquad (5.5)$$

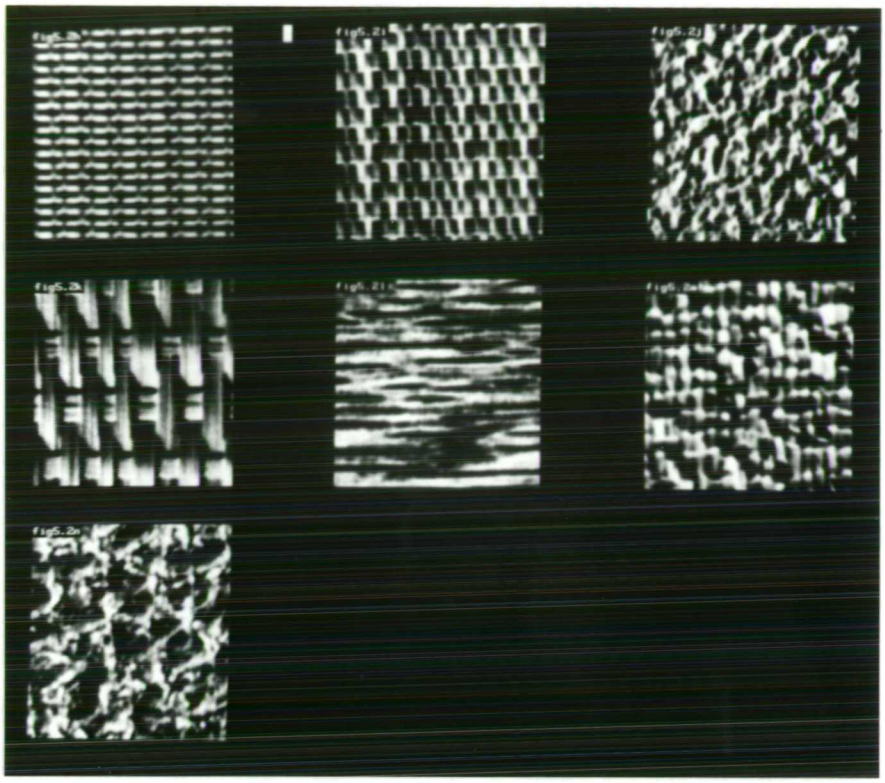


Figure 5.2. The 14 Texture Classes Used in Texture Classification Experiments. (a) Pressed Cork (D4); (b) Expanded Mica (D5); (c) Grass Lawn (D9); (d) Woolen Cloth (D19); (e) Pressed Calf Leather (D24); (f) Beach Sand (D28); (g) Beach Sand (D29); (h) Oriental Straw Cloth (D53); (i) Straw Matting (D55); (j) Handmade Paper (D57); (k) Oriental Rattan (D65); (l) Wood Grain (D68); (m) Raffia (D84) and (n) Pigskin (D92).

where $\hat{F} = [f_1 f_2 \cdots f_N]^T$, and f_{ik} and σ_{ik} are the mean and the standard deviation of the k^{th} feature of class i respectively. Similar classifiers to (5.5) have also been employed in [9] and [10].

For each method of primitive extraction, a set of three classification experiments are conducted by using all the 28 features, 14 dark primitive features and 14 bright primitive

features respectively. The classification results are given in the form of *confusion matrix*. The entire set of results are available in [11]. Only part of the experimental results can be presented here because of space limitation. Table 5.1 shows the confusion matrix obtained by using all 28 features from Method A (GLPF & FPT). Perfect classification is achieved for texture classes D5, D19, D53, D55, D65 and D68 whereas classification accuracies are less than 50% for classes D4 and D84. Table 5.2 presents the classification result of Method A using only the 14 dark primitive features. A brief comparison between Tables 5.1 and 5.2 makes it clear that the joint application of both dark and bright primitives improves the classification accuracy dramatically. Tables 5.3 and 5.4 are the classification results of Methods B and C respectively by using all 28 features. Except for a 100% misclassification for class D57, Method B performs better for other classes than either Method A or Method C. Both tables show that very high classification accuracies ($\geq 90\%$) can be achieved for texture classes D5, D28, D53, D55, D65 and D68. A brief glance at Fig.5.2 tells that these textures may be classified as either macrotextures or structural textures. On the contrary, the classification accuracies given in the two tables are very low for texture classes D4, D24, D29, D57 and D84, all of which may be considered as stochastic textures.

From these results (and also those given in [11]), the following conclusions can be drawn:

1. The joint application of both dark and bright primitives gives much better performances than either dark or bright primitives alone. Thus the inclusion of only one type of primitive as in [3,4] is insufficient.
2. Methods A and C give comparable results, while Method B produces the best.
3. The texture features defined are particularly effective for discriminating macro- and structural textures like D5, D28, D53, D55 and D65, while ineffective for micro- and stochastic textures like D4, D24 and D57, thus having affirmed the assertion cited at the beginning of this chapter.

5.5 Conclusions

In this chapter, we have discussed texture feature extraction based on texture primitive analysis. Three simple techniques have been presented for primitive extraction. Texture features are defined as means and standard deviations of the first-order distributions of

Table 5.1

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	4		5											1	40%
D5		10													100%
D9			8				1					1			80%
D19				10											100%
D24			2		5		1					2			50%
D28						9					1				90%
D29			2				6					2			60%
D53								10							100%
D55									10						100%
D57	1		1				2			6					60%
D65											10				100%
D68												10			100%
D84			1	1		1	4					1	2		20%
D92			2	1							2			5	50%

Table 5.2

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	3		6											1	30%
D5		4				1					5				40%
D9			9	1											90%
D19				10											100%
D24			1		1		2				5	1			10%
D28						6					4				60%
D29			1				3				4	1		1	30%
D53								5			5				50%
D55				3					7						70%
D57	1				1		2			3	3				30%
D65											10				100%
D68											2	8			80%
D84			1				3				5		1		10%
D92				1							4			5	50%

Table 5.3

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	8		1		1										80%
D5		10													100%
D9	1		7		1		1								70%
D19		1		9											90%
D24					6		3					1			60%
D28				1		9									90%
D29			2	2			6								60%
D53								10							100%
D55			1						9						90%
D57			6				2			0			2		0%
D65											10				100%
D68												10			100%
D84			1				2						7		70%
D92			1	2										7	70%

Table 5.4

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	3		4		1		1							1	30%
D5		10													100%
D9			8	1			1								80%
D19				8		1								1	80%
D24			1		2		4					1	2		20%
D28						10									100%
D29			3				6						1		60%
D53								10							100%
D55									10						100%
D57			2	1			2			4			1		40%
D65											10				100%
D68												10			100%
D84			1				4						5		50%
D92				1										9	90%

Table 5.1. Texture Classification Results by Using 28 Dark and Bright Primitive Features from Method A.

Table 5.2. Texture Classification Results by Using 14 Dark Primitive Features from Method A.

Table 5.3. Texture Classification Results by Using 28 Dark and Bright Primitive Features from Method B.

Table 5.4. Texture Classification Results by Using 28 Dark and Bright Primitive Features from Method C.

primitive attributes such as area, perimeter, compactness, orientation, eccentricity, color and

contrast. The effectiveness of the extracted texture features is illustrated in the context of texture classification. Experimental results show that the extracted texture features are particularly effective for macro- and structural texture discrimination but not so for micro- and stochastic textures.

5.6 References

[1] **R.M.Haralick**, *Statistical and Structural Approaches to Textures*, Proc. IEEE, vol.67, 1979, pp.786-804.

[2] **L.V.Gool** *et al.*, *Survey: Texture Analysis Anno 1983*, Comput. Vision, Graphics and Image Proc., vol. 29, 1985, pp.336-357.

[3] **T.H.Hong** *et al*, *Texture Primitive Extraction Using An Edge-Based Approach*, IEEE Trans. Syst. Man Cybern., vol.SMC-10, 1980, pp.659-675.

[4] **S.Wang** *et al*, *Relative Effectiveness of Selected Texture Primitive Statistics for Texture Discrimination*, IEEE Trans. on Syst. Man Cybern., vol.SMC-11, 1981, pp.360-370.

[5] **R.N.Bracewell**, *The Fourier Transform and Its Applications*, McGraw-Hill, New York, 1978.

[6] **O.J.Morris** *et al*, *Graph Theory for Image Analysis: An Approach Based on Shortest Spanning Tree*, Proc of IEE, part F, vol.133, 1986, pp.146-152.

[7] **W.H.Tsai**, *Moment-Preserving Thresholding: A New Approach*, Comput. Vision, Graphics and Image Proc., vol.29, 1985, pp.377-393.

[8] **P.Brodatz**, *Textures: A Photographic Album for Artists and Designers*, New York: Diver, 1966.

[9] **R.Chellappa** and **S.Chatterjee**, *Classification of Textures Using Gaussian Markov Random Fields*, IEEE Trans. on Acoustic, Speech and Signal Processing, vol. ASSP-33, 8/85, pp.959-963.

[10] **R.L.Kashyap** and **A.Khotanzad**, *A Model-Based Method for Rotation Invariant Texture Classification* , IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, Jul. 1986, pp.472-482.

[11] **T.N.Tan**, *Texture Primitive and Feature Extraction for Texture Classification*, TR-07-TNT, SPABSS/EE/ICST, August 1988.

6. GABOR FILTERS IN TEXTURE ANALYSIS I: TEXTURE FEATURE EXTRACTION AND CLASSIFICATION

6.1 Introduction

So far, we have developed three novel algorithms for texture feature extraction and classification, namely the local vector mapping scheme elaborated in Chapter 3, the local surface fitting procedure presented in Chapter 4, and the weak texture analysis approach described in Chapter 5. As the first two can be considered to be statistical in nature and thus may be ineffective for structural textures, and as the third is clearly a structural approach and has been shown not to be effective for stochastic textures, the general applicability of these algorithms is in doubt. However, in reality, perceivable textures can range from highly structural (deterministic) to highly stochastic (random), and such an algorithm is often desirable that is equally applicable under all circumstances. This chapter is intended to develop an algorithm of this kind.

Before searching for such an algorithm, the following question can be asked: Does there exist a system in nature which is efficient as well as effective for texture analysis under a rather wide spectrum of circumstances? The answer is yes, and such a system has been known to be the **human visual system (HVS)**. Thus, the algorithm we are seeking for should be such that mimics the HVS in texture analysis. Apparently, before the development of such an algorithm is possible, the principles of the HVS in pictorial information processing and texture analysis have to be first understood. Although there is still much left to know about the HVS, a vast amount of evidence has affirmed the hypothesis that the processing of afferent pictorial information in the HVS (the visual cortex in particular) involves a set of parallel and quasi-independent mechanisms or *channels* [1]. If each channel is envisaged to perform a two-dimensional or spatial filtering operation in a broader sense, the underlying visual model may be called a **Multichannel Spatial Filtering Model (MSFM)**.

The establishment of the MSFM in the human vision community has stimulated a great deal of recent interest in applying the model to texture analysis in the image analysis community as we surveyed in Section 2.4.7. The survey also makes it clear that the MSFM based approaches to texture feature extraction and discrimination previously reported by others either define individual channels (e.g., transfer functions of the underlying spatial filters) empirically

without any physiological and psychophysical supports [2,3] or, as **Wright** and **Jernigan** commented on MSFM approaches including their own [4,5], "...implicitly assumed that the filter responses are polar separable so that [texture] features can be examined along the r [radial frequency] and θ [spatial orientation] dimensions independently. No direct support for this assumption is available." [p.520, 5]. However, convincing evidence in favour of the non-separability of filter responses has been offered by **Gaugman** [6]. He concluded upon empirical measurements that visual threshold elevation surfaces or the transfer functions of the envisaged channel filters are highly polar nonseparable, and can not be described as the product of a spatial frequency tuning curve and an orientation tuning curve. This non-separability hypothesis is further enforced by the fact that the profiles of the receptive fields of most of the cortical neurons (especially simple cortical cells) can be well described by Gabor functions [7] which are polar separable neither in the spatial domain nor in the spatial frequency domain.

In this chapter, we present a MSFM approach based on a new cortical channel model described in [8]. Each channel employs two Gabor filters of quadrature phase corresponding to a pair of adjacent simple cells found in the visual cortex [7]. Channel design, i.e., the determination of the number of channels and other channel parameters, is rooted on image coding and energy processing characteristics of the HVS. The means and the standard deviations of the channel outputs are defined to be texture features. These features, after being interpreted in terms of visual texture features, are applied in texture classification. For comparison and judgement of effectiveness, the proposed algorithm for texture feature extraction is subsequently compared with the well-known standard texture description technique the *Gray Level Dependence Matrix (GLDM)* technique.

The rest of this chapter is formatted as follows. The following section starts with a brief description of Gabor functions/filters followed by a summary of the principles of the visual cortex in pictorial information processing. The cortical channel model is also formulated in this section. Section 6.3 then presents a set of experiments investigating the image coding and energy processing characteristics of the HVS and subsequently facilitating the channel and filtering pattern design. The algorithm for texture feature extraction, together with texture classification results, is presented in Section 6.4. Section 6.5 is devoted to the description of feature extraction by GLDM. Comparison between the proposed algorithm and the standard technique is summarized in Section 6.6, with Section 6.7 concluding the chapter.

6.2 Gabor Filters and The Visual Cortex

In this section, we shall give the descriptions of Gabor filters on both spatial and frequency domains. Their peculiar feature of achieving simultaneous localizations on both domains are briefly discussed. The connection between Gabor filters and the receptive fields of neurons in the visual cortex is exploited. The principles of the visual cortex on processing pictorial information will be summarized in order to facilitate the formulation of the cortical channel model.

6.2.1 Mathematical Descriptions of Gabor Filters

Gabor filters are originated from Gabor's famous Theory of Communication [9]. A 2-D Gabor filter or function has the following general form

$$h(x,y) = g(x,y) \bullet e^{-2\pi j(u_0 x + v_0 y)} \tag{6.1}$$

where $g(x,y)$ is a 2-D Gaussian defined as

$$g(x,y) = e^{-\frac{1}{2}[(\frac{x}{\sigma_x})^2 + (\frac{y}{\sigma_y})^2]} \tag{6.2}$$

The Gabor filter $h(x,y)$ specified by (6.1) is a spatial complex sinusoidal grating modulated by the 2-D tapering Gaussian defined in (6.2). The spatial constants σ_x and σ_y determine the tapering speed and thus the scale of the Gaussian along the x and y axes. The ratio $\lambda = \frac{\sigma_x}{\sigma_y}$ is usually termed the *aspect ratio* of the Gaussian and can be used to impart the degree of the orientation sensitivity of the overall Gabor filter as illustrated in Fig.6.1. The overall modulation frequency ω_0 of the filter $h(x,y)$ is $\omega_0 = (u_0^2 + v_0^2)^{1/2}$. The axis of the modulation is oriented at an angle of $\theta = \tan^{-1}(\frac{v_0}{u_0})$ from the horizontal axis. If ω_0 and θ_0, instead of u_0 and v_0, are used as filter parameters, then the Gabor filter $h(x,y)$ defined in (6.1) can be equivalently expressed as

$$h(x,y) = g(x,y) \bullet e^{-2\pi\omega_0 j[x\cos\theta_0 + y\sin\theta_0]} \tag{6.3}$$

where $g(x,y)$ is defined in (6.2).

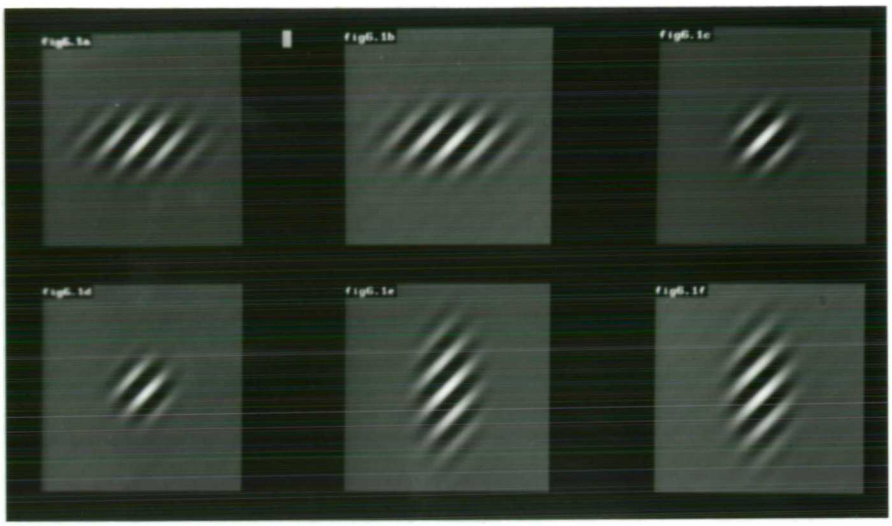


Figure 6.1. Gabor Filters under Different Aspect Ratio λ. (a) and (b), (c) and (d), and (e) and (f) are three symmetrical (the real part of $h(x,y)$ in (6.1)) and anti-symmetrical (the imaginary part of $h(x,y)$ in (6.1)) Gabor filter pairs with λ being 0.5, 1.0 and 2.0 respectively.

On the spatial frequency domain, the Gabor filter defined in (6.1) is given by

$$H(u,v) = 2\pi\sigma_x\sigma_y \bullet e^{-2\pi^2[\sigma_x^2(u+u_0)^2+\sigma_y^2(v+v_0)^2]} = G(u+u_0, v+v_0) \tag{6.4}$$

where $G(u,v)$ is the 2-D Fourier transform of $g(x,y)$ given by

$$G(u,v) = 2\pi\sigma_x\sigma_y \bullet e^{-2\pi^2(\sigma_x^2u^2+\sigma_y^2v^2)} \tag{6.5}$$

which is also a 2-D Gaussian located at the origin (0,0) on the spatial frequency domain. Clearly, $H(u,v)$ is a 2-D Gaussian centered at frequency coordinates (u_0,v_0), and has an aspect ratio of $\lambda' = \dfrac{\frac{1}{\sigma_x}}{\frac{1}{\sigma_y}} = \dfrac{\sigma_y}{\sigma_x} = \dfrac{1}{\lambda}$, i.e., the reciprocal of the aspect ratio of the filter on the spatial time domain. The distance from the center of $H(u,v)$ to the origin is $(u_0^2+v_0^2)^{1/2} = \omega_0$, i.e., the overall modulation frequency of $h(x,y)$ on the spatial time domain. The orientation of $H(u,v)$ is

at an angle from the horizontal axis equal to the axis of the modulation on the spatial time domain. It is clear that σ_x, σ_y ,u_0 and v_0 or σ_x, σ_y, ω_0 and θ_0 are the four independent variables or parameters of the Gabor filter in (6.1). They also uniquely describe the Gabor filter and obviously determine the spatial frequency and orientation bandwidths of the filter.

The filter given in (6.1) is clearly of complex nature. In practical applications, however, one is interested in Gabor filters of real nature as may be defined by

$$h(x,y) = g(x,y)\bullet\sin[2\pi(u_0x+v_0y)+\rho] \tag{6.6}$$

where $g(x,y)$ is the 2-D Gaussian defined in (6.2) and ρ is the phase of the modulating sinusoid. Customarily, ρ is either $\frac{\pi}{2}$ or 0, and $h(x,y)$ has the form of either

$$h_c(x,y) = g(x,y)\bullet\cos[2\pi(u_0x+v_0y)] \tag{6.7}$$

for $\rho = \frac{\pi}{2}$ or

$$h_s(x,y) = g(x,y)\bullet sin[2\pi(u_0x+v_0y)] \tag{6.8}$$

for $\rho = 0$. As $h_c(x,y) = h_c(-x,-y)$ and $h_s(x,y) = -h_s(-x,-y)$, $h_c(x,y)$ is sometimes called a *symmetrical* Gabor filter and $h_s(x,y)$ an *anti-symmetrical* Gabor filter.

It is trivial to show that the complex Gabor filter $h(x,y)$ in (6.1), the symmetrical Gabor filter $h_c(x,y)$ in (6.7) and the anti-symmetrical Gabor filter $h_s(x,y)$ in (6.8) are simply related to each other by

$$\begin{aligned} h(x,y) &= h_c(x,y) - jh_s(x,y) \\ h_c(x,y) &= \frac{h(x,y)+h(-x,-y)}{2} \\ h_s(x,y) &= \frac{h(x,y)-h(-x,-y)}{-2j} \end{aligned} \tag{6.9}$$

on the spatial time domain or by

$$H_c(u,v) = \frac{H(u,v)+H(-u,-v)}{2} \tag{6.10}$$

and

$$H_s(u,v) = \frac{H(u,v) - H(-u,-v)}{-2j} \tag{6.11}$$

on the spatial frequency domain. By substituting (6.4) into (6.10) and (6.11), one obtains the representations of the symmetrical and anti-symmetrical Gabor filters on the spatial frequency domain:

$$H_c(u,v) = \frac{1}{2}[G(u-u_0,v-v_0) + G(u+u_0,v+v_0)] \tag{6.12}$$

and

$$H_s(u,v) = \frac{j}{2}[G(u-u_0,v-v_0) - G(u+u_0,v+v_0)] \tag{6.13}$$

respectively. Equations (6.12) and (6.13) can, of course, have been derived directly by taking Fourier transforms upon Equations (6.7) and (6.8). (6.12) and (6.13) tell us that $h_c(x,y)$ is the sum of two Gaussians on the frequency domain located at (u_0,v_0) and $(-u_0,-v_0)$ respectively whereas $h_s(x,y)$ is the difference between the two Gaussians together with a $\frac{\pi}{2}$ phase shift at each spatial frequency.

In the subsequent discussions, symmetrical and anti-symmetrical Gabor filters with unity aspect ratio will be used for simplicity. In other words, the spatial constants σ_x and σ_y are assumed to be equal, i.e., $\sigma_x = \sigma_y = \sigma$. The two filters are therefore given by

$$h_c(x,y) = e^{-(x^2+y^2)/2\sigma^2} \bullet \cos[2\pi(u_0x + v_0y)] \tag{6.14}$$

and

$$h_s(x,y) = e^{-(x^2+y^2)/2\sigma^2} \bullet \sin[2\pi(u_0x + v_0y)] \tag{6.15}$$

respectively. If the spatial constant σ is predefined, the center frequencies (u_0,v_0) are the only two parameters to be determined to specify a Gabor filter.

6.2.2 Some Properties of Gabor Filters

To exploit the relationship between the symmetrical and anti-symmetrical Gabor filters, let us recall their representations on the spatial frequency domain Equations (6.12) and (6.13). For fixed v, $H_c(u,v)$ is a 1-D function of u, and is the sum of two 1-D Gaussians $G(u-u_0,v-v_0)$ and

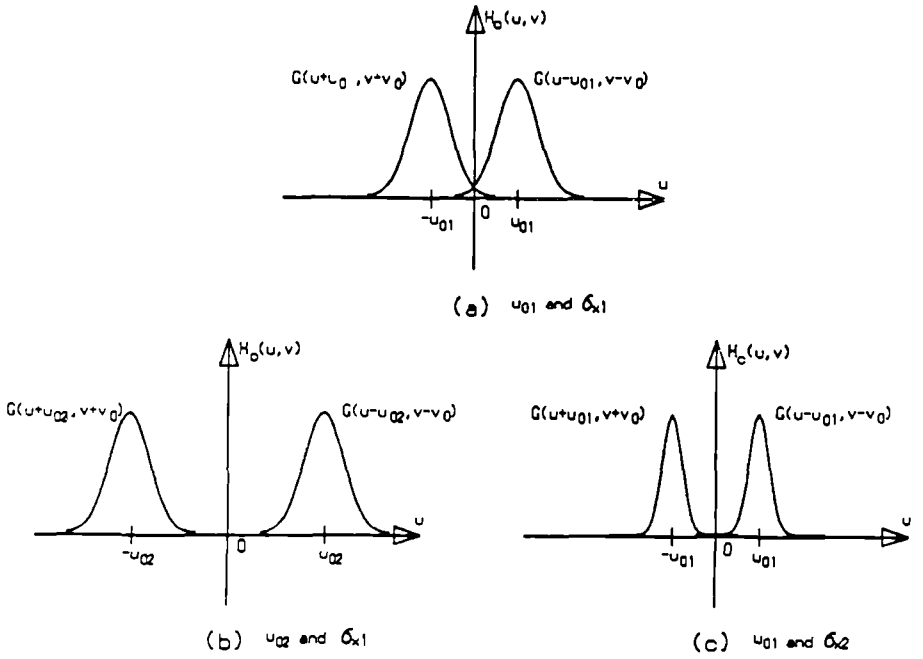


Figure 6.2. The Symmetrical Gabor Filter $H_c(u,v)$ on The Frequency Domain for Fixed v, and under Different Central Frequency u_0 and Space Constant σ_x, where $u_{01}<u_{02}$ and $\sigma_{x1}<\sigma_{x2}$.

$G(u+u_0,v+v_0)$ centered at u_0 and $-u_0$ respectively as illustrated in Fig.6.2. Similar comments can be made for fixed u and on $H_s(u,v)$. Now, let us concentrate on $G(u-u_0,v-v_0)$ and $G(u+u_0,v+v_0)$ for fixed v. Some thoughts will reveal that if the spatial constant σ_x is sufficiently large and/or the central frequency u_0 is high enough, the overlap between $G(u-u_0,v-v_0)$ and $G(u+u_0,v+v_0)$ is negligible as demonstrated in Fig.6.2. This implies that the following holds true:

$$\begin{cases} G(u-u_0,v-v_0) \approx 0; & \text{for } u \le 0; \\ G(u+u_0,v+v_0) \approx 0; & \text{for } u \ge 0; \end{cases} \tag{6.16}$$

for fixed v, large σ_x and high u_0. Substituting this into Equations (6.12) and (6.13), we have

$$H_c(u,v) \approx \begin{cases} \frac{1}{2}G(u-u_0,v-v_0); & \text{for } u \geq 0; \\ \\ \frac{1}{2}G(u+u_0,v+v_0); & \text{for } u \leq 0; \end{cases} \tag{6.17}$$

and

$$H_s(u,v) \approx \begin{cases} \frac{-j}{2}G(u-u_0,v-v_0); & \text{for } u \geq 0; \\ \\ \frac{+j}{2}G(u+u_0,v+v_0); & \text{for } u \leq 0; \end{cases} \tag{6.18}$$

From (6.17) and (6.18), one can immediately write

$$H_s(u,v) \approx \begin{cases} jH_c(u,v), & \text{for } u \geq 0; \\ \\ +jH_c(u,v), & \text{for } u \leq 0; \end{cases} \tag{6.19}$$

Recalling the definition of the *Hilbert Transform* [10], one can conclude that for fixed v, large σ_x and high central frequency u_0, $h_c(x,y)$ and $h_s(x,y)$ compose approximately a Hilbert transform pair w.r.t. variable u. Similar conclusions can be drawn upon $h_c(x,y)$ and $h_s(x,y)$ w.r.t. variable v. These conclusions justify the terming of $h_c(x,y)$ and $h_s(x,y)$ as a *quadrature* filter pair.

The increasing popularity of Gabor filters in image analysis and processing is mostly attributed to their optimization of the *uncertainty principle* of linear filters [10]. In 1-D, if $f(x)$ and $F(u)$ are the filter's impulse response and frequency response respectively, then the uncertainty principle reads

$$\Delta x \bullet \Delta u \geq \frac{1}{4\pi} \tag{6.20}$$

where Δx, the effective width in time domain, is given by

$$\Delta x = \frac{\int_{-\infty}^{+\infty} x^2 f(x) f^*(x) dx}{\int_{-\infty}^{+\infty} f(x) f^*(x) dx} \tag{6.21}$$

and Δu, the effective width in frequency domain, is defined by

$$\Delta u = \frac{\int_{-\infty}^{+\infty} u^2 F(u) F^*(u) du}{\int_{-\infty}^{+\infty} F(u) F^*(u) du} \tag{6.22}$$

where * denotes taking complex conjugate operation. Physically interpreting, Δx is a measure of the localization of the filter $f(x)$ on the time domain. Smaller Δx implies that filter $f(x)$ has less spread and thus is better localized on the time domain. Similarly, Δu is a measure of the localization and bandwidth of the filter on the frequency domain. Smaller Δu means a narrower bandwidth and thus a better localization of the filter on the frequency domain. With these interpretations in mind, one knows that the uncertainty principle of (6.20) states that a linear filter can not achieve simultaneous localizations on both domains. The price paid for a better localization on one domain is necessarily a worsened localization on the other. In 1946, **Gabor** [9] proved that $(\Delta x)(\Delta u)$ achieves its low bound $\frac{1}{4\pi}$, i.e., achieves optimal localizations simultaneously on both domains, only when $f(x)$ is specified by

$$f(x) = e^{-\frac{x^2}{2\sigma^2}} \bullet e^{-2\pi j u_0 x} \tag{6.23}$$

which has been conventionally referred to as a 1-D Gabor filter. The 1-D uncertainty principle and Gabor's work [9] have been subsequently extended to 2-D by **Daugman** [11]. He formulated the 2-D uncertainty principle as follows:

$$(\Delta x)(\Delta y)(\Delta u)(\Delta v) \geq \frac{1}{16\pi^2} \tag{6.24}$$

where the individual effective widths are defined similarly as in (6.21) and (6.22) except the 1-D functions $f(x)$ and $F(u)$ are now replaced by 2-D functions $f(x,y)$ and $F(u,v)$ respectively. Daugman [11] also demonstrated that 2-D filters having the form given in (6.1) optimize the 2-D uncertainty principle (6.24). Hence, 2-D complex Gabor filters achieve optimal localization/resolution simultaneously on both the spatial time domain and the spatial frequency domain. Recalling (6.9), one understands that the symmetrical and anti-symmetrical Gabor filters specified in (6.14) and (6.15) also minimize the uncertainties. To exploit the relationship between Gabor filters and the human visual system, let us now turn our attention to the visual cortex.

6.2.3 The Visual Cortex

The visual cortex is probably the most important station on the human visual pathway. After being pre-processed at preceding stations by means of low-pass or high-pass filtering, a 2-D light distribution or an image undergoes some critical processing such as feature extraction and pattern discrimination in the visual cortex [12]. The visual cortex is a folded layer of cortical neurons or cells of about 2mm in thickness. It is located at the back of the brain. There are about 10^{10} neurons contained in the visual cortex which are hierarchically organized in layers or slabs. Most of cortical neurons or cells can be categorized as *simple cells, complex cells* and *hypercomplex cells* as identified by **Hubel** and **Wiesel** [13]. Simple cells were principally found in the middle cortical layers whereas complex cells were seen most frequently in the superficial and deep layers of the visual cortex. Hubel and Wiesel also suggested a simplified model for the functional organization of various cells. The model states that lateral geniculate neurons converge on simple cortical cells (i.e., simple cortical cells receive direct input from lateral geniculate cells), simple cells in turn converge on complex cells, and complex cells on hypercomplex cells.

Associated with each cortical neuron or cell is the concept of *receptive field*. The receptive field of a cortical cell is simply the cortical area which can influence the behaviour of that cell. Because of the parallel and feedback connections, a receptive field may be divided into regions. Some of them produce excitation (indicating excitatory regions) whereas some others inhibition (indicating inhibitory regions). These excitatory and inhibitory regions are often called *ON areas* and *OFF areas* respectively. With these notions in mind, we can now define simple cells as those whose receptive fields are composed of spatially interleaved "ON" and "OFF" areas separated by parallel straight lines. Simple cells have been shown to be primary and general purpose analyzers of afferent visual information to the visual cortex [14]. Thus the following discussions will be primarily concerned with simple cells.

Although tremendous progress has been made on the understanding of the principles of the visual cortex, the understanding is far from complete. The question of how the visual cortex processes the afferent visual information (say pictorial information) has been debated in both perceptual and neurophysiological vision research communities in the past two decades. Over the years, two different approaches have been emerged. The first based on **Hubel** and **Wiesel**

[13] has concentrated attention on the spatial organization of cortical receptive fields, and edges and bars have been regarded as the elementary visual features extracted by simple cells. In this model, the simple cells are assumed to be bar and edge detectors. The alternative approach, first established by **Compbell** and **Robson** [15], is based on the application of the spatial frequency (Fourier) method, and by concentrating attention on the sensitivity of cortical cells to sinusoidal gratings of varying spatial frequency. In this model, the visual cortex is envisaged as a set of parallel spatial filters or channels, and each simple cell has been shown to be sensitive only to a limited range of spatial frequencies, i.e, an independent bandpass filter of narrow bandwidth [15]. Although recent conciliatory voices [14,16] have affirmed that the debate over the two models was based on an illusory dichotomy and the two approaches are equivalent to each other within the linear operating range of simple cells, the notion of *spatial frequency channels* seems to be preferred [6,13,17,18]. This is also the model adopted in this thesis.

To understand the mechanisms of the cortical channels, a detailed study is needed on simple cells. Various aspects of the simple cells have been reported in a number of papers [19-21]. In [19], for instance, **Maffei** and **Fiorentini** discoverd that the simple cortical cells have the narrowest tuning curves when compared with those of other neurons. They concluded that the narrow tuning curves for the spatial frequency of simple cortical cells could be the neural substrate for the spatial frequency channels that have been postulated on the basis of psychophysical experiments. Treating the visual cortex as a Fourier analyzer is based on the tacit assumption that simple cortical cells operate linearly on afferent visual information. This assumption is rationalized by **Movshon** and **Tolhurst** [20], and by **Andrews** and **Pollen** [21]. The data reported in [21] suggests that simple cells may perform approximately linear summation of inputs to the visual cortex. Nevertheless, both papers have found that the output of a simple cell is generally non-linear since a half-wave rectification has been observed at the output stage of simple cells. This may be due to the low or absent resting discharge in these cells at the underlying instants. Complementary to these findings is the discovery of **Pollen** and **Ronner** [18] on the phase relationships between adjacent simple cells. They found adjacent simple cells tuned to the same spatial frequency, orientation and direction respond identically except for a difference in spatial phase by approximately 90^0. If simple cells are envisaged as narrow bandpass filters, Pollen and Ronner's discovery implies that two adjacent simple cells form a *quadrature* bandpass filter pair. Recalling the non-linearity at the output stage of simple

cells, one knows that for a given pair of adjacent simple cells, a second pair with response polarities opposite to the given pair is required to preserve the information that would otherwise be lost due to half-wave rectification. Therefore, to describe a channel in the visual cortex tuned to a specific band of spatial frequency, orientation and direction, four elemental simple cells are required. These four cells scan a common subdomain of visual space at the preferred spatial frequency, orientation and direction.

6.2.4 The Cortical Channel Model

Let the receptive field profiles of two adjacent simple cells be denoted by $h_1(x,y)$ and $h_2(x,y)$ respectively. If each simple cell is regarded as a bandpass filter as discussed above, then $h_1(x,y)$ and $h_2(x,y)$ form a quadrature filter pair. To completely specify a channel, we have shown that another quadrature filter pair $h_3(x,y)$ and $h_4(x,y)$ are required with $h_3(x,y) = -h_1(x,y)$ and $h_4(x,y) = -h_2(x,y)$. If we denote the input visual signal to a channel by $p(x,y)$ and the outputs of the underlying four simple cells by $q_1(x,y)$, $q_2(x,y)$, $q_3(x,y)$ and $q_4(x,y)$ respectively, then most of the properties of simple cells outlined in the preceding section can be summarized by the following mathematical equations:

$$\begin{aligned} q_1(x,y) &= \frac{1}{2}\left[|h_1(x,y)^{**}p(x,y)| + h_1(x,y)^{**}p(x,y)\right] \\ q_2(x,y) &= \frac{1}{2}\left[|h_2(x,y)^{**}p(x,y)| + h_2(x,y)^{**}p(x,y)\right] \\ q_3(x,y) &= \frac{1}{2}\left[h_3(x,y)^{**}p(x,y)| + h_3(x,y)^{**}p(x,y)\right] \\ q_4(x,y) &= \frac{1}{2}\left[|h_4(x,y)^{**}p(x,y)| + h_4(x,y)^{**}p(x,y)\right] \end{aligned} \tag{6.25}$$

where ** denotes 2-D linear convolution. At this point, it is still unclear what the channel output $q(x,y)$ should be. We simply assume that $q(x,y)$ is some function of $q_1(x,y)$, $q_2(x,y)$, $q_3(x,y)$ and $q_4(x,y)$. Specifically, we assume that $q(x,y)$ is given by

$$q(x,y) = \left[[q_1(x,y)+q_3(x,y)]^2 + [q_2(x,y)+q_4(x,y)]^2\right]^{\frac{1}{2}} \tag{6.26}$$

Let

$$\begin{aligned} a(x,y) &= h_1(x,y)^{**}p(x,y) \\ b(x,y) &= h_2(x,y)^{**}p(x,y) \end{aligned} \tag{6.27}$$

and recall (6.25) and the relations between $h_i(x,y)$, $i=1,2,3,4$, we can rewrite (6.26) as

$$q(x,y) = \left[a^2(x,y)+b^2(x,y)\right]^{\frac{1}{2}} \tag{6.28}$$

So far, little has been said about the general shape of the receptive field profiles of simple cells in the visual cortex, i.e., the functional form of $h_i(x,y)$, $i=1,2,3,4$. Over the years, numerous experiments have been conducted to characterize the receptive fields of simple cortical cells. Consistent experimental results have convincingly confirmed that the receptive field profiles of simple cortical cells often resemble even-symmetrical or odd-symmetrical Gabor filters, i.e., their receptive field profiles can be described by the product of a Gaussian and either a cosine or sine function [6,7,11,16]. In other words, the receptive field profiles of simple cells can be described either by

$$h_e(x,y) = e^{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right]} \bullet cos[2\pi(u_0x+v_0y)] \tag{6.29}$$

corresponding to even-symmetrical or by

$$h_o(x,y) = e^{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right]} \bullet sin[2\pi(u_0x+v_0y)] \tag{6.30}$$

corresponding to odd-symmetrical. If a simple cell has an even-symmetrical receptive field profile, its adjacent partner often has an odd-symmetrical profile or vice versa as discussed in [7]. Note, $h_e(x,y)$ in (6.29) and $h_o(x,y)$ in (6.30) are exactly the quadrature Gabor filter pair defined in (6.7) and (6.8). Therefore, the four simple cortical cells or cortical filters $h_i(x,y)$, $i=1,2,3,4$, which uniquely specify a given cortical channel, may be defined as

$$\begin{aligned} h_1(x,y) &= h_c(x,y) \\ h_2(x,y) &= h_s(x,y) \\ h_3(x,y) &= -h_c(x,y) \\ h_4(x,y) &= -h_s(x,y) \end{aligned} \tag{6.31}$$

where $h_c(x,y)$ and $h_s(x,y)$ are the symmetrical and anti-symmetrical Gabor filters defined in (6.7) and (6.8). Fig.6.3 displays these four filters on the spatial time domain as intensity images. By combining Equations (6.27), (6.28) and (6.31), we obtain the cortical channel model to be employed in the subsequent discussions:

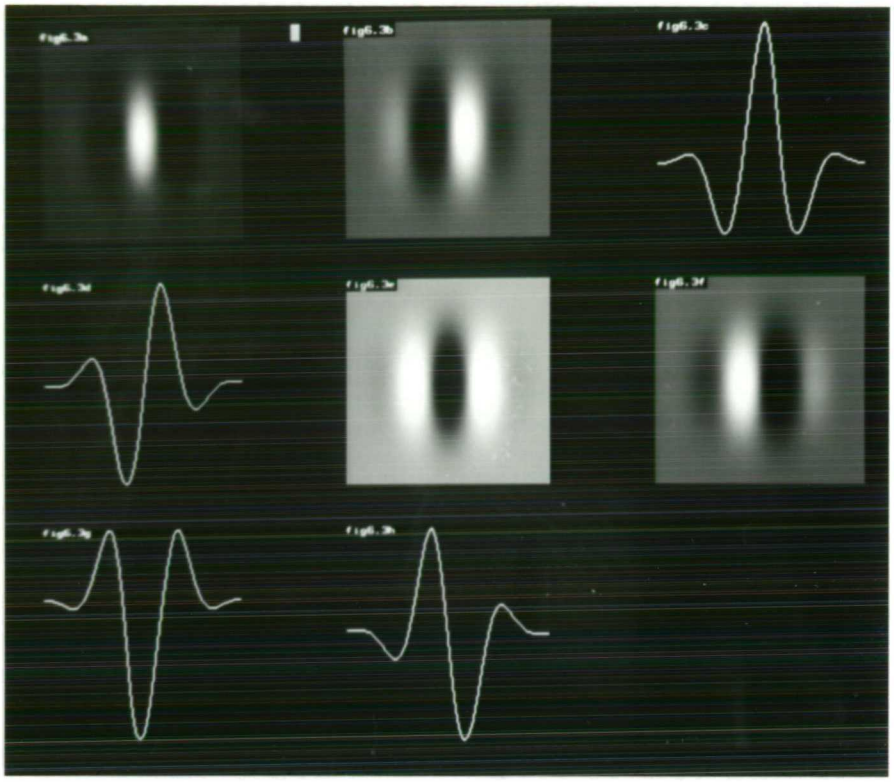


Figure 6.3. The Four Elemental Gabor Filters on The Spatial Time Domain. (a) $h_c(x,y)$; (b) $h_s(x,y)$; (c) Profile of the Middle Row of (a); (d) Profile of the Middle Row of (b); (e) $-h_c(x,y)$; (f) $-h_s(x,y)$; (g) Profile of the Middle Row of (e) and (h) Profile of the Middle Row of (f).

$$\begin{aligned} a(x,y) &= h_c(x,y) ** p(x,y) \\ b(x,y) &= h_s(x,y) ** p(x,y) \\ q(x,y) &= \left[a^2(x,y) + b^2(x,y)\right]^{\frac{1}{2}} \end{aligned} \tag{6.32}$$

where $p(x,y)$ and $q(x,y)$ are the input and output images of the underlying channel respectively. This model is schematically sketched in Fig.6.4. The appropriateness of model (6.32) in the context of texture analysis is theoretically justified in Chapter 9.

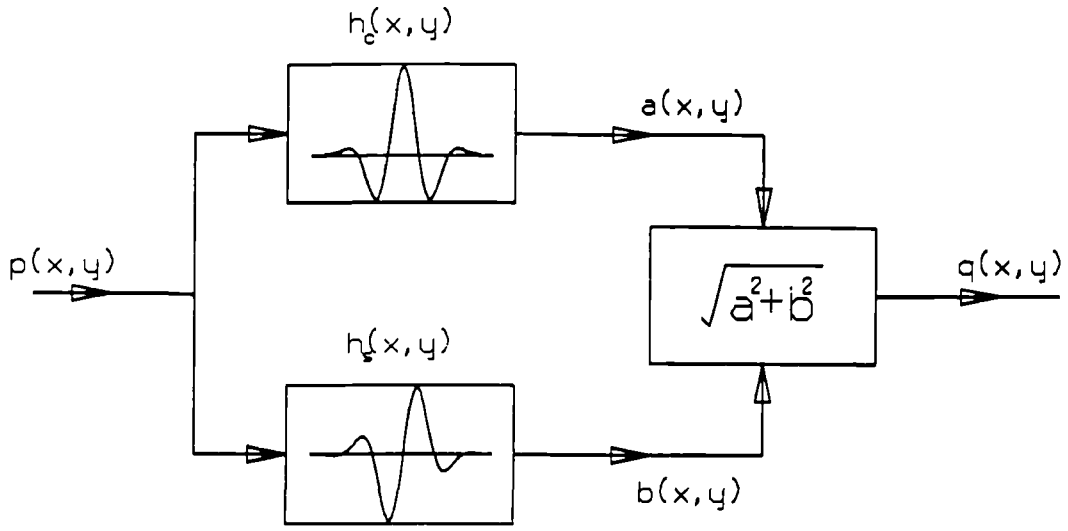


Figure 6.4. Schematic Diagram of the Proposed Cortical Channel Model (6.32).

At this point, one may ask: Why should the visual cortex process visual information in terms of Gabor filters? To answer this question, we are in concordance with **Daugman** [6] stating that "... the visual system is concerned with extracting information jointly in the 2D space domain and in the 2D frequency domain, and because of the incompatibility of these two demands, has evolved towards the optimal solution via 2D channels that roughly approximate 2D Gabor filters."

6.3 Channel and Filtering Pattern Design

The visual cortex has been hitherto hypothesized as being composed of a set of cortical channels each of which is modeled by (6.32). If such a channel is thought to be a basic unit in the visual cortex for feature extraction, then an immediate question to ask is how many such channels should be employed in order to extract most of the textural information present in a given image? To answer this question, we shall, in this section, first describe two experiments to investigate the frequency component resolution characteristics of the HVS. Such an investigation, together with some known results in the literature, will guide us in the determination of the number of channels and the underlying filter parameters, and thus the design of the filtering pattern for texture feature extraction.

6.3.1 The Frequency Component Resolution Characteristics of The HVS

In the following discussions, we will assume that if two textures appear visually identical, i.e., indiscriminable to a human observer without scrutinization, they will be said to contain identical textural information. In order to extract most of, if not all, the textural information contained in a given image via the implementation of the cortical channel model (6.32), the appropriate number of channels to be implemented has to be first determined. If each individual frequency component indicates a perceivable texture feature in the spatial domain, the number of channels to be implemented would be the number of frequency components with each channel being tuned to one particular component. If, in general, only a group of components are representative of what can be perceived in an image, then the number of channels needed being implemented may be reduced dramatically. Therefore, we have to resolve, first of all, whether each individual frequency component is indicative of perceivable texture features. Putting this another way, we have to answer the question: Does the HVS have a limited resolution for frequency components? To answer this question, two experiments are designed.

EXPERIMENT I: Frequency Component Resolution for Stochastic Textures

By stochastic texture, we mean textures which do not possess any globally perceivable regularities, periodicities or orientations. On the spatial frequency domain, these textures do not exhibit any perceivable spectral peaks. To investigate the frequency resolution of the HVS under stochastic textures, we follow a similar methodology as **Caelli** [22]. For a given texture image $f(x,y)$, its Fourier transform $F(u,v)$ is first calculated. Then a circular region S of radius R centered at (u_0,v_0) is chosen in the frequency plane. The average energy of S is determined and assigned to each individual frequency component in S resulting in a Fourier transform $\hat{F}(u,v)$ having the same phase spectrum as the original transform $F(u,v)$. Note, taking into account the symmetry of energy spectrum, it is tacitly assumed that the same energy averaging operation has been performed at location $(-u_0,-v_0)$ on the frequency domain as well. $\hat{F}(u,v)$ is then inverse Fourier transformed into a texture $\hat{f}(x,y)$. The two textures $f(x,y)$ and $\hat{f}(x,y)$ should not be discriminable visually from each other if

1. R is of appropriate size;

2. only the total energy in the region S is processed in the HVS; and
3. the HVS takes all the frequency components instead of each individual component in S as an indivisible unit as far as visual texture discrimination and coding is concerned.

Note, hypothesis 3) is equivalent to the hypothesis that the HVS has a limited resolution on frequency components constrained by some upper limits.

Fig.6.5 illustrates the visual influences of the energy spectrum averaging operation described above on a stochastic texture Handmade Paper (D57) taken from **Brodatz** [23].

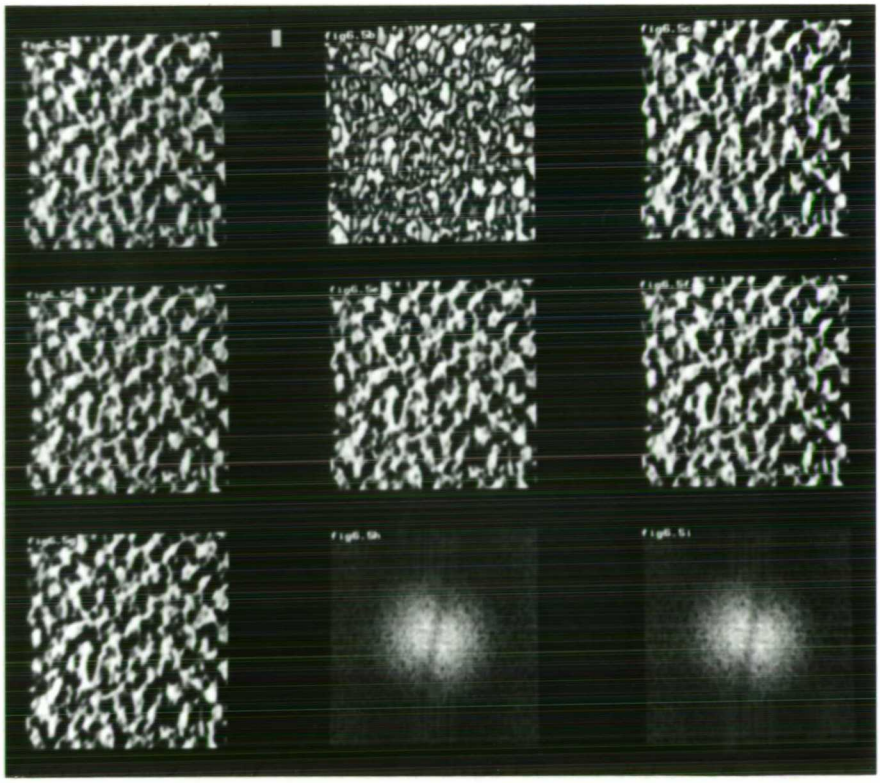


Figure 6.5. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Stochastic Texture D57 [23]. The Averaging Radius R in pixels is 5. For detailed caption, see text.

The original texture image Fig.6.5(a) is digitized into 8 bits on a 128x128 grid. Histogram equalization has been performed before the application of the discrete Fourier transform. A series of energy averaging operations are performed along the horizontal direction with the central frequencies being 10 pixels apart on the discrete Fourier domain. These central frequencies (u_0,v_0) are (0,0), (10,0), (20,0), (30,0), (40,0) and (50,0) respectively. At all central frequencies, the region radius R has been kept to be 5 pixels. Fig.6.5(b)-(g) show the resultant textures $\hat{f}(x,y)$ corresponding to the central frequency (0,0), (10,0), (20,0), (30,0), (40,0) and (50,0) respectively. Fig.6.5(h) is the energy spectrum of Fig.6.5(e) whereas Fig.6.5(i) shows the original energy spectrum of Fig.6.5(a). While Fig.6.5(b) is significantly different from the original texture Fig.6.5(a), the others (Fig.6.5(c)-(g)) are hardly discriminable from Fig.6.5(a). In fact, when human observers are asked to judge the differences between Fig.6.5(c)-(g) and Fig.6.5(a), most of them claim with conviction that these textures are identical.

A similar illustration to Fig.6.5 is given in Fig.6.6 except that R is now increased to 10 pixels. The increase in averaging area has made two resultant textures (Fig.6.6(b) and (c)) significantly different from the original texture Fig.6.6(a). It should be noticed, however, while Fig.6.6(a) and Fig.6.6(e) appear visually identical, their energy spectra (Fig.6.6(i) and Fig.6.6(h)) are apparently distinct.

Similar experiments under other directions and texture types, and with central-frequency-dependent radius R have also been conducted but not included here because of space limitation. Throughout these experiments, the following has been observed if the two textures $f(x,y)$ and $\hat{f}(x,y)$ have to be visually indiscriminable:

1. If the central frequency (u_0,v_0) is near the origin (0,0) on the frequency domain, then $R \to 0$. This demonstrates the importance of background intensity (average intensity) and/or contrast in visual perception of textures;
2. In general, R in pixels is proportional to the magnitude of the central frequency $\omega_0 = [u_0^2 + v_0^2]^{1/2}$; and
3. As far as visual texture discrimination is concerned, the important frequency components are likely within $\omega_0 \leq 30$ if the textures under study are of size 128x128.

Overall, the above observations are consistent with those of **Caelli** and his co-workers [22,24].

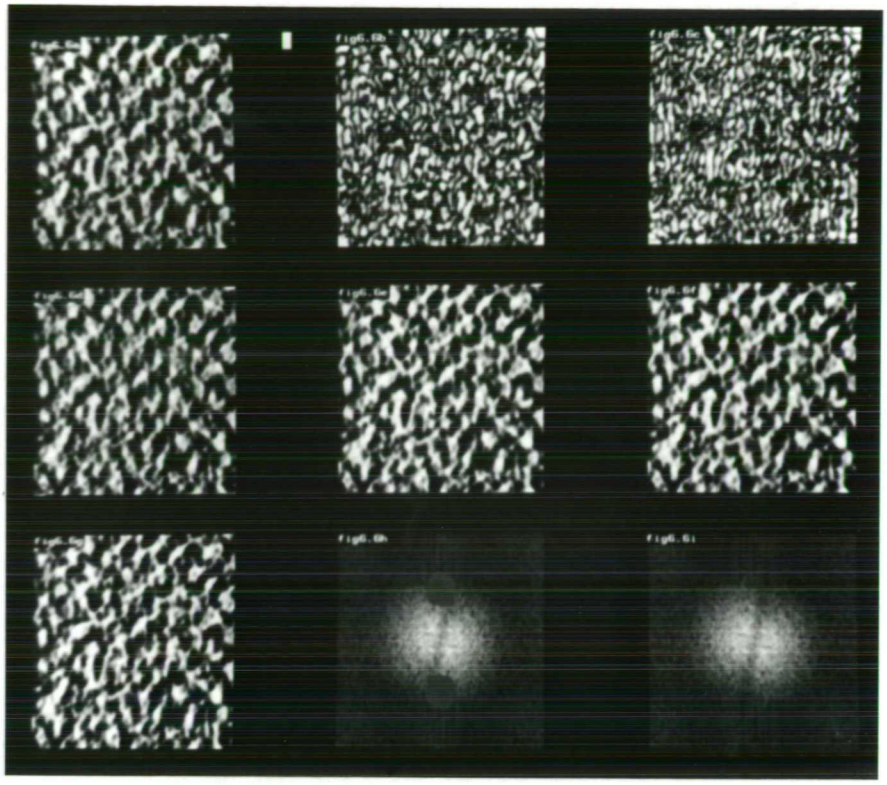


Figure 6.6. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Stochastic Texture D57 [23]. The Averaging Radius in Pixels is 10. For detailed caption, see text.

At this point, we have apparently answered the question cited at the beginning of this subsection: the HVS does have a limited resolution on frequency components, and the type of two-dimensional spatial frequency information which is representative of what can be perceived in an image on the spatial domain is certainly not defined by individual frequency components, rather the upper limits for frequency component resolution are limited to a set of small spectral regions.

EXPERIMENT II: Frequency Component Resolution for Structural Textures

Unlike stochastic textures, structural textures are those which exhibit globally perceivable structures like periodicities, regularities and orientations. On the spatial frequency domain, these textures often display prominent energy spectral peaks. As we concluded in Experiment I that the frequency resolution of the HVS measured in pixels (if the energy spectrum is regarded as an image array) is dependent on the central frequency, let us now concentrate on the frequency component resolution of the HVS around energy spectral peaks. We want to demonstrate the importance of energy spectral peaks in visual texture discrimination. The same experimental methodology as in Experment I is followed.

The first example of structural textures is given in Fig.6.7(a). This texture is also taken from [23] (Oriental Straw Cloth, D53). From its energy spectrum shown in Fig.6.7(i), the presence of a few energy spectral peaks is self-evident. The most prominent spectral peak is located at (18,0) or equivalently at (−18,0). Accordingly, we set the central frequency as $u_0 = 18$ and $v_0 = 0$, and keep the central frequency unchanged while applying a set of region radii $R = 1,2,3,5,7,9$ in pixels. Fig.6.7(b)-(g) show the resultant textures $\hat{f}(x,y)$ corresponding to radius size 1,2,3,5,7 and 9 respectively. By comparing these images with the original texture Fig.6.7(a), it is clear that with the increase in R, the resultant $\hat{f}(x,y)$ is becoming more and more discriminable from the original texture. Note, even with the smallest radius $R = 1$, the resultant texture $\hat{f}(x,y)$ (Fig.6.7(b)) is still easily discriminable from $f(x,y)$ (Fig.6.7(a)). The energy averaging operation at $R = 9$ is illustrated in Fig.6.7(h) showing the energy spectrum of Fig.6.7(g).

A second example of structural textures is presented in Fig.6.8. The original texture Cotton Canvas (D77) taken from [23] is shown in Fig.6.8(a). The layout of Fig.6.8 is similar to that of Fig.6.7. The same set of radii have been used. The energy spectrum of the original texture given in Fig.6.8(i) shows two prominent spectral peaks located at (−9,18) and (−18,0) respectively. We have chosen the first peak because it does not correspond to either horizontal direction or vertical direction. Thus, the central frequency is $u = -9$ and $v_0 = 18$. Fig.6.8(b)-(g) present the resultant textures $\hat{f}(x,y)$ corresponding to radius size 1,2,3,5,7 and 9 respectively. Again, even with the smallest region size $R = 1$, the resultant texture $\hat{f}(x,y)$ (Fig.6.8(b)) is discriminable from Fig.6.8(a). Similarly, the textures in Fig.6.8(b)-(g) are becoming more and more discriminable from Fig.6.8(a) as R increases. In fact, from Fig.6.8(a) to Fig.6.8(g), the original texture, being a regular layout of roughly hexagon-shaped blobs, is step by step

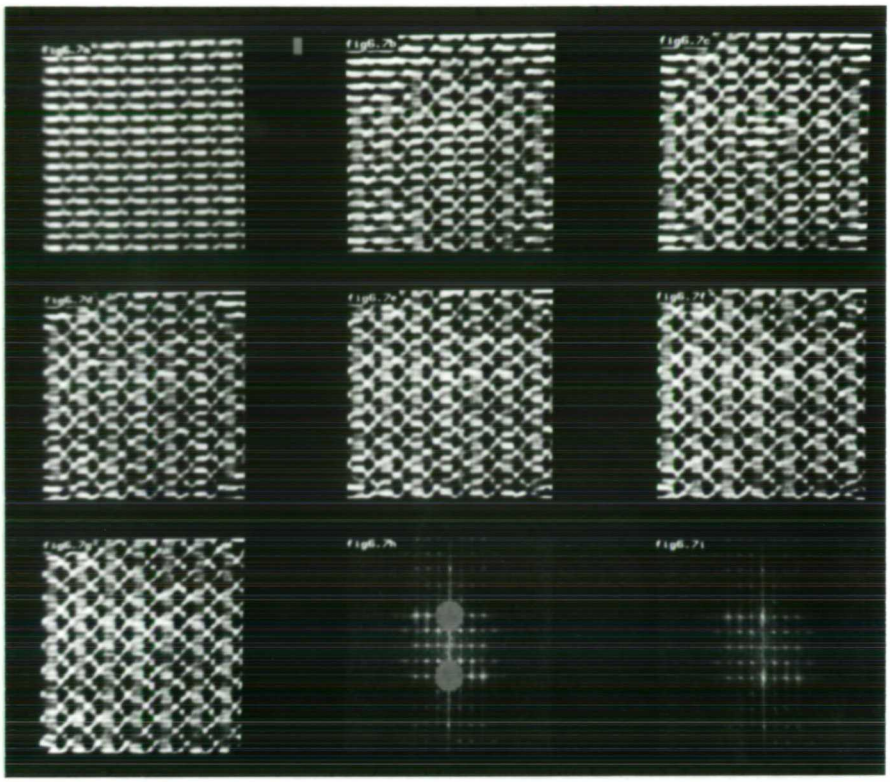


Figure 6.7. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Structural Texture D53 [23]. For detailed caption, see text.

degenerated into a texture of roughly parallel horizontal strips.

In summary, if the two textures $f(x,y)$ and $\hat{f}(x,y)$ have to be visually identical, then the demonstrations presented in Fig.6.7 and Fig.6.8 illustrate that

1. If the central frequency (u_0,v_0) is near an energy spectral peak, the averaging region size $R \to 0$. This clearly demonstrates the importance of spectral peaks in texture discrimination;
2. For textures with a prominent global orientation θ, the averaging region size R along θ tends to be very small. This statement is deducible from (1).

While for stochastic textures, we have obtained similar conclusions as [22,24], the above two

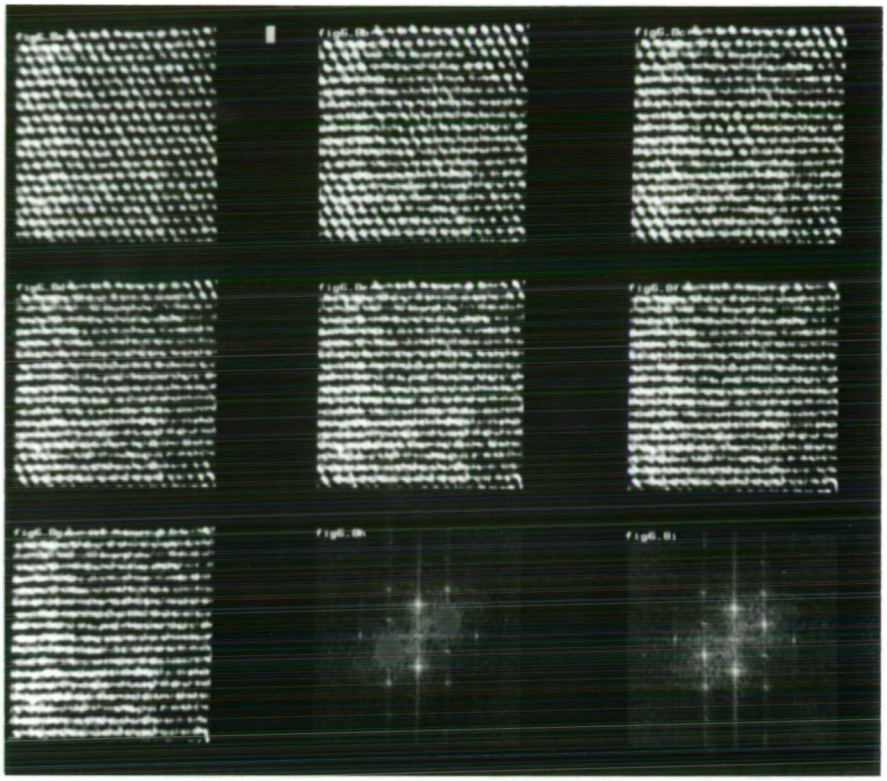


Figure 6.8. Illustration of the Visual Influences of Energy Spectrum Averaging Operations on a Structural Texture D77 [23]. For detailed caption, see text.

conclusions are new to our work.

6.3.2 Filtering Pattern for Texture Feature Extraction

Although the discussions in the preceding section can not directly guide us to determine the number of channels to be implemented and the associated filter parameters, the conclusions made there on the frequency component resolutions of the visual system are well instructive for us to tackle the problem. In particular, the statement (2) in Experiment I tells us that the effective area a filter occupies on the spatial frequency domain should be made to be proportional to the magnitude of the central frequency the underlying channel is tuned to. This

has the effect of decreasing the spatial extent of the filter on the spatial domain with increasing central frequency, which has been justified by physiological evidence [25]. Secondly, the statement (3) in Experiment I implies that as far as texture feature extraction and discrimination are concerned, the spatial frequency domain need not be uniformly covered, and that higher frequency components may be ignored. This is reasonable since textures are usually bandpass signals and high frequency components are more likely due to noise.

Having realizing the implications of these statements, we choose four frequencies and four orientations. This gives us 4x4=16 central frequency locations and thus 16 channels to be implemented. The four frequencies are ω_0 = 4,8,16 and 32 corresponding to wavelength in pixels λ = 32,16,8 and 4. These four frequencies are found to be the peak sensitivity frequencies in the visual channel mechanisms described by **Wilson** and **Bergen** [26] and subsequently found to provide a good model for texture similarity judgements by **Harvey** and **Gervais** [27]. The four orientations are $\theta_0 = 0^0,45^0,90^0$ and 135^0. Therefore, along each orientation, there are four channels at distinct central frequencies. Similarly, at each central frequency, there are four channels tuned to different orientations. As mentioned in Section 6.2, to uniquely describe a channel requires the simultaneous specification of σ_x,σ_y as well as ω_0 and θ_0. For simplicity, all Gabor filters have been chosen to be circularly symmetrical (unity aspect ratio), i.e., $\sigma_x = \sigma_y = \sigma$. Specifically, we have chosen σ = 0.045, 0.0225, 0.015 and 0.01125 associated with frequency ω_0 = 4,8,16 and 32 respectively.

In summary, the chosen filtering pattern for texture feature extraction in this chapter is described by the following:

$$
\begin{aligned}
a_{ij}(x,y) &= h_{cij}(x,y) ** p(x,y) \\
b_{ij}(x,y) &= h_{sij}(x,y) ** p(x,y) \\
q_{ij}(x,y) &= \left[a_{ij}^2(x,y) + b_{ij}^2(x,y)\right]^{1/2} \\
h_{cij}(x,y) &= e^{-\frac{x^2+y^2}{2\sigma_i^2}} \bullet cos[2\pi\omega_{0i}(xcos\theta_{0j} + ysin\theta_{0j})] \\
h_{sij}(x,y) &= e^{-\frac{x^2+y^2}{2\sigma_i^2}} \bullet sin[2\pi\omega_{0i}(xcos\theta_{0j} + ysin\theta_{0j})]
\end{aligned}
\qquad (6.33)
$$

where ω_{0i} = 4,8,16,32 and σ_i = 0.045,0.0225,0.015,0.01125 for i = 1,2,3,4 respectively, and $\theta_j = 0^0,45^0,90^0,135^0$ for j = 1,2,3,4 respectively. For visual illustration, the filtering pattern is sketched in Fig.6.9.

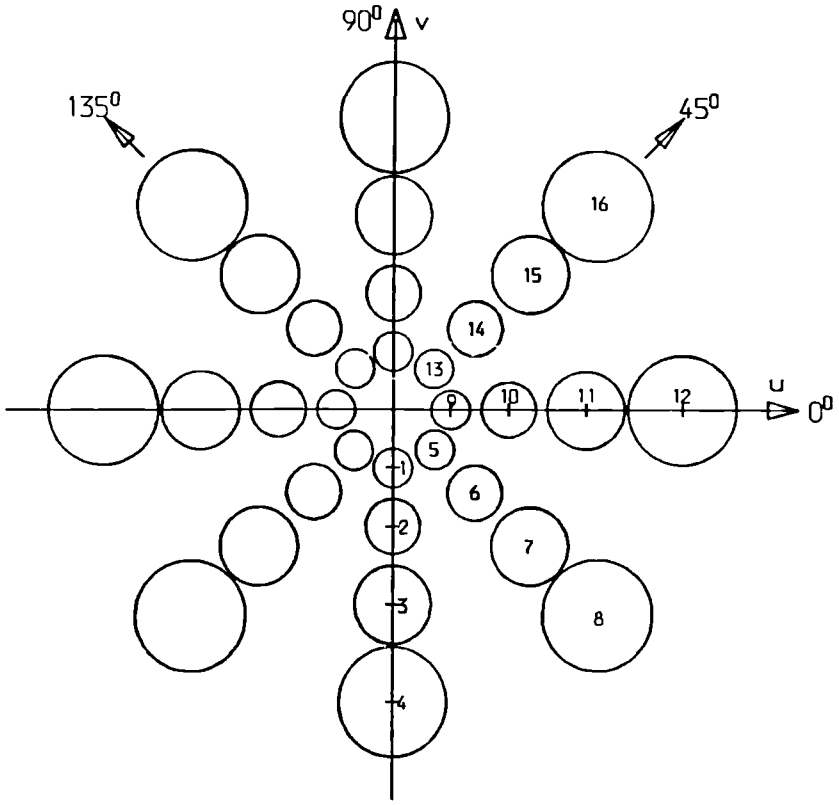


Figure 6.9. Illustration of the Proposed Filtering Pattern for Texture Feature Extraction. The number in each circle is the reference order of the underlying filter such as Filter 1, Filter 2, etc..

6.4 Texture Feature Extraction via Gabor Filtering

Having specified the filtering pattern (i.e., the number and the positions of the channels to be implemented) in the last section, we are now in the position to present the proposed algorithm for texture feature extraction and classification. The procedure for feature extraction will be given a full mathematical description. The defined texture features are interpreted in terms of some perceptual texture features such as coarseness, fineness and directionality. The classification scheme and results using the defined features compose the last part of this section.

6.4.1 Outline of The Proposed Algorithm

As we mentioned in the first section, the proposed algorithm for texture analysis belongs to a class of very recent and promising techniques based on what we called a Multichannel

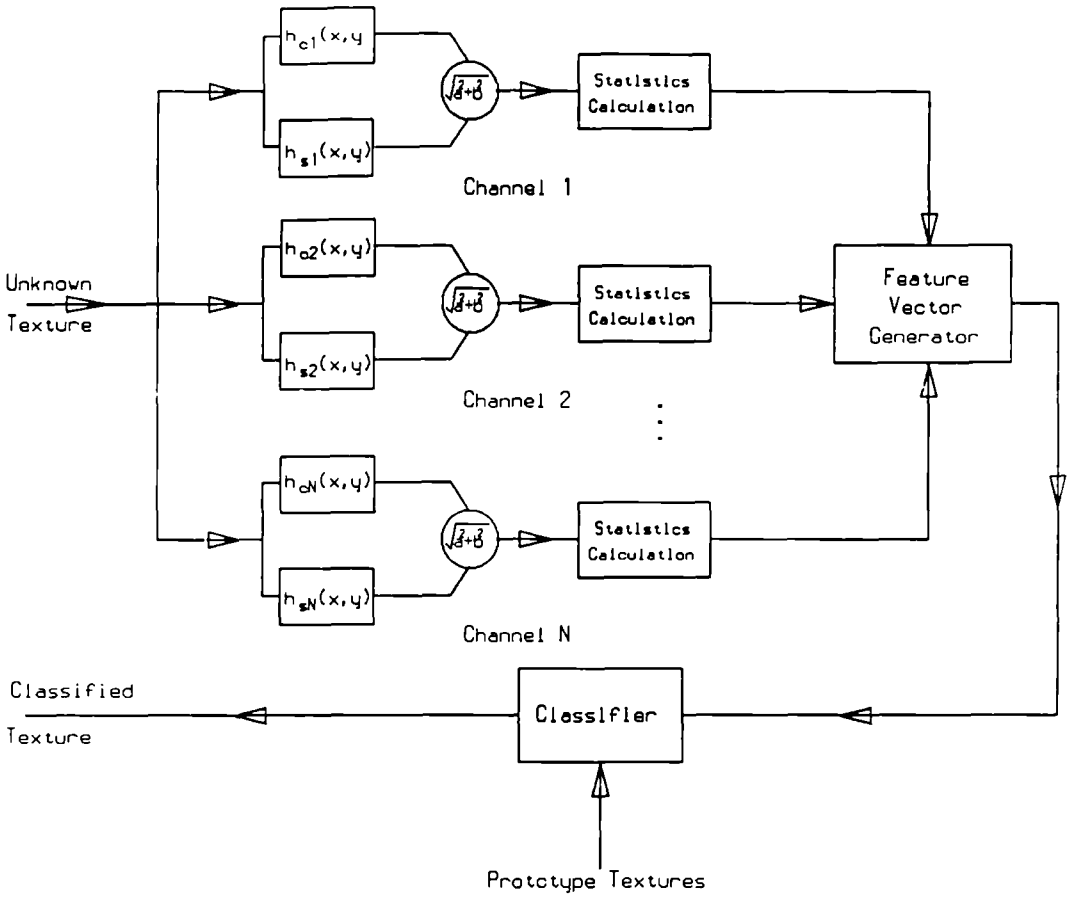


Figure 6.10. The Block Diagram of the Proposed Algorithm for Texture Feature Extraction and Classification.

Spatial Filtering Model (MSFM). Fig.6.10 shows its block diagram. The algorithm involves N parallel channels each of which employs a pair of Gabor filters $h_{cn}(x,y)$ and $h_{sn}(x,y)$ having opposite symmetry, i.e., having quadrature phase relations as we discussed. The overall structure may be regarded as being composed of four stages namely a) implementation of the cortical channel model; b) channel statistics calculation; c) feature vector generation; and d) sample classification. As each channel is tuned to a specific portion of spatial frequencies and orientations on the spatial frequency domain, it can be thought as being matched to specific texture features or "textons" such as blobs of certain sizes and orientations on the spatial time domain. It is hoped that the higher-order statistics (order $\geq$ 2) of the input image to the cortical channel can be reflected in the first-order statistics of the channel output image. This ability of

Gabor filters to reduce the statistical complexity has been demonstrated by **Turner** [28] with artificial textures. This ability of statistical complexity reduction has encouraged us, at the second stage of the algorithm, to compute the first-order statistics of the output image from the cortical channel as texture features. As the histograms of channel output images are more or less Gaussian-like (See Fig.6.11 and Fig.6.12), only the mean (average intensity) and the standard deviation (square root of intensity variance) for each channel output image are calculated since a Gaussian is uniquely specified by its mean and standard deviation. Thus, for each channel, two texture features are defined: the mean value and the standard deviation of the channel output image. In the subsequent discussions, we will refer to mean and standard deviation features simply as **channel means** and **channel deviations** respectively. Recalling that the total number of cortical channels implemented is 16 (4 frequencies times 4 orientations), we eventually define, at the stage of feature vector generation, 32 texture features for the input texture sample. Obviously, these 32 features contain 16 channel means and 16 channel deviations. At this stage, the input texture sample is represented by a feature vector of dimension 32. In order to classify the input sample, this feature vector is then fed into a chosen classifier.

This completes the description of the principles of the proposed algorithm.

6.4.2 Texture Feature Extraction

As the classification stage in the proposed algorithm is essentially supervised, classifier training is required. In other words, "representative" feature vectors of prototype textures have to be available at the classifier. In this chapter, the 14-class database shown in Fig.5.2 is employed. As we did in Chapter 5, for classifier training, 16 non-overlapped samples of size 128x128 for each texture class are derived from the corresponding original sample of size 512x512. Each derived sample undergoes a histogram equalization. To simplify the following mathematical descriptions for the texture feature extraction procedure, we assume all the signals/images and functions involved are continuous. To proceed further, we define a set of notation given as follows:

- $p_{ks}(x,y)$: The s^{th} sample from texture class k; s = 1,2,...,16 and k = 1,2,...,14;

- $q_{ijks}(x,y)$: The channel output image from the $(ixj)^{th}$ channel which is specified by channel parameters $(\omega_{0i},\theta_{0j},\sigma_i)$ with the input image being $p_{ks}(x,y)$ (See (6.32));
- $a_{ijks}(x,y)$: The output image from the symmetrical Gabor filter $h_{cij}(x,y)$ in the $(ixj)^{th}$ channel with the input image being $p_{ks}(i,j)$ (See (6.32));
- $b_{ijks}(x,y)$: The output image from the anti-symmetrical Gabor filter $h_{sij}(x,y)$ in the $(ixj)^{th}$ channel with the input image being $p_{ks}(i,j)$ (See (6.32));
- m_{ijks}: The mean value of $q_{ijks}(x,y)$;
- σ_{ijks}: The standard deviation of $q_{ijks}(x,y)$;
- m_{ijk}: The average of m_{ijks}, $s = 1,2,\ldots,16$;
- σ_{ijk}: The average of σ_{ijks}, $s = 1,2,\ldots,16$;
- v_{mijk}: The square root of the variance of m_{ijks}, $s = 1,2,\ldots,16$;
- $v_{\sigma ijk}$: The square root of the variance of σ_{ijks}, $s = 1,2,\ldots,16$;
- $\vec{F}_k = [f_{1k}\ f_{2k}\ \cdots\ f_{nk}\ \cdots\ f_{Nk}]^T$: The texture feature vector extracted for texture class k; $k = 1,2,\ldots,14$ and $N = 4x4x2 = 32$;

Now, we are at the position to present the mathematical description. We know, from the cortical channel model (6.32), that the output images from the symmetrical and anti-symmetrical Gabor filters are 2-D linear convolutions between the filter responses and the input image to the channel. For computational efficiency, 2-D linear convolutions are computed on the spatial frequency domain, i.e.,

$$\begin{aligned} a_{ijks}(x,y) &= F^{-1}[H_{cij}(u,v)P_{ks}(u,v)] \\ b_{ijks}(x,y) &= F^{-1}[H_{sij}(u,v)P_{ks}(u,v)] \end{aligned} \tag{6.34}$$

where $F^{-1}[\bullet]$ denotes inverse Fourier transform, $P_{ks}(u,v)$ is the Fourier transform of the input sample $p_{ks}(x,y)$, and $H_{cij}(u,v)$ and $H_{sij}(u,v)$ denote the transfer functions of the symmetrical and anti-symmetrical Gabor filters respectively. With $a_{ijks}(x,y)$ and $b_{ijks}(x,y)$ having been calculated, the channel output image is determined by

$$q_{ijks}(x,y) = \left[a^2_{ijks} + b^2_{ijks}\right]^{1/2} \tag{6.35}$$

Fig.6.11 displays the 16 channel output images for an input sample (shown in the first row) from texture class D57 (Handmade Paper).

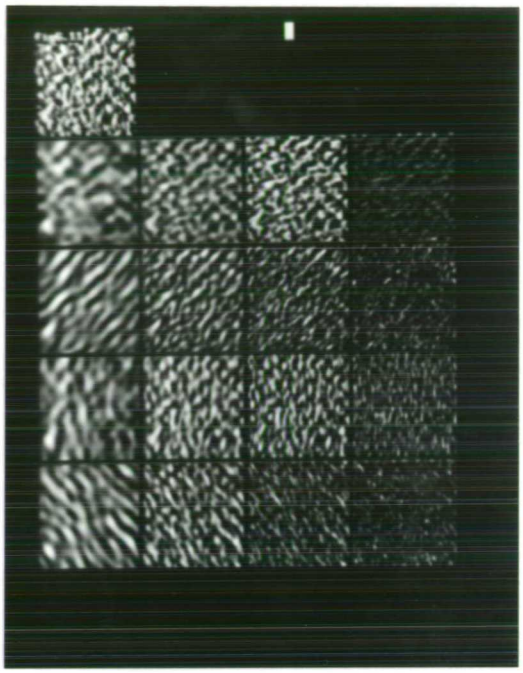


Figure 6.11. The 16 Channel Outputs Images for a D57 Sample.

The four images in the second row correspond to the four channels tuned to the same orientation $\theta_0 = 0^0$ whereas the images in Rows 3,4 and 5 correspond to channels tuned to orientations $\theta_0 = 45^0, 90^0$ and 135^0 respectively. Similarly, looking column-wise, the four images (excluding the input sample) shown in the first column correspond to channels tuned to the same radial frequency $\omega_0 = 4$ whereas the images in Columns 2,3 and 4 correspond to channels tuned to frequencies $\omega_0 = 8, 16$ and 32 respectively. Fig.6.12 shows the histograms of the images in Fig.6.11. Note, while the histogram of the input sample (top left corner) is roughly flat (as a result of histogram equalization), histograms of channel output images are all more or less Gaussian-like. After the channel output image $q_{ijkl}(x,y)$ has been obtained, the channel mean

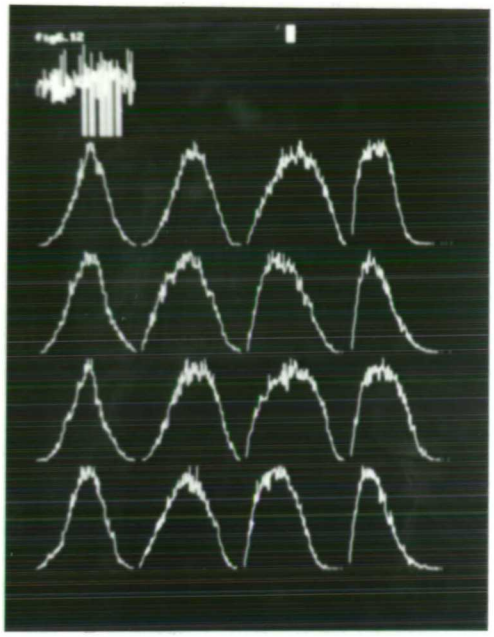


Figure 6.12. Histograms of the Images Shown in Fig.6.11.

and channel deviation are calculated by

$$m_{ijks} = \frac{1}{A}\iint_{\Psi} q_{ijks}(x,y)dxdy$$
$$\sigma_{ijks} = \frac{1}{A}\left[\iint_{\Psi}[q_{ijks}(x,y)-m_{ijks}]^2 dxdy\right]^{1/2} \tag{6.36}$$

where Ψ is the definition domain of $q_{ijks}(x,y)$ and A is the area of Ψ. After operations specified in (6.34) to (6.36) have been carried out for all samples belonging to texture class k, we compute the "representative" channel means and channel deviations as follows:

$$m_{ijk} = \frac{1}{16}\sum_{s=1}^{16} m_{ijks}$$
$$\sigma_{ijk} = \frac{1}{16}\sum_{s=1}^{16} \sigma_{ijks} \tag{6.37}$$

Finally, the "representative" feature vector extracted for texture class k is defined to be

$$\vec{F}_k=[m_{11k}\cdots m_{41k}\ m_{12k}\cdots m_{42k}\cdots m_{44k}\ \sigma_{11k}\cdots\sigma_{41k}\ \sigma_{12k}\cdots\sigma_{42k}\cdots\sigma_{44k}]^T \tag{6.38}$$

Note, the first 16 elements of $\vec{F}_k$ are the channel means from the 16 channels as numbered in Fig.6.9 whereas the last 16 elements are the channel deviations from these channels.

To complete texture feature extraction for classifier training, the above procedure ((6.34)-(6.35)) is repeated for all texture classes. Fig.6.13(a) shows the output from the computer program which implements the above procedure for texture class D4 (Pressed Cork), while Fig.6.13(b) shows the results for texture class D5 (Expanded Mica).

6.4.3 Perceptual Interpretation of The Defined Features

It has been tacitly assumed in the texture feature extraction scheme described in the preceding paragraphs that a texture field and its rotated version are regarded as different textures. The texture features thus defined are rotation-variant, i.e., a texture field and its rotated version will result in two distinct feature vectors. For example, if a texture field $f(x,y)$, having a texture feature vector

$$\vec{F}=[m_{11}\cdots m_{41}\ m_{12}\cdots m_{42}\cdots m_{44}\ \sigma_{11}\cdots\sigma_{41}\ \sigma_{12}\cdots\sigma_{42}\cdots\sigma_{44}]^T$$

extracted according to the procedure described in Section 6.4.2, is 45^0-rotated anti-clockwise, the spectrum of $f(x,y)$ will also be 45^0-rotated anti-clockwise, and as a consequence the feature vector extracted from the rotated texture field will be

$$\vec{F}_r=[m_{14}\cdots m_{44}\ m_{11}\cdots m_{41}\cdots m_{13}\cdots m_{43}\ \sigma_{14}\cdots\sigma_{44}\ \sigma_{11}\cdots\sigma_{41}\cdots\sigma_{13}\cdots\sigma_{43}]^T$$

Clearly, $\vec{F}\neq\vec{F}_r$ as they are different permutations of the same elements. Rotation-invariant texture features may be defined based on the extracted rotation-variant features. For instance, if we define 4 channel means and 4 channel deviations associated with each of the four chosen frequencies as (See (6.36) as well):

	Sample Mean	Sample Variance
Channel Mean Features:		
Filter No. 1:	1.3730015573879e+02	2.6765113282819e-01
Filter No. 2:	1.4023719983749e+02	3.6359897866404e-01
Filter No. 3:	9.1427188632886e+01	5.6693846455792e-01
Filter No. 4:	3.0417417988867e+01	1.1240410206732e+00
Filter No. 5:	7.6918079641551e+01	9.9868761883793e-01
Filter No. 6:	8.4634247052471e+01	9.8970563538739e-01
Filter No. 7:	4.6365927829765e+01	1.1699702757026e+00
Filter No. 8:	1.0194912882565e+01	9.7578121192601e-01
Filter No. 9:	1.3709396718107e+02	2.7904470007494e-01
Filter No. 10:	1.3973262203535e+02	4.1321681084102e-01
Filter No. 11:	9.1084453093626e+01	5.1051558112912e-01
Filter No. 12:	3.1569214224070e+01	1.4916876535037e+00
Filter No. 13:	7.3585116372725e+01	3.2114132576960e-01
Filter No. 14:	7.6088923449768e+01	2.8882245696513e-01
Filter No. 15:	3.3502299536833e+01	5.5644714107090e-01
Filter No. 16:	6.8449492240014e+00	4.7214139649456e-01
Channel Deviation Features:		
Filter No. 1:	2.6905966027811e+01	1.8585137365153e+00
Filter No. 2:	4.6646368773215e+01	1.6892109950037e+00
Filter No. 3:	4.1839692197681e+01	8.6841870605974e-01
Filter No. 4:	1.6217713761806e+01	5.9159759631854e-01
Filter No. 5:	2.6789092565742e+01	1.8189419359875e+00
Filter No. 6:	3.8786857755071e+01	9.8477950875214e-01
Filter No. 7:	2.3042444689158e+01	5.2027673523316e-01
Filter No. 8:	5.8796707416285e+00	4.6717938076431e-01
Filter No. 9:	2.5966227998954e+01	1.7336152645441e+00
Filter No. 10:	4.5538621513235e+01	1.8411761040283e+00
Filter No. 11:	4.1356595977354e+01	6.1625945209069e-01
Filter No. 12:	1.6268474471213e+01	6.7522981409404e-01
Filter No. 13:	1.8737218760996e+01	1.2702504258558e+00
Filter No. 14:	2.8728212475909e+01	8.8666126567983e-01
Filter No. 15:	1.6477384159590e+01	3.7023413494464e-01
Filter No. 16:	3.9034319242809e+00	2.2429768847017e-01

(a)

(To be continued on the next page)

$$m_{ik} = \frac{1}{4}\sum_{j=1}^{4} m_{ijk}$$
$$\sigma_{ik} = \frac{1}{4}\sum_{j=1}^{4} \sigma_{ijk} \qquad (6.39)$$

Then, these features will be invariant to rotations of multiples of 45 . For arbitrary rotations, rotation-invariant texture features may be defined by implementing more channels at sufficiently small orientation steps (say 10^0) for each radial frequency. This will of course

	Sample Mean	Sample Variance
Channel Mean Features:		
Filter No. 1:	1.3925181540968e+02	3.4621598419467e-01
Filter No. 2:	1.3894165282512e+02	4.9788261209689e-01
Filter No. 3:	8.5489969142519e+01	5.4191840803755e-01
Filter No. 4:	2.1834471970439e+01	3.7572691062767e-01
Filter No. 5:	8.1752579026328e+01	1.0477536992083e+00
Filter No. 6:	7.8413381134405e+01	9.6885054759959e-01
Filter No. 7:	3.0434680150771e+01	7.9035301253233e-01
Filter No. 8:	3.9702204806757e+00	3.9201484792385e-01
Filter No. 9:	1.4166581773579e+02	5.7675092410194e-01
Filter No. 10:	1.4072741124981e+02	4.3361507986956e-01
Filter No. 11:	8.6911428417336e+01	6.0874481626815e-01
Filter No. 12:	2.2781504268520e+01	4.5522735706033e 01
Filter No. 13:	7.9088836193471e+01	8.9219747737168e-01
Filter No. 14:	7.6828731495513e+01	7.0823574879371e-01
Filter No. 15:	2.9552084003917e+01	5.5041281472094e-01
Filter No. 16:	3.8688038169242e+00	4.3547457244857e-01
Channel Deviation Features:		
Filter No. 1:	5.2147134898942e+01	2.2153776490731e+00
Filter No. 2:	6.5934316425215e+01	1.3540080593250e+00
Filter No. 3:	4.4657728084347e+01	4.3360401007796e-01
Filter No. 4:	1.2624072781824e+01	4.3546183516402e-01
Filter No. 5:	3.8331875379594e+01	1.3916903322327e+00
Filter No. 6:	4.0378631902029e+01	6.0765662991238e-01
Filter No. 7:	1.6811659630191e+01	4.0061365615816e-01
Filter No. 8:	2.1334882457099e+00	2.4322015654945e-01
Filter No. 9:	5.7093476939767e+01	2.9887901811299e+00
Filter No. 10:	6.8703037302229e+01	1.4063084024138e+00
Filter No. 11:	4.6486556537728e+01	4.3719603909756e-01
Filter No. 12:	1.3079027111568e+01	4.5328692143047e-01
Filter No. 13:	3.6236321113460e+01	1.2031463146413e+00
Filter No. 14:	3.9110919181141e+01	6.1506322845867e-01
Filter No. 15:	1.6188290346907e+01	1.9213342916300e-01
Filter No. 16:	2.0606271707930e+00	2.6337421171756e-01

(b)

Figure 6.13. Outputs from the Classifier Training Program for Texture Classes (a) Pressed Cork (D4) and (b) Expanded Mica (D5).

increase the computational burden.

Previous work [29] has shown that human beings are particularly sensitive to textural differences in coarseness (vs. fineness) and directionality/regularity (vs. randomness). For computational texture features, the ability of indicating these "subjective" texture features is desirable. Although texture features defined in this chapter do not have clear indications on

regularity, they are directly related to texture coarseness and directionality. As the energy spectra of coarse textures are more concentrated in low frequency bands, channel means from channels tuned to low frequencies will be relatively larger. In contrast, for fine textures, image energy will be more scattered around high frequency components, and as a consequence, channels tuned to high frequencies will give us relatively larger channel means. By comparing the channel means of texture classes D55 (the finest texture), and D5 and D65 (the two coarsest textures) in the database, this argument is found convincing. Similarly, for directional textures, image energy will be concentrated along the underlying orientation, and channels tuned to this orientation will result in relatively higher channel means than channels tuned to any other orientations. On the other hand, the energy spectra of random textures are usually not oriented making all channels give similar channels means. An example of oriented textures is D68 (Wood Grain). By simple comparison, we find channels tuned to 0^0 result in relatively larger channel means. This is expected since D68 is horizontally oriented.

6.4.4 Texture Classification Scheme

For the purpose of texture classification, 10 testing or "unknown" samples from each texture class are derived from the original database. Recalling 14 different classes have been chosen, the testing database contains 10x14=140 "unknown" samples. As we did in classifier training, each testing sample is pre-processed by means of histogram equalization to remove first-order statistic differences. To classify a given "unknown" sample, texture feature extraction is first performed by executing the set of operations specified in (6.34) to (6.36) with the elimination of the subscripts k and s. The resultant texture feature vector from the sample will be

$$\vec{F} = [m_{11} \cdots m_{41}\, m_{12} \cdots m_{42} \cdots m_{44}\, \sigma_{11} \cdots \sigma_{41}\, \sigma_{12} \cdots \sigma_{42} \cdots \sigma_{44}]^T \tag{6.40}$$

This vector is then fed into the classifier. The classifier classifies the sample into texture class k *iff* the following distance function is a minimum at k:

$$d(\vec{F},k) = \sum_{n=1}^{N} \frac{[f_n - f_{in}]^2}{v_{in}^2} \tag{6.41}$$

where f_n is the n^{th} component feature of feature vector $\vec{F}$, and f_{in} and v_{in} are the sample mean

and sample deviation of the n^{th} component feature of the "representative" feature vector $\vec{F}_k$ of texture class k respectively. Note, the above classifier is identical to the one used in Chapter 5 (See (5.5)).

6.4.5 Experimental Results

To investigate the textural discrimination capabilities of the defined texture features, a set of 7 classification experiments have been conducted according to the classification scheme outlined in the preceding paragraph. In the first experiment, all the channel means and channel deviations, i.e., 16 channel means plus 16 channel deviations, are used. This application of a total of 32 features from 16 channels specified in (6.33) gives us error-free classifications under all chosen texture classes. Table 6.1 presents the classification results in the form of classification confusion matrix.

Experiment II is devoted to the classification study using only channel means (16 features) whereas Experiment III to the classification study using only channel deviations (16 features as well). Tables 6.2 and 6.3 show the corresponding experimental results. The 16 channel means result in perfect classifications. On the other hand, the 16 channel deviations show slightly worse performance than the channel means as one misclassification out of 140 testing samples occurred. The misclassification happened between texture classes D29 and D9 as one "unknown" sample from D29 is labeled with D9. The better performance of channel means compared to channel deviations may be what we expected since channel means are directly related to channel output energy (See (6.34) to (6.36)).

In Experiments IV, V, VI and VII then, we focus our attention on investigating the texture discrimination capabilities of texture features computed from channels tuned to a particular radial frequency. For example, in Experiment IV, only features from channels tuned to $\omega_0 = 4$ are used. These features include 4 channel means and 4 channel deviations making the total number of texture features used be 8. As shown in Table 6.4, there are 6 misclassifications out of 140 testing samples. Two samples from D9 are misclassified into D29, two from D29 are mislabeled with D9, one from D24 is misclassified into D68, and one from class D68 is wrongly labeled with class D55. The average classification accuracy is about 95.7%. Similarly, in Experiments V, VI and VII, we use the 8 features from channels tuned to radial frequencies

Table 6.1

	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28						10									100%
D29							10								100%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92														10	100%

Table 6.2

	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28						10									100%
D29							10								100%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92														10	100%

Table 6.3

	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28						10									100%
D29			1				9								90%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92														10	100%

Table 6.1. Texture Classification Confusion Matrix by Using All 32 Gabor Filtering Features

Table 6.2. Texture Classification Confusion Matrix by Using the 16 Channel Means

Table 6.3. Texture Classification Confusion Matrix by Using the 16 Channel Deviations

ω – 8,16 and 32 respectively, and the classification results are tabulated in Tables 6.5, 6.6 and 6.7. It is very encouraging to note that with the 8 features from channels at ω_0 – 16, error-free classification is achieved under all texture classes (Table 6.6), and there are only 3 misclassifications in Experiment V (Table 6.5) and 5 misclassifications in Experiment VII

Table 6.4

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			8				2								80%
D19				10											100%
D24					9							1			90%
D28						10									100%
D29			2				8								80%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68									1			9			90%
D84													10		100%
D92														10	100%

Table 6.5

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				9										1	90%
D24					10										100%
D28						10									100%
D29							10								100%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84							2						8		80%
D92														10	100%

Table 6.6

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28						10									100%
D29							10								100%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92														10	100%

Table 6.7

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28				1		9									90%
D29			2				8								80%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92			2											8	80%

Table 6.4. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega_0 = 4$.

Table 6.5. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega = 8$.

Table 6.6. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega = 16$.

Table 6.7. Texture Classification Confusion Matrix by Using the 8 Channel Features at Frequency $\omega = 32$.

(Table 6.7).

A simple comparison between Tables 6.4, 6.5, 6.6 and 6.7 reveals that the radial frequencies at which the underlying channel features give increasingly higher classification accuracies are of the order ω_0 = 16,8,32 and 4. The average classification accuracies at these frequencies are 100%, 97.9%, 96.43% and 95.7% respectively. This order is intuitively reasonable since

1. textures are basically bandpass signals;
2. features from low frequency channels are more correlated with the first-order statistics like the average intensity of the input sample; and
3. features from high frequency channels are more "contaminated" by noise as high frequency components are more likely due to noise.

6.5 A Standard Technique for Texture Feature Extraction

It is common practice to compare a new technique with some standard technique. We intend to compare the proposed algorithm for texture feature extraction with the conventional benchmark for texture analysis approaches, i.e., the GLDM (Gray Level Dependence Matrix) method. These considerations warrant this section on texture feature extraction and classification based on GLDMs.

6.5.1 Texture Description by GLDM

The gray level dependence matrix method is based on the approximation of the second-order statistics, or precisely the second-order joint conditional probability density functions $f(i,j|d,\theta)$ under a given spatial constraint specified by the parameters (d,θ). For digital images, common values for direction constraint θ are multiples of 45^0 including 0^0, i.e., $0^0,45^0,90^0,135^0,180^0,225^0,270^0$ and 315^0 whereas the intersample distance constraint d can be anything in theory. As correlations between pixels are usually getting weaker and weaker with increasing d, one is usually interested in small ds (say $d=1$ or $d=2$) in practice. Each $f(i,j|d,\theta)$ is the probability of going from gray level i to gray level j at the given intersample distance d and the given direction θ. Conventionally, $f(i,j|d,\theta)$s are estimated based on a single sample image and are usually written in the matrix form

$$\Phi(i,j|d,\theta) = [f'(i,j|d,\theta)] \tag{6.42}$$

where $f'(i,j|d,\theta)$ is the estimate of the true joint probability density function $f(i,j|d,\theta)$. (6.42) justifies the name of **Gray Level Dependence Matrix (GLDM)**. If no differences are made between direction pairs 0^0 and 180^0, 45^0 and 225^0, 90^0 and 270^0, and 135^0 and 315^0, then the directions of interest are 0^0,45^0,90^0 and 135^0, and the resultant GLDMs would be symmetrical along the main diagonals. Let $I(k,l)$ denote a given digital image defined on a lattice $L_r x L_c$; one can calculate these symmetrical GLDMs as follows:

$$\begin{aligned}
\Phi(i,j|d,0^0) &= \#[((k,l),(m,n)) \in (L_r x L_c) x (L_r x L_c) \,|\, (k-m=0,\ |l-n|-d), \\
&\quad (I(k,l)=i,\ I(m,n)=j)] \\
\Phi(i,j|d,45^0) &= \#[((k,l),(m,n)) \in (L_r x L_c) x (L_r x L_c) \,|\, (k-m=d,\ l-n=-d) \\
&\quad or\ (k-m=-d,\ l-n=d),(I(k,l)=i,\ I(m,n)=j)] \\
\Phi(i,j|d,90^0) &= \#[((k,l),(m,n)) \in (L_r x L_c) x (L_r x L_c) \,|\, (|k-m|=d,\ l-n=0), \\
&\quad (I(k,l)=i,\ I(m,n)=j)] \\
\Phi(i,j|d,135^0) &= \#[((k,l),(m,n)) \in (L_r x L_c) x (L_r x L_c) \,|\, (k-m=d,\ l-n=d) \\
&\quad or\ (k-m=-d,\ l-n=-d),(I(k,l)=i,\ I(m,n)=j)]
\end{aligned} \tag{6.43}$$

where # denotes the number of elements in a set.

As the GLDMs themselves are not in a form suitable for texture analysis, a set of measures are usually extracted from them and defined as texture features. For instance, **Haralick** *et al.* [30] proposed 14 features from each GLDM. Nevertheless, only five features from the set are usually computed and used. These are as follows:

1) **Energy:**

$$E(d,\theta) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [\phi(i,j|d,\theta)]^2 \tag{6.44}$$

2) **Entropy:**

$$H(d,\theta) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \phi(i,j|d,\theta) \log[1+\phi(i,j|d,\theta)] \tag{6.45}$$

3) **Correlation:**

$$C(d,\theta) = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i-\mu_x)(j-\mu_y)\phi(i,j|d,\theta)}{\sigma_x \sigma_y} \tag{6.46}$$

4) **Local Homogeneity:**

$$L(d,\theta) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\frac{\phi(i,j|d,\theta)}{1+(i-j)^2} \tag{6.47}$$

5) **Inertia:**

$$I(d,\theta) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i-j)^2\phi(i,j|d,\theta) \tag{6.48}$$

where $\phi(i,j|d,\theta)$ is the $(i,j)^{th}$ element of the matrix $\Phi(i,j|d,\theta)$, N is the number of gray levels in $I(k,l)$, and finally where

$$\mu_x = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} i\phi(i,j|d,\theta)$$
$$\mu_y = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1} j\phi(i,j|d,\theta)$$
$$\sigma_x^2 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i-\mu_x)^2\phi(i,j|d,\theta)$$
$$\sigma_y^2 = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}(j-\mu_y)^2\phi(i,j|d,\theta)$$

Following previous work by others [31], we use two intersample distances $d=1$ and $d=2$. As for each d, there are four directions and thus four GLDMs, and for each GLDM five texture features are defined, the total number of features extracted from a given sample image will be $2x4x5-40$. In detail, there are 8 energy features, 8 entropy features, 8 correlation features, 8 local homogeneity features and 8 inertia features respectively. To obtain the "representative" GLDM feature vectors for each texture class, a similar classifier training procedure to what is described in Section 6.4.2 is followed.

6.5.2 Texture Classification Results

As we want to compare the proposed algorithm with the GLDM technique based on texture classification performance, a set of 8 classification experiments using GLDM features have been conducted. The same classification scheme outlined in Section 6.4.4 is applied. In the following, we shall summarize the experimental results.

In the first experiment, all 40 features are used. No misclassification has been found. The results are tabulated in Table 6.8. In Experiment II, only features associated with intersample distance $d=1$ are used. This feature set consists of 4 energy features, 4 entropy features, 4 correlation features, 4 local homogeneity features and 4 inertia features making a total of 20

features. Table 6.9 presents the classification results. One misclassification (of one D28 sample into class D65) is observed. Similarly, in Experiment III, only the 20 features from intersample distance $d=2$ are used with classification results being given in Table 6.10. One classification error occurs between classes D29 and D9. From Tables 6.9 and 6.10, one may claim that features from $d=1$ and $d=2$ have comparable texture discrimination abilities under the chosen database.

Then, in Experiment IV to Experiment VIII, the number of texture features used is kept as 8. In Experiment IV, only the 8 energy features associated with both intersample distances are employed. The classification results are shown in Table 6.11. Note, a total number of 36 classification errors are observed. The worst performance (classification accuracy=10%) is associated with class D9 where only one sample out of 10 testing samples is correctly classified. The average classification accuracy is 74.3%. Similar results are reported in Experiment V as shown in Table 6.12 using the 8 entropy features. The average classification accuracy is also 74.3%. In Experiment VI we use the 8 correlation features and the results are given Table 6.13. A total of 14 misclassifications are found giving an average accuracy of 90%. Table 6.14 shows the experimental results obtained in Experiment VII by using the 8 local homogeneity features. 4 classification errors are found in this case showing an average classification accuracy of 97.1%. Finally, Experiment VIII uses the 8 inertia features with the results given in Table 6.15. 6 misclassifications are reported resulting in an average classification accuracy of 95.7%.

To summarize, we find that the whole set of 40 GLDM features are powerful enough to achieve perfect classification. Features from $d=1$ and from $d=2$ perform comparably. Finally, with only 8 features being used, we find that the features giving decreasing classification accuracies are local homogeneity features, inertia features, correlation features and energy/entropy features. The individual accuracies are 97.1%, 95.7%, 90% and 74.3%/74.3% respectively.

6.6 The Proposed Algorithm versus The Standard Technique

In this section, comparisons between the proposed algorithm for texture feature extraction and the GLDM technique will be made. This will be done in terms of classification performance, computational considerations, noise performance and other considerations.

Table 6.8

TEXTURE / TEXTURE	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28						10									100%
D29							10								100%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92														10	100%

Table 6.9

TEXTURE / TEXTURE	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28						9					1				90%
D29							10								100%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92														10	100%

Table 6.10

TEXTURE / TEXTURE	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28						10									100%
D29			1				9								90%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92														10	100%

Table 6.11

TEXTURE / TEXTURE	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	6		4												60%
D5	1	7							2						70%
D9	1		1				8								10%
D19				10											100%
D24					10										100%
D28	4					5			1						50%
D29			3				7								70%
D53								10							100%
D55									7		3				70%
D57										10					100%
D65									2		8				80%
D68									1			9			90%
D84													10		100%
D92		1			5									4	40%

Table 6.8. Texture Classification Confusion Matrix by Using All 40 GLDM Features at d=1 and 2.

Table 6.9. Texture Classification Confusion Matrix by Using All 20 GLDM Features at d=1.

Table 6.10. Texture Classification Confusion Matrix by Using All 20 GLDM Features at d 2.

Table 6.11. Texture Classification Confusion Matrix by Using the 8 Energy Measures at d−1 and 2.

Table 6.12

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	6		4												60%
D5		7				1			1						70%
D9	1		1				8								10%
D19				10											100%
D24					10										100%
D28	4	1				5									50%
D29			3				7								70%
D53								10							100%
D55									7	3					70%
D57										10					100%
D65									2		8				80%
D68									1			9			90%
D84													10		100%
D92		4				3								3	30%

Table 6.13

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												00%
D19				10											100%
D24				1	9										90%
D28				2		8									80%
D29			2				8								80%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68								2				8			80%
D84			2										8		80%
D92			2	3										5	50%

Table 6.14

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24			1		9										90%
D28		1				9									90%
D29							9						1		90%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84													10		100%
D92			1											9	90%

Table 6.15

Texture / Texture	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92	Accuracy
D4	10														100%
D5		10													100%
D9			10												100%
D19				10											100%
D24					10										100%
D28				1		9									90%
D29			3				7								70%
D53								10							100%
D55									10						100%
D57										10					100%
D65											10				100%
D68												10			100%
D84			2										8		80%
D92														10	100%

Table 6.12. Texture Classification Confusion Matrix by Using the 8 Entropy Measures at d 1 and 2.

Table 6.13. Texture Classification Confusion Matrix by Using the 8 Correlation Measures at d−1 and 2.

Table 6.14. Texture Classification Confusion Matrix by Using the 8 Local Homogeneity Measures at d 1 and 2.

Table 6.15. Texture Classification Confusion Matrix by Using the 8 Inertia Measures at d−1 and 2.

6.6.1 Performance Comparison

In general, classification performance comparison between different techniques for texture analysis is difficult. The difficulty is due to many factors such as the training and testing database used, the number of training and testing samples involved, the pre-processing steps involved, the number of texture features used and the particular type of classifier applied. Since these factors have been kept identical for the proposed algorithm and the GLDM technique in this chapter, comparison between them is possible and sensible.

With both techniques, perfect classification is achieved in some cases. With the proposed features, error-free classification is obtained in three cases, i.e., using the whole set of 32 features defined (16 channel means and 16 channel deviations), the 16 channel means only and using only the 8 features (4 channel means plus 4 channel deviations) from channels at $\omega_0 = 16$ (See Tables 6.1, 6.2 and 6.6). However, error-free classification is only achieved by using the whole set of 40 features in the standard technique (See Table 6.8). In addition, one error in classification is obtained in the standard technique by using all the 20 features from intersample distance $d=1$ or from $d=2$ (See Tables 6.9 and 6.10) whereas in the proposed algorithm, the same performance is achieved using only 16 channel deviations (See Table 6.3).

To make our comparison more sensible, let us focus our attention on cases where the number of texture features used is identical. As we discussed in Section 6.4.5, four experiments have been conducted using only 8 proposed features in each experiment. The experimental results are tabulated in Table 6.4 to Table 6.7. Similarly, for the standard technique, a set of five experiments have been performed using the same number of features. The corresponding experimental results are given in Table 6.11 to Table 6.15. From these tables, it is clear that the best accuracy in the proposed algorithm is 100% whereas it is 97.1% in the standard technique. Considering the classification accuracy the other way around, one finds that while the poorest performance in the proposed algorithm is 95.7%, it is 74.3% in the standard technique.

These simple comparisons make it evident that the proposed texture features are more powerful than the standard GLDM features in texture description and discrimination.

6.6.2 Computational Considerations

In addition to the superior classification performance, the proposed algorithm also has

computational advantages over the standard technique. Referring to Fig.6.10, one notices that the proposed algorithm is highly parallel in structure. Thus the computational efficiency can be much improved by parallel implementation. Furthermore, Gabor filtering can be dramatically accelerated via FFT techniques as we did in Section 6.4.4. This is an important consideration as FFT chips are now readily available on the market. On the contrary, it is well-known that the GLDM method is computationally rather demanding [31]. Its computational burden increases sharply as the number of gray levels and the size of a given image increase.

6.6.3 Noise Performance

In realistic applications, texture samples to be classified may have been contaminated by some types of noise, and a good set of texture features should be relatively insensitive to noise distortions. We claim that the proposed texture features, which may be called Gabor Filtering (GF) features, perform much better than GLDM features under noisy environments. This is clearly demonstrated in Fig.6.14. The figure is obtained by conducting a series of texture classification experiments under the same testing database used in Sections 6.4.5 and 6.5.2 except that a certain amount of noise is added to each "unknown" sample. The noise level is measured in terms of the signal-to-noise ratio (SNR) in dbs. The figure shows the classification accuracies by using all the 32 GF features (solid line) and by using all the 40 GLDM features (broken line) under different noise levels such as 5db, 15db, etc.. From the figure, the following three observations are significant:

1. GF features always perform better than GLDM features in terms of classification accuracy;
2. There is a *threshold* for noise level below which the classification accuracy is constantly low (7.14%). The threshold for GF features is much smaller than that for GLDM features (3.8db vs. 7.5db); and
3. Compared with GLDM features, the GF features are relatively insensitive to noise. At SNR level=20db or above, the accuracy of GF features is constantly 100% whereas the accuracy for GLDM features is around 60%.

6.6.4 Other Considerations

Another drawback of the GLDM method is the empirical choice of the intersample

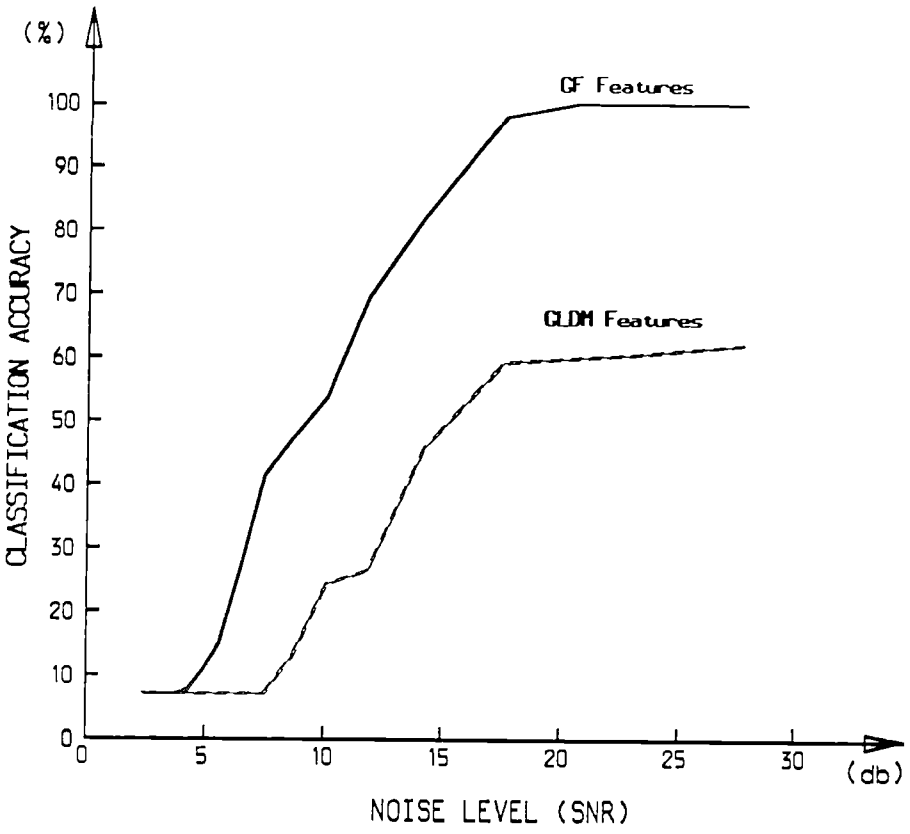


Figure 6.14. The Noise Performance of the Proposed (GF) and the Classical (GLDM) Texture Features.

distance d. Intuitively, for coarse textures, a large d is required in order to capture intrinsic textural information. Similarly for fine textures, a small intersample distance d should be used. This implies that the choice of a pertinent intersample distance d to calculate GLDMs is texture-dependent, and inappropriate choice of d could make the resultant GLDMs completely useless for texture description. Problems of this kind clearly do not exist in the proposed algorithm.

The remarkable effectiveness of the HVS in texture discrimination is partially due to, we guess, its relative sample-invariance when performing texture feature extraction. By sample-invariance, we mean texture features extracted from different samples belonging to the same texture class are very close to each other. These arguments make us reason that relative sample-invariance should be a desirable feature of computational algorithms for texture feature extraction. The relative sample-invariance may be measured by means of the ratio of the

standard deviation of sample features to the mean value of these sample features. Fig.6.15 shows these statistics for both the proposed algorithm and the GLDM method under each of the chosen texture classes. Apparently, the proposed features (Gabor filtering features) show better sample-invariance than the GLDM features.

Texture Feature Statistics: $\frac{\text{Sample Std. Dev.}}{\text{Sample Mean}} \times 100\%$

(%)

Texture Class / Feature Type	D4	D5	D9	D19	D24	D28	D29	D53	D55	D57	D65	D68	D84	D92
Gabor Filtering Features	3.265	3.023	3.667	3.217	3.206	2.533	3.614	3.444	3.328	3.119	2.308	3 693	2.215	2.942
GLDM Features	4.311	6.635	4.522	5.795	3.303	4. 167	4.596	2.777	5.393	2.808	4.815	5.904	2.984	4. 162

Figure 6.15. The Deviation-to-Mean Ratios of the Proposed (GF) and the Classical (GLDM) Texture Features.

6.7 Conclusions

Very recently, a great deal of research interest within the image analysis community lies in the application of human visual models in computational texture analysis. This research direction is natural as the Human Visual System is by far the most effective and efficient system for texture analysis. The progress in this direction clearly depends on how much we understand about the HVS. For the last two decades, convincing evidence from the vision perception, physiology and psychophysics research communities has confirmed the hypothesis that the HVS (mainly in the visual cortex, an important station on the visual pathway) processes the input pictorial information via a set of parallel and quasi-independent frequency channels. This hypothesis has motivated a general scheme for texture analysis which we called the **Multichannel Spatial Filtering Model (MSFM)** approach.

Overall, the algorithm presented in this chapter may be classified as a MSFM approach. It possesses a highly parallel structure well suitable for parallel implementation. The algorithm involves a set of channels each of which is tuned to a specific band of spatial frequency and orientation. To determine the appropriate number of channels to be implemented together with

channel parameters (i.e., central frequency and bandwidth), the frequency component resolution and energy processing characteristics of the HVS have been investigated in the context of visual texture discrimination. This investigation enabled us to determine the filtering pattern adopted for texture feature extraction. After outlining the principles of the proposed algorithm, both the texture feature extraction scheme (classifier training) and the texture classification scheme are subsequently detailed. A series of classification experiments have been conducted to test the effectiveness of the proposed texture features for texture description and discrimination.

To make our experimental results more convincing, the proposed algorithm is compared with the well-known standard technique for texture analysis, the **Gray Level Dependence Matrix (GLDM)** method. Comparisons between the two techniques are also made in terms of computational efficiency, noise performance and the sample-invariance of the corresponding texture features. These comparisons lead us to conclude this chapter with the following remarks:

- **Generality:** The proposed algorithm is applicable under all types of textures (microtextures or macrotextures, stochastic or structural, etc.);
- **Superiority:** The proposed algorithm shows superior classification performance to the standard GLDM technique;
- **Efficiency:** The proposed algorithm is highly parallel in structure and thus well suited for parallel implementation;
- **Robustness:** The proposed texture features show a better noise performance than the GLDM features; and
- **Invariance:** The proposed algorithm has a good sample-invariance characteristic as the features extracted from different samples of the same texture class are very close to each other.

6.8 References

[1] **K.K.DeValois**, *Spatial Vision*, Annual Review of Physiology, 1984.

[2] **A.Ikonomopoulos** and **M.Unser**, *A Directional Filtering Approach to Texture Discrimination*, Proc. of the Inter. Conf. on Pattern Recognition, Montreal, Canada, July

30-Aug. 2 1984, pp.80-82.

[3] **D.Wermser** and **C.E.Leidtke**, *Texture Analysis Using A Model of Visual System*, Proc. of the Inter. Conf. on Pattern Recognition, Munich, 1982, pp.1078-1080.

[4] **G.A.Wright** and **M.E.Jernigan**, *Spatial Frequency and Orientation Correlates of Textures in Human Vision*, Proc. of Inter. Conf. on Cybern. and Soc., Tucson, Arizona, Nov.12-15, 1985, pp.697-700.

[5] **G.A.Wright** and **M.E.Jernigan**, *Texture Discrimination from Spatial Frequency Channels*, Proc. of Inter. Conf. on Syst. Man and Cybern., Atlanta, GA, USA, Oct.14-17, 1986, pp.519-524.

[6] **J.G.Daugman**, *Spatial Visual Channels in the Fourier Plane*, Vision Research, vol.24, 1984, pp.891-910.

[7] **D.A.Pollen** and **S.F.Ronner**, *Visual Cortical Neurons as Localized Spatial Frequency Filters*, IEEE Trans. on Syst. Man and Cybern., vol.13, 1983, pp.907-916.

[8] **T.N.Tan**, *Texture Edge Detection Via Gabor Filtering*, Internal Report, TR-08-TNT, SPABSS/EE/ICST, November 1988.

[9] **D.Gabor**, *Theory of Communication*, J.Inst.Elec.Eng., vol.93, part III, 1946, pp.429-459.

[10] **R.N.Bracewell**, *The Fourier Transform and Its Applications*, McGraw-Hill, New York, 1978.

[11] **J.G.Daugman**, *Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters*, J Opt Soc Am A, vol.2, 1985, pp.1160-1169.

[12] **D.Marr**, *Vision*, W.H.Freeman, 1982.

[13] **D.H.Hubel** and **T.N.Wiesel**, *Receptive Fields, Binocular Interaction and Functional Architecture in The Cat's Visual Cortex*, J Physiol, vol.160, 1962, pp.106-154.

[14] **J.J.Kulikowski** and **P.O.Bishop**, *Fourier Analysis and Spatial Representation in The Visual Cortex*, Experientia, vol.37, 1981, pp.160-163.

[15] **F.W.Compbell** and **J.G.Robson**, *Application of Fourier Analysis to The Visibility of Gratings*, J Physiol, vol.197, 1968, pp.551-566.

[16] **S.Marcelja**, *Mathematical Description of the Responses of the Simple Cortical Cells*, J Opt Soc Am, vol.70, 1980, pp.1297-1300.

[17] **R.L.Devalois** *et al*, *Spatial Frequency Selectivity of Cells in Macaque Visual Cortex*, Vision Research, vol.22, 1982, pp.549-559.

[18] **D.A.Pollen** and **S.F.Ronner**, *Phase Relationship Between Adjacent Simple Cells in the Visual Cortex*, Science, vol.212, 1981, pp.1409-1411.

[19] **L.Maffei** and **A.Fiorentini**, *The Visual Cortex as A Spatial Frequency Analyzer*, Vision Research, vol.13, 1973, pp.1255-1267.

[20] **J.A.Movshon** and **D.J.Tolhurst**, *On The Response Linearity of Neurons in Cat Visual Cortex*, J Physiol, vol.249, 1975, pp.56p-57p.

[21] **B.W.Andrews** and **D.A.Pollen**, *Relationship Between Spatial Frequency Selectivity and Receptive Field Profile of Simple Cells*, J Physiol, vol.287, 1979, pp.163-176.

[22] **T.Caelli**, *Energy Processing and Coding Factors in Texture Discrimination and Image Processing*, Perception and Psychophysics, vol.34, 1983, pp.349-355.

[23] **P.Brodatz**, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, 1966.

[24] **T.Caelli** and **M.Hubner**, *Coding Images in the Frequency Domain: Filter Design and Energy Processing Characteristics of the Human Visual System*, IEEE Trans. on Syst. Man and Cybern., vol.13, 1983, pp.1018-1021.

[25] **J.J.Kulikowski** and **P.O.Bishop**, *Fourier Analysis and Spatial Representation in The Visual Cortex*, Experientia, vol.37, 1981, pp.160-163.

[26] **H.R.Wilson** and **J.R.Bergen**, *A Four Mechanism Model for Threshold Spatial Vision*, Vision Research, vol.19, 1979, pp.19-32.

[27] **L.O.Harvey** and **M.J.Gervais**, *Internal Representation of Visual Texture as The Basis for The Judgement of Similarity*, J. Exp. Psychol (Human Perception), vol.7, 1981, pp.741-753.

[28] **M.R.Turner**, *Texture Discrimination By Gabor Functions*, Biol Cybern, vol.55, 1986, pp.71-82.

[29] **H.Tamura** *et al.*, *Texture Features Corresponding to Visual Perception*, IEEE Trans. on Syst. Man and Cybern., vol.8, 1978,pp.460-472.

[30] **R.M.Haralick** *et al.*, *Textural Features for Image Classification*, IEEE Trans. on Syst. Man and Cybern., vol.3, 1973, pp.610-621.

[31] **L.Van Gool** *et al.*, *SURVEY: Texture Analysis Anno 1983*, Comput. Vision, Graphics and Image Proc., vol.29, 1985, pp.336-357.

7. TEXTURE SEGMENTATION: A REVIEW

7.1 Introduction

In the preceding chapters, our attention has been concentrated on the problem of texture feature extraction and classification. From this chapter on, our attention will be focused on another major concern of this thesis, the problem of texture segmentation. Texture segmentation is clearly a very difficult and challenging problem. Briefly stated, texture segmentation, or image segmentation by texture (to be differentiated from image segmentation based on intensity), is a problem of partitioning a given image into a number of disjointed constituent regions. These regions should have connected interiors, be homogeneous in texture and should not be expandable without infringing the homogeneity of region texture.

An image region can be delimited at two levels: it can be specified by its interior or it can be outlined by its contour or boundary. Accordingly, a general texture segmentation problem can be solved by means of two alternatives: *Uniform Texture Region Extraction* (UNITRE) and *Texture Edge Detection* (TEXTED). For the former, we shall classify the methods found in the open literature into five categories: feature space clustering (FSC), statistical classification (STC), generalization of intensity segmentation methods (GIS), state estimation approaches (SEA) and spatial filtering approaches (SFA). For the latter, we shall distinguish the existing methods into texture gradient operator (TGO), the spatial filtering technique (SFT), the theoretical estimation approach (TEA) and the texton-based method (TBM). Depending on the amount of *a priori* knowledge required on input images, these methods may be alternatively distinguished into two major groups: *unsupervised texture segmentation (UTS)* methods and *supervised texture segmentation (STS)* methods. Unsupervised texture segmentation requires no *a priori* information on the number and type of textures which may be contained in the given image whereas this kind of information is usually essential in supervised texture segmentation.

In this chapter, we shall outline these methods by summarizing their basic principles, describing appropriate examples and by pointing out their relative merits and problems.

7.2 Uniform Texture Region Extraction

7.2.1 FSC: Feature Space Clustering

It is well known that clustering techniques including thresholding are useful for gray level image segmentation [4,5]. If the gray intensity property is replaced by some texture features, texture segmentation may also be achievable via these techniques. The general FSC texture segmentation category can be further divided into Texture Feature Histogram Thresholding and N-D Texture Feature Space Clustering depending on the number of texture features used (1 or N features). Since texture property is a regional image characteristic, windows of appropriate size have to be first defined in which to calculate texture features. What type of texture features to use may be problem-dependent. In principle, any features useful for texture classification can be employed. Examples are gray histogram features, co-occurrence matrix features, edge features and features derived from some texture models. Because of the self-similarity property of textures, windows from the same texture region are expected to possess similar texture features. After feature extraction is performed for all windows, each window is then represented by the extracted feature vector. Now, if a dimension is defined for each of the N features extracted, each window will be seen as a point in the resultant N-D feature space. In this way, windows from the same texture region will be "close" to each other in such a space, and windows from differently textured regions will form distinct clusters. After such a feature space is constructed from a given textured image, one has to detect the number of major clusters and to effectively separate them in order to facilitate subsequent operations. Mapping the separated clusters back into the spatial image domain results in the identification of uniform texture regions, i.e., the extraction of uniform texture regions. Note, a single cluster may be mapped into several disjointed regions which have identical textures. The procedure just outlined may be called the **N-D Texture Feature Space Clustering** technique.

For some applications, a single texture feature may be adequate. In this case, the N-D feature space degenerates into a 1-D feature space, which is usually viewed as a *feature histogram*. When the computed window texture features are histogramized, the result will be a multimoded histogram with each mode indicating a texture class. Detecting these modes and thresholding the windows at appropriate feature values will give one a segmentation of the input image on the basis of the underlying texture feature. For distinguishing, we call this 1-D approach the **Texture Feature Histogram Thresholding** technique.

In [6], **Wang** et al. define a number of orientation-sensitive local windows such as horizontal window, vertical window etc.. In each of these windows, the standard deviations of gray intensities are calculated and histogramized. The histograms exhibit some distinct peaks corresponding to some uniform texture regions. Thresholding at these peaks enables them to obtain a reasonable segmentation of two textured images. Similar thresholding techniques are applied in [7] and [8]. In [7], **Zucker** *et al.* used local spot texture features whereas in [8] **Mitchell** *et al.* used local extrema texture measures.

Examples of the N-D texture feature space clustering technique are [9] and [10]. In [9], **Kashyap** and **Khotanzad** addressed the problem of texture segmentation by means of a spatial interaction texture model. They divided the image into non-overlapped windows and fit the assumed model to these windows. Texture features were then derived from the estimates of the model fitting parameters within each window. Different pairs of features were plotted in a 2-D space. They then chose the plot which has the most separated clusters and defined the decision boundaries on the clusters in the 2-D feature space. Finally, they mapped the detected clusters back to the spatial image domain to obtain the final texture segmentation. A similar procedure is suggested by **Wechsler** [10] with texture features derived from random walk models.

The FSC technique usually achieves UTS but suffers from a number of problems. First of all, the spatial contiguity (proximity) has not been taken into account when constructing the feature space and mapping clusters back into the image domain. Points in the same cluster do not necessarily come from spatially adjacent windows and the final segmentation is inevitably spotty (noisy). This problem may be overcome by incorporating the spatial coordinates of pixels into the feature space but this has not been shown to be a good solution [1]. In addition, the workability of FSC is highly dependent on the separability of clusters, which can be greatly influenced by the effectiveness of texture features chosen and the size of windows.

7.2.2 STC: Statistical Classification

As is reviewed in Chapter 2, numerous work has been done in the area of texture feature extraction and classification. Thus it is natural for one to think of ways to extend these classification techniques to images consisting of multiple textures, i.e, texture segmentation. These considerations lead to the development of the STC technique for texture segmentation. The general principle of this category may be stated as follows. In order to label each pixel in a

given textured image, one has to first associate with each pixel some texture measures. As such, the first step is to extract some texture measures or a texture feature vector at each pixel. The feature vector is subsequently attached to the pixel. After this operation is carried out over the entire image, all pixels are then specified by the computed feature vectors. Treating these vectors as patterns, one can then employ some standard pattern classifiers to classify each pixel into one of the known patterns. After this pixel labeling operation, pixels in a uniform texture region are expected to possess identical labels whereas pixels from differently textured regions distinct labels, so achieving uniform texture region extraction. Two alternatives exist for the assignment of feature vectors to pixels. One may use a *sliding* window, compute features within it, and then assign the feature vector obtained to the central pixel of the window. This is repeated at every pixel. Alternatively, one may first divide the given image into non-overlapped windows or blocks of some size, extract a feature vector from the window and then assign the feature vector to the entire window, i.e., to all the pixels in the window. Apparently, the former procedure is computationally more expensive than the latter but it is capable of finding more accurate region boundaries.

Deguchi and **Morishita** [11] adopted the pixel-based statistical classification technique. Texture features are derived from a 2-D linear prediction model (similar to an AR model). By assuming the input image only contains two textures, an appropriate model is first determined for one of the textures. They then used this model to fit the entire image field and calculated the fitting errors at each pixel. Finally, some statistics are defined based on the fitting errors at each pixel and used as the input to a Bayesian classifier. When the input image is composed of more than two textures, they developed a region growing algorithm based on this basic procedure. The pixel-based statistical classification technique is also followed by **Barba** and **Ronsin** [12]. In this paper, they extracted a feature vector at each pixel based on the so-called curvilinear integration operation. Other examples of this category are [13] and [14].

In [15], **Conners** *et al.* employed the block-based statistical classification technique for the segmentation of a high-resolution urban scene. Given such an image, they attempted to segment the scene into meaningful regions such as Residential, Commercial/Industrial, Mobile home and Water etc.. The input image is first partitioned into subregions and co-occurrence matrices are formulated in these regions. Then, six texture features namely Inertia, Cluster Shade, Cluster Prominence, Local Homogeneity, Energy and Entropy are derived from these matrices forming

a feature vector for the corresponding region. Finally, these regions are labeled with one of the known nine texture patterns based on the computed texture features. In [3], **Laws** first processed a given image by means of local linear masking and calculated texture energy measures within small image blocks. To obtain texture segmentation, he then proceeded with block classification based on the extracted texture measures. Similar operations are performed in [2].

The STC technique is usually useful for supervised texture segmentation as the number and type of textures contained in the input image have to be specified by the user. As in the case of the FSC technique, the STC technique also suffers from neglecting the spatial contiguity and thus producing noisy segmentations. Moreover, the segmentation accuracy is also very much dependent on the type of features used and the size of windows.

The STC technique may also be incorporated into unsupervised texture segmentation. Such an algorithm is presented by **Michel** *et al.* [37]. The input image is first convolved with Hadamard masks of size 3x3 and texture-energy-type features are computed within small regions. By comparing texture features in contiguous regions, texturally homogeneous and inhomogeneous portions are identified where inhomogeneous portions are regions containing texture boundaries. Then the detected homogeneous regions are used as training samples or prototypes for the classification of inhomogeneous regions in order to refine texture boundaries.

7.2.3 SEA: State Estimation Approach

Sometimes it may be desirable that some criteria associated with the problem of texture segmentation should be optimized. The SEA technique is of this type. To introduce the technique, let us first define the following notations. Assume the input textured image $f(i,j)$ consists of q exclusive and exhaustive regions each of which has an unique texture, and represent these regions by $A_1, A_2, \ldots, A_q$, then for an ideal segmentation we should have

- *Exhaustiveness:* $\bigcup_{k=1}^{q} A_k = f(i,j)$;
- *Exclusiveness:* $A_k \bigcap A_l = \phi, \quad k,l = 1,2, \ldots, q$ and $k \neq l$;

where ϕ is an empty set. If we define a state S associated with every texture region A_S, $S \in [1,2,\ldots,q]$, then each pixel in $f(i,j)$ has one of the q states. Bearing these in mind, we can now view the problem of segmenting $f(i,j)$ by texture as a state estimation problem at each pixel. It is not difficult to see that uniform texture region A_S is the union of pixels with state S. For optimal estimation of states at each pixel or estimation of regions $A_1, A_2, \ldots, A_q$, one attempts to maximize the *posterior* probability $P[A_1, A_2, \ldots, A_q | f(i,j)]$. Suppose $f(i,j)$ is of size MxN and Let y be the $MNx1$ vector obtained by lexicographically concatenating $f(i,j)$, then using the Bayes rule, one can rewrite the posterior probability as

$$P[A_1, A_2, \ldots, A_q | f(i,j)] = \frac{P[y | A_1, A_2, \ldots, A_q] P[A_1, A_2, \ldots, A_q]}{P(y)}$$

Since $P(y)$ is not related to $A_1, A_2, \ldots, A_q$, one looks for a MAP texture segmentation by maximizing $P[y | A_1, A_2, \ldots, A_q] P[A_1, A_2, \ldots, A_q]$. If no *a priori* knowledge is available about the statistics of $A_1, A_2, \ldots, A_q$, one then seeks for a ML texture segmentation by optimizing $P[y | A_1, A_2, \ldots, A_q]$. Different assumptions about the models for $f(i,j)$ and $A_1, A_2, \ldots, A_q$ will yield different segmentations. Some appropriate choices of these models will enable one to employ some conventional optimal linear filters such as Kalman Filters to carry out the state estimation problem. From the above formulation of and solution to the texture segmentation problem, one may already realize that the SEA technique is necessarily of high computational complexity. In practice, one is forced to seek for some sub-optimal solutions to the problem.

Based on Gibbs random field models, **Derin** and **Cole** [16] developed algorithms for segmenting some artificial textured images. They assumed that data in uniform texture regions can be modeled by one of the known Gibbs random field models. Each model is specified by several model parameters. They model the pixel states or region transitions also by a Gibbs random field model forming a so-called *doubly stochastic Gibbs random field model*. The posterior probability is formulated in such an way that the optimization is performed via Bellman's dynamic programming [38] for MAP or ML texture segmentation. Because of the impractical computational complexity of the algorithms, images are limited to only contain a few textures with a small number of gray levels, and segmentation is carried out on a strip-by-strip basis. This work is further extended to segmentation of noisy and textured images by **Derin** and **Elliott** [17].

The SEA technique has also been applied to the segmentation of terrain images. Results are summarized in [18]. In this paper, **Therrien** modeled region textures by 2-D linear filtering models. Superimposed on this is a simple Markov random field model that describes the state transition from regions of one terrain type to another. MAP as well as ML segmentation has been discussed with MAP giving better results. Similar work has been reported by **Cohen** and **Cooper** [19], and **Fan** and **Cohen** [39].

The SEA technique is of interest primarily for supervised texture segmentation as *a priori* information on the characteristics of region textures (say model parameters) is usually required. The main drawback of SEA is its excessive requirements on computation and memory.

7.2.4 GIS: Generalization of Intensity Segmentation Methods

A number of techniques for gray image segmentation have been successfully used in practice [4]. Some of these methods can be generalized to texture segmentation. As the gray intensity property is definable at a single pixel whereas the texture property is a local regional characteristic, the first thing to do is to convert a textured image into an image to which some conventional intensity segmentation techniques can be directly applied. Such a conversion can be achieved in a number of ways. One way is to use a sliding window as one does in the STC technique. At each pixel, texture feature extraction is performed within the sliding window and the computed texture measure or an appropriate function of several such measures is defined as the "intensity" of the central pixel in the transformed image. In this way, the input textured image is converted into a feature image of the same size as the original textured image. Segmentation of this feature image on the basis of intensity will generate a segmentation of the original image by texture. Alternatively, one may use non-overlapped windows of predefined size. Each of these windows is then represented by a single pixel in the transformed image and its intensity can be defined to be the texture measure or a function of the measures calculated within the window. If the input image is of size MxN and the window mxn, then this procedure generates a feature image of size $\frac{M}{m}x\frac{N}{n}$.

Chen and **Pavlidis** [20] extended their Split-and-Merge gray image segmentation technique to texture segmentation. They first partitioned the input image into small regions. Co-occurrence matrices are then formulated in these regions. Region-splitting and region-merging

are controlled by the similarity test based on the calculated matrices. After these steps, a preliminary segmentation is obtained. They then proceeded with a grouping algorithm using the co-occurrence matrices and region adjacency graphs (RAG), plus a small region elimination process to achieve the final segmentation. Similar work is reported by the same authors in [21] using correlation features.

Pyramid segmentation techniques have also been applied in texture segmentation. Examples are **Pietikainen and Rosenfeld** [22], and **Kjell and Dyer** [23]. In [22], the algorithm starts with an initial decomposition of the input image into small blocks of size 8x8, the contrast measure from co-occurrence matrices is computed within each block yielding an array of property values, i.e., a feature image of size $\frac{M}{8}x\frac{N}{8}$. Then a pyramid node linking strategy (see [26]) is applied to this array producing a texture segmentation of the input image. Median filtering is iteratively used to improve the segmentation performance. In [23], Kjell and Dyer used the technique in [26] for texture segmentation based on the oriented edge separation texture features.

Another major category of image segmentation approaches, the region growing techniques, is also illustrated in texture segmentation by **Raafat and Wong** [24]. The segmentation process is directed by textural information inherent in image regions. Based on a new texture metric [40], a textural dissimilarity measure between pairs of windows is defined. A resolution dependent texture information measure is then used to initiate uniform texture regions and together with the texture dissimilarity measure directs the region growing process. The same methodology is also followed in [41].

Before concluding this sub-section, several remarks are in order. First, special attention needs to be paid to the selection of the windows within which to extract texture measures. Second, the weakness of the basic gray image segmentation methods used will certainly be reflected in the results of texture segmentation. For instance, they may yield jagged and somehow artificial region boundaries. Finally, during the conversion from a textured image to a gray image, only windows of a single size are usually used. The application of multiple window sizes will improve the segmentation results as shown by **Weber and Sawchuk** [25].

7.2.5 SFA: Spatial Filtering Approach

The SFA technique attempts to extract uniform texture regions by direct spatial filtering of input images, or to transform input textured images into non-textured intensity images plus supplementary refining operations. The decisive step in SFA is no doubt the selection of appropriate spatial filters (linear or nonlinear). Because of the huge number of attributes associated with texture characteristics, the general design of such filters is practically impossible. Nevertheless, linear filters of this kind may be obtainable under certain special circumstances. For example, if the textures in the image to be segmented differ mainly in the orientation characteristics, then one may apply a set of filters which are highly sensitive to these orientation differences. Similarly, if the spatial frequency or coarseness attribute is a predominant discriminating feature in the textures, one then can choose filters which are "tuned" to specific frequency bands. Concerned with linear spatial filtering, the *uncertainty principle* [27] should also be recalled. This principle limits the resolution simultaneously attainable in the spatial and spectral domains. For these reasons, linear filters used in texture segmentation have been mainly confined to be members of a family of the so-called *Gabor* filters because they possess some desirable properties (See Chapter 6).

Turner [28] applied a set of Gabor filters to discriminate some artificial textures. This set of filters is generated by varying certain filter parameters such as orientation, frequency bandwidth and initial phase. The set contains 4 frequencies, 4 orientations and one pair of quadrature phases for each orientation and frequency combination, yielding a total of 32 filters. The author found that these filters are capable of reducing higher-order texture statistics to lower-order statistics such as means and variances. It is also found that in order to produce good discrimination/segmentation, successive Gabor filering may be necessary. This work is extended to some natural texture segmentation by **Clark** *et al.* [29]. Assuming textures differ from each other in orientations and spatial frequencies, they choose by visual inspection a set of appropriate Gabor filters tuned to these texture characteristics, and convolve these 2-D filters with the input textured images generating images which can then be processed by some simple operations to get the final texture segmentation.

In addition to Gabor filters, **Bashir** [42] suggested the design of linear filters for texture segmentation based on image energy spectra. For a particular texture, he placed ideal

rectangular filters onto those frequency bands which show maximum disparity in energy compared with the corresponding bands of other textures in the training set. Implementable filters are then obtained by approximating the ideal filters.

Texture segmentation based on nonlinear spatial filtering is presented by **Richards** in [43]. The algorithm is essentially based on the modification of the Fourier transform of a given input image. To extract uniform texture regions, some parts of the spectrum, which are assumed to represent coherently linked structures, are either inhibited or enhanced by applying what he called the *rejection filter* or the *injection filter*. These filters are basically exponential functions with the time constant proportional to the power spectrum.

The UTS algorithm proposed by **Reed** and **Wechsler** [45] can be regarded as the SFA technique in a broader sense. The algorithm consists of four major steps, i.e., anti-alias filtering, pseudo-Wigner distribution (PWD) calculation, selection of the most significant frequency components in the PWD and relaxation on the primary frequency plane.

In summary, the SFA technique is relatively recent. Most of the current methods in this category are *ad hoc* in nature, and the implementation results reported are far from satisfactory. More investigation is worthwhile. For more details, see Chapter 9 of this thesis.

7.3 Texture Edge Detection

7.3.1 TGO: Texture Gradient Operator

Conventional gradient operators such as the Sobel operator and Roberts operator [5], which attempt to detect discontinuities in intensity, are usually incapable of locating boundaries between differently textured regions. As texture, unlike intensity, is a local regional image property, discontinuities in local texture properties defined over some local neighborhoods should be used. Extraction of texture edges can, in principle, be obtained after enhancement by a texture dissimilarity sensitive operator or a texture gradient operator followed by thresholding or nonmaxima suppression. As is the case in gray intensity edge detection, post-processing steps such as line thinning and line tracking may be required to obtain connected texture edges of single pixel width.

In [33], **Wermser** and **Liedtke**, based on a model of the HVS, derived a measure from the amount of local texture changes, and called such a measure the *texture gradient* measure. After

texture feature extraction and gradient computation from the input image, they produced a texture gradient image. Simple thresholding on this gradient image locates major texture boundaries.

In practice, it is unlikely that a single texture feature would be adequate for effective texture edge detection. More frequently, one has to apply multiple texture features and be able to find ways of incorporating these features into a single compound measure. This is shown in a paper by **Grinaker** [34]. Some general principles are given in this paper for the generalization of conventional edge detectors to enable them to detect texture edges as well. Starting with feature calculation, he generated a set of feature images. Application of simple gradient operators on these feature images produces a corresponding set of texture gradient images. To get a final texturally enhanced image, the author proceeded with weighting these gradient images on a pixel basis. Use of multiple local texture properties is also demonstrated in [35] and [44].

7.3.2 SFT: Spatial Filtering Technique

The SFT is based on the application of spatial filters which are capable of amassing image energy onto areas of texture edges followed by intensity thresholding. In practice, no formal design procedure for this kind of filters has been suggested. Instead, one may try to generalize some optimal intensity edge detection filters to texture edge detection. In [30], **Shanmugan** *et al.* developed a filter for optimal edge detection in gray images. The procedure is later generalized to texture edge detection in 1-D by **Shanmugan** and **Townsend** [31]. They assumed that textures are specified by sinusoidal functions with different frequencies. This 1-D technique is extended to 2-D by **Park** *et al.* [32]. They found that the optimal filter can be well approximated by the derivative of a Gaussian function which they called the $\omega - Gaussian\ filter$.

7.3.3 TEA: Theoretical Estimation Approach

In principle, texture edge detection algorithms parallel to the SEA technique for uniform texture region extraction can be designed. This necessitates the modeling of both the region textures and the texture boundaries. For simplicity, let us focus our attention on 1-D data. Assume the 1-D image $\bar{y}$ of pixels $x_1, x_2, \ldots, x_N$ contains two textures t_1 and t_2. Let n_0 be the true boundary position between t_1 and t_2, and n be a candidate for the boundary. Represent all the

pixels to the left of position n by vector $\vec{y}_1$ and all the pixels to the right of n by $\vec{y}_2$. Further let the two conditional joint probability density functions (pdf) denoted by $p(\vec{y}_1|n)$ and $p(\vec{y}_2|n)$. Now, if we assume that pixels in t_1 and t_2 are independent, then the joint probability function of $\vec{y}$ can be written as

$$p(\vec{y}|n) = p(\vec{y}_1|n)p(\vec{y}_2|n)$$

or the likelihood function

$$L(n) = log[p(\vec{y}|n)] = \sum_{i=1}^{2} \log[p(\vec{y}_i|n)]$$

For the maximum likelihood estimate n^* of the texture boundary position, one seeks for the maximum of $L(n)$:

$$n^* = arg\Big[\max_{1<n<N} L(n)\Big]$$

with $L_{\max} = L(n^*)$.

A TEA algorithm for unsupervised texture edge detection is proposed by **Kashyap** and **Eom** [46]. They divided a 2-D image into horizontal (vertical) strips which are assumed to contain only vertical (horizontal) texture edges. By row (column) concatenation, pixels in each strip are represented by a 1-D vector $\vec{y}$. Then for each strip, the procedure outlined above is executed in order to detect the texture edge position. The joint pdf $p(\vec{y}_i|n)$, $i=1,2$ is derived from the assumed *long correlation model* for the image data $\vec{y}$. The final edge map is then obtained by combining results from all strips.

In addition to being expensive in computation, results from a TEA technique may be very noisy and the detected texture edges may contain many disconnected segments as seen in [46].

7.3.4 TBM: Texton-Based Method

As we pointed out in Chapter 1, the general mechanism of human visual perception of texture may operate in two distinct models: preattentive and attentive texture perception with the former being predominant in early vision. Preattentive texture perception and discrimination occurs almost instantaneously without element-by-element scrutinization. The

question is what exactly causes preattentive texture discrimination? Recent computational and psychological findings on human texture vision assert that preattentive texture discrimination is based on the differences in some conspicuous local features [36,47,48]. These conspicuous features are termed as *textons* by **Julesz** [48]. Precisely, Julesz defined the following as textons: nonoverlapped line segments, terminators of line segments, crossings of line segments and elongated blobs. In the past few years, numerous experiments have demonstrated that the differences in the density, orientation, size, or contrast of textons cause immediate perception of texture boundaries [49].

In realistic texton-based texture boundary detection, the central problem, however, is how can textons be computed from gray level images of natural scenes, and how exactly can texture boundaries be detected? The answers to these questions have been attempted by **Voorhees** and **Poggio** [50,51]. They presented a texture edge detection algorithm based on blob texton analysis. The input image is first convolved with the Laplacian of a Gaussian of standard deviation 1.5 pixels generating a map image of blobs. For each blob, some attributes such as contrast and orientation are computed. Then a non-parametric statistic is defined which predicts the presence or absence of a texture boundary between two regions. This statistic is based on the maximum difference between corresponding blob attribute histogram buckets in adjacent neighborhoods.

Another example of the TBM technique is given by **Rearick** [52]. The proposed algorithm is based on the calculation of the local changes in the densities of three local features or textons: end point, middle point (connecting point) and vertex (crossing point). The input image is first binarized and skeletonized in order to generate a syntactic map of texton types. Then, from this map, texton density gradients are computed and texture edges are located.

In conclusion, the TBM technique is at its infant stage and more efforts in this direction are required.

7.4 Conclusion

In this chapter, we attempted to review previous work on texture segmentation. Although review papers on gray level image segmentation are readily available [4], a review paper on texture segmentation techniques has not been seen in the open literature. Due to its inherent

difficulties, texture segmentation has been long recognized as a very knotty problem and yet it is an essential part of any low-level vision systems. Although texture segmentation draws increasing attention from researchers, current progress in this area is far from satisfactory as may be seen in this chapter.

In the coming two chapters, we shall present a number of novel algorithms for both uniform texture region extraction and texture edge detection.

7.5 References

[1] **B.J.Schachter** *et al.*, *Some Experiments in Image Segmentation by Clustering of Local Feature Values*, Pattern Recognition, vol.11, 1979, pp.19-28.

[2] **E.E.Triendl**, *Texturerkennung und Texturreproduktion*, Kybernetik, vol.13, 1973, pp.1-5.

[3] **K.I.Laws**, *Textured Image Segmentation*, PhD dissertation, University of Southern California, California, 1980.

[4] **R.M.Haralick** and **L.G.Shapiro**, *Image Segmentation Techniques*, Comput. Vision, Graphics and Image Proc., vol.29, pp.100-132, 1985.

[5] **W.K.Pratt**, *Digital Image Processing*, New York: Wiley Interscience, 1978.

[6] **R.Wang** *et al.*, *Texture Analysis Based on Local Standard Deviation of Intensity*, Proc. of the Inter. Conf. on Comput. Vision and Pattern Recognition, FL,USA, 22-26 June 1986, pp.482-488.

[7] **S.W.Zucker** *et al.*, *Picture Segmentation by Texture Discrimination*, IEEE Trans. Compt., vol.C-24, pp.1228-1233, 1975.

[8] **O.R.Mitchell** and **S.G.Charlton**, *Image Segmentation Using a Local Extrema Texture Measure*, Pattern Recognition, vol.10, No.3, pp.205-210, 1978.

[9] **R.L.Kashyap** *et al.*, *Stochastic Model Based Technique for Texture Segmentation*, Proc. of the Inter. Conf. on Pattern Recognition, Montreal, Que., Canada, 30 July - 2 Aug., 1984, pp.1104-7. pp.1202-1205.

[10] **H.Wechsler**, *Taxonomy and Segmentation of Textured Images*, Proc. of 5th Inter. Conf. on Pattern Recognition, Miami Beach, Florida, Dec.1-4, 1980, pp.535-539.

[11] **K.Deguchi** and **I.Morishita**, *Texture Characterization and Texture-Based Image Partitioning Using 2-D Linear Estimation Techniques*, IEEE Trans. Compt., vol.C-27, pp.739-745, 1978.

[12] **D.Barba** and **J.Ronsin**, *New Method in Texture Analysis in the Context of Image Segmentation*, Sig. Proc. II: Theories and Appl., Proc. of EUSIPCO'83, Erlanger, Germany, 12-16 Sept. 1983, pp.283-286.

[13] **S.Chatterjee** and **R.Chellappa**, *Texture Segmentation Using Gaussian Markov Random Field Models*, Topic Meeting on Machine Vision, Incline Village, NV, USA, 20-22 march 1985, pp.FB1/1-4.

[14] **C.H.Chen**, *On Statistical Image Segmentation Techniques*, Proc. of Inter. Conf. on Pattern Recognition and Image Processing, Aug.3-5 1981, pp.230-236.

[15] **R.W.Conners** *et al.*, *Segmentation of a High-Resolution Urban Scene Using Texture Operators*, Comput. Vision, Graphics and Image Proc., vol.25, pp.273-310, 1984.

[16] **H.Derin** and **W.S.Cole**, *Segmentation of Textured Images Using Gibbs Random Fields*, Comput. Vision, Graphics and Image Proc., vol.35, pp.72-98, 1986.

[17] **H.Derin** and **H.Elliott**, *Modeling nad Segmentation of Noisy and Textured Images Using Gibbs Random Fields*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-9, pp.39-55, 1987.

[18] **C.W.Therrien**, *An Estimation-Theoretic Approach to Terrain Image Segmentation*, Comput. Vision, Graphics and Image Proc., vol.22, pp.313-326, 1983.

[19] **F.S.Cohen** and **D.B.Cooper**, *Simple Parallel Hierarchical and Relaxation Algorithm for Segmenting Textured Images based on Noncausal Markovian Random Field Models*, Proc. of the Inter. Conf. on Pattern Recognition, Montreal, Que., Canada, 30 July - 2 Aug. 1984, pp.1104-1107.

[20] **P.C.Chen** and **T.Pavlidis**, *Segmentation by Texture Using a Co-occurrence Matrix and a Split-and-Merge Algorithm*, Comput. Graphics and Image Proc., vol.10, pp.172-182, 1979.

[21] **P.C.Chen** and **T.Pavlidis**, *Segmentation by Texture Using Correlation*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-5, pp.64-69, 1983.

[22] **M.Pietikainen** and **A.Rosenfeld**, *Image Segmentation by Texture Using Pyramid Node Linking*, IEEE Trans. on Syst. Man and Cybern., vol.SMC-11, pp.822-825, 1981.

[23] **B.P.Kjell** and **C.R.Dyer**, *Segmentation of Textured Images*, Proc. of the Inter. Conf. on Comput. Vision and Pattern Recognition, Miami Beach, Florida, June 22-26, 1986, pp.476-481.

[24] **H.M.Raafat** and **A.K.C.Wong**, *Texture-Based Image Segmentation*, *ibid*, pp.469-475.

[25] **A.G.Weber** and **A.A.Sawchuk**, *Segmentation of Textured Images*, Topic Meeting on Machine Vision, Incline Village, NV, USA, 20-22 March 1985, pp.FB1/1-4.

[26] **J.M.Cibulskis** and **C.R.Dyer**, *An Analysis of Node Linking in Overlapped Pyramids*, IEEE Trans. on Syst. Man and Cybern., vol.SMC-14, pp.424-436, 1984.

[27] **J.G.Daugman**, *Uncertainty Relations for Resolutions in Space, Spatial Frequency and Orientation Optimized by Two-Dimensional Visual Cortical Fields*, J.Opt.Soc.Am. A, vol.2, No.7, pp.1160-1169, 1985.

[28] **M.R.Turner**, *Texture Discrimination by Gabor Functions*, Biol. Cybern., vol.55, pp.71-82, 1986.

[29] **M.Clark** *et al.*, *Texture Segmentation Using a Class of Narrowband Filters*, Proc. of the Inter. Conf. on Acoustics, Speech and Signal Processing, Dallas, Texas, USA, 6-9 April 1987, pp.571-574.

[30] **K.S.Shanmugan** and *et al.*, *An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.PAMI-1, pp.37-49, 1979.

[31] **K.S.Shanmugan** and **J.K.Townsend**, *An Optimal Frequency Domain Textural Edge Detection Filter*, Topic Meeting on Machine Vision, Incline Village, NV, USA, 20-22 March 1984, pp.FB2/1-4.

[32] **J.Park** *et al.*, *Two-Dimensional Textural Edge Detection Using w-Gaussian Filter*, Appl. Opt., vol.25, pp.2357-2364, 1986.

[33] **D.Wermser** and **C.E.Liedtke**, *Texture Gradient: A New Tool for the Unsupervised Segmentation*, Proc. of the 3rd Scandinavian Conf. on Image Analysis, Copenhagen,

Denmark,July 12-14, 1983, pp.85-89.

[34] **S.Grinaker**, *Edge Based Segmentation and Texture Separation*, Proc. of the Inter. Conf. on Pattern Recognition, Miami Beach, Florida, USA, Dec.1-4 1980, pp.554-557.

[35] **W.B.Thompson**, *Texture Boundary Analysis*, IEEE Trans. on Comput., vol.C-26, pp.272-276, 1977.

[36] **B.Julesz** and **J.R.Bergen**, *Textons, the Fundamental Elements in Preattentive Vision and Perception of Textures*, Bell Sys. Tech. Journal, vol.62, pp.1619-1645, 1983.

[37] **X.Michel** *et al.*, *Unsupervised Segmentation of Texture Images*, Proc. of SPIE, vol.1001, 1988, pp.582-590.

[38] **R.E.Bellman**, *Dynamic Programming*, Oxford: University Press, 1957.

[39] **Z.Fan** and **F.S.Cohen**, *Textured Image Segmentation as a Multiple Hypothesis Test*, IEEE Trans. on Circuits and Systems, vol.35, 1988, pp.691-702.

[40] **H.Shen** and **A.K.C.Wong**, *Generalized Texture Representation and Metric*, Comput. Vision, Graphics and Image Proc., vol.23, 1983, pp.187-206.

[41] **H.M.Raafat** and **A.K.C.Wong**, *A Texture Information-Directed Region Growing Algorithm for Image Segmentation and Region Classification*, Comput. Vision, Graphics and Image Proc., vol.43, pp.1-21, 1988.

[42] **P.K.Bashir**, *A Filter Design Approach to Textured Image Segmentation*, Proc. of SPIE, vol.635, 1986, pp.352-356.

[43] **C.L.Richards, Jr**, *A New Image Segmentation and Texture Analysis Algorithm*, Proc. of SPIE, vol.697, 1986, pp.308-315.

[44] **E.E.Triendl** and **T.Henderson**, *A Model for Texture Edges*, Proc. of the Inter. Conf. on Pattern Recognition, Miami Beach, Florida, Dec. 1-4, 1980, pp.1100-1102.

[45] **T.R.Reed** and **H.Wechsler**, *Texture Segmentation and Organization Using The Wigner Distribution*, Proc. of EUSIPCO-88, Grenoble, France, Sept. 5-8, 1988, pp.263-266.

[46] **R.L.Kashyap** and **E.B.Eom**, *Texture Boundary Detection Based on the Long Correlation Model*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.11, 1989, pp.58-67.

[47] **J.J.Beck** *et al.*, *A Theory of Texture Segmentation*, in J.Beck et al. (editors): *Human and Machine Vision*, London: Academic Press, 1983.

[48] **B.Julesz**, *Texton Gradient: The Texton Theory Revisited*, Biol. Cybern., vol.54, 1986, pp.245-251.

[49] **B.Julesz**, *A Theory of Preattentive Texture Discrimination Based on First-Order Statistics of Textons*, Biol. Cybern. 41, 1981, pp.131-138.

[50] **H.Voorhees** and **T.Poggio**, *Detecting Textons and Texture Boundaries in Natural Images*, Proc. of 1st Inter. Conf. on Computer Vision, London, England, June 8-11, 1987, pp.250-258.

[51] **H.Voorhees** and **T.Poggio**, *Computing Texture Boundaries from Images*, Nature, vol.333, 1988, pp.364-367.

[52] **T.C.Rearick**, *A Texture Analysis Algorithm Inspired by A Theory of Preattentive Vision*, Proc. of the Inter. Conf. on Comput. Vision and Pattern Recognition, San Francisco, California, June 19-23, 1985, pp.312-317.

8. TEXTURE SEGMENTATION BASED ON LVM

8.1 Introduction

In Chapter 3, we described a general scheme for texture analysis called the *local vector mapping (LVM)* scheme. The scheme is subsequently incorporated into texture feature extraction and classification. Texture features defined therein are the first-order statistics (specifically the first four moments) of the transformed or LVMed images, and have proven to be powerful in discriminating textures which possess identical first-order statistics (as a result of histogram equalization) in Chapter 3. These properties of those texture features imply that

1. LVMed images of different textures usually have distinct first-order probability distribution functions (pdfs); and
2. LVM operations are capable of transforming higher-order statistic differences (e.g., second-order statistic differences) into lower-order statistic differences (e.g., first-order statistic differences).

With the above two implications in mind, we shall in this chapter investigate how the LVM scheme may be incorporated into texture segmentation. As mentioned in the preceding chapter, a texture segmentation problem can be solved either by *uniform texture region extraction* or by *texture edge detection*. Algorithms for both alternatives will be presented in this chapter. Specifically, two algorithms for uniform texture region extraction will be described in the next section, followed by the development of an algorithm for texture edge detection in Section 8.3. All the algorithms are evaluated under a variety of textures. The chapter is concluded in Section 8.4.

8.2 Uniform Texture Region Extraction Based on LVM

Uniform texture region extraction (UNITRE), also known as *region-based texture segmentation*, attempts to identify regions containing uniform textures. A basic requirement on UNITRE techniques is the connected interiors of the extracted regions. This requirement is, however, not satisfied by most of the existing UNITRE techniques, particularly those belonging to FSC, STC and SEA (See Chapter 7). The criterion governing the segmentation process in UNITRE is the homogeneity of region textures. In this section, we shall present two UNITRE algorithms capable of extracting connected regions. Their schematic diagrams will first be given

followed by the explanations of their principles step by step. Implementation results of the algorithms are summarized in the last part of the section.

8.2.1 The Basic LVM Algorithm

The first algorithm is the basic local vector mapping algorithm whose block diagram is illustrated in Fig.8.1. It is composed of local vector mapping, intensity thresholding, optimal low-pass filtering and gray intensity segmentation.

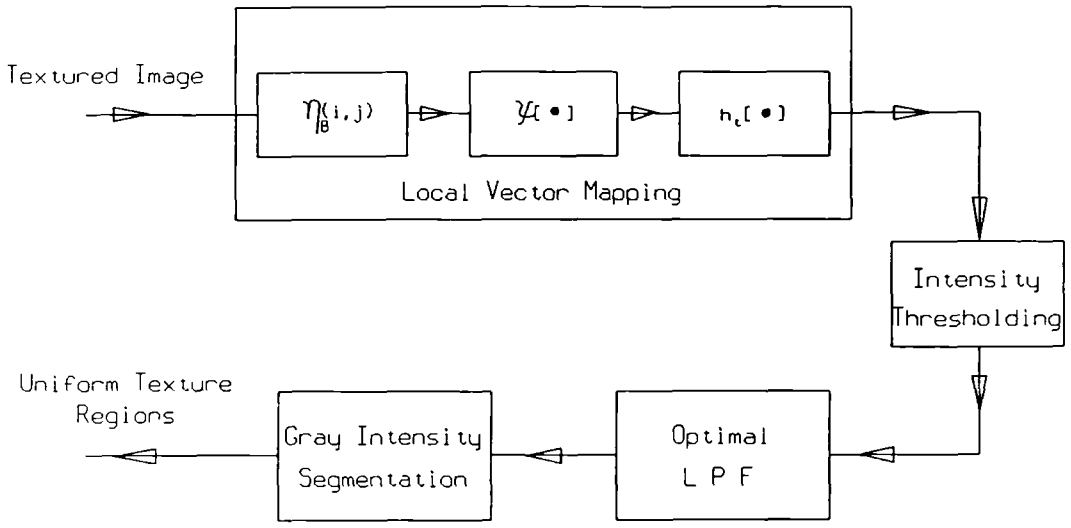


Figure 8.1. Block Diagram of the Basic LVM Algorithm for UNITRE.

A. Local Vector Mapping

This is the first and also the crucial step of the algorithm. The principles of LVM have been detailed in Chapter 3. A complete LVM, as shown in Fig.8.1, consists of the specification of an appropriate local neighborhood $\eta_B(i,j)$, the selection of a vector formation scheme $\Psi[\eta_B(i,j)]$, and the choice of an effective vector mapping operation $h_t[\vec{V}(i,j)]$. After LVM, the original image $f(i,j)$ is expected to be transformed into a texture feature image $g_t(i,j)$ in which differently textured regions exhibit distinct first-order pdfs.

B. Intensity Thresholding

The distinct first-order pdfs of differently textured regions in the LVMed image $g_t(i,j)$

imply that the density of pixels, whose intensities are above or below some threshold t, will be different for distinct textures as illustrated in Fig.8.2.

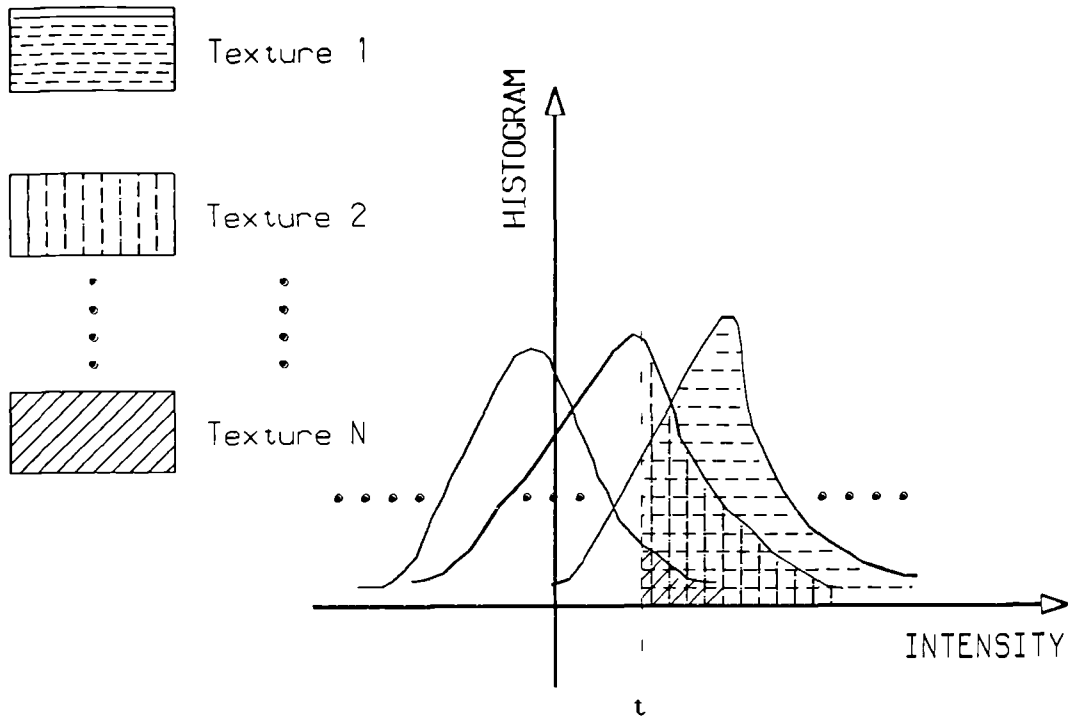


Figure 8.2. Illustration of Intensity Thresholding of LVMed Images.

This implication prompts the inclusion of an intensity thresholding step in the proposed algorithm shown in Fig.8.1. This step attempts to convert the "implicit" differences in the first-order statistics (not necessary the differences in average intensities) into the "explicit" differences in the density of white or black pixels. Regarding this step, some remarks should be made. The selection of a pertinent threshold is a non-trivial task and inappropriate thresholds may make distinct texture fields indiscriminable after intensity thresholding. Nevertheless, the distinct first-order pdfs of the differently textured regions in LVMed images ease the task of threshold selection to some extent. Finally, although the selection of a "good" threshold can be done usually by visual inspection, automatic selection of an optimal threshold is possible. For example, let $f(i,j)$ contain two distinct textures T_1 and T_2, further suppose that the LVMed image from T_1 is normally distributed with $N(m_1,\sigma_1)$, and the corresponding LVMed image from T_2 with $N(m_2,\sigma_2)$, then to minimize the classification error, an optimal threshold would be

chosen to be

$$t = \frac{m_1\sigma_2 + m_2\sigma_1}{\sigma_1 + \sigma_2} \tag{8.1}$$

When $\sigma_1 = \sigma_2$, (8.1) becomes

$$t = \frac{m_1 + m_2}{2} \tag{8.2}$$

where t is the threshold, m_1 and m_2 are the means and σ_1 and σ_2 the standard deviations. Equations (8.1) and (8.2) are schematically shown in Fig.8.3(a) and (b) respectively.

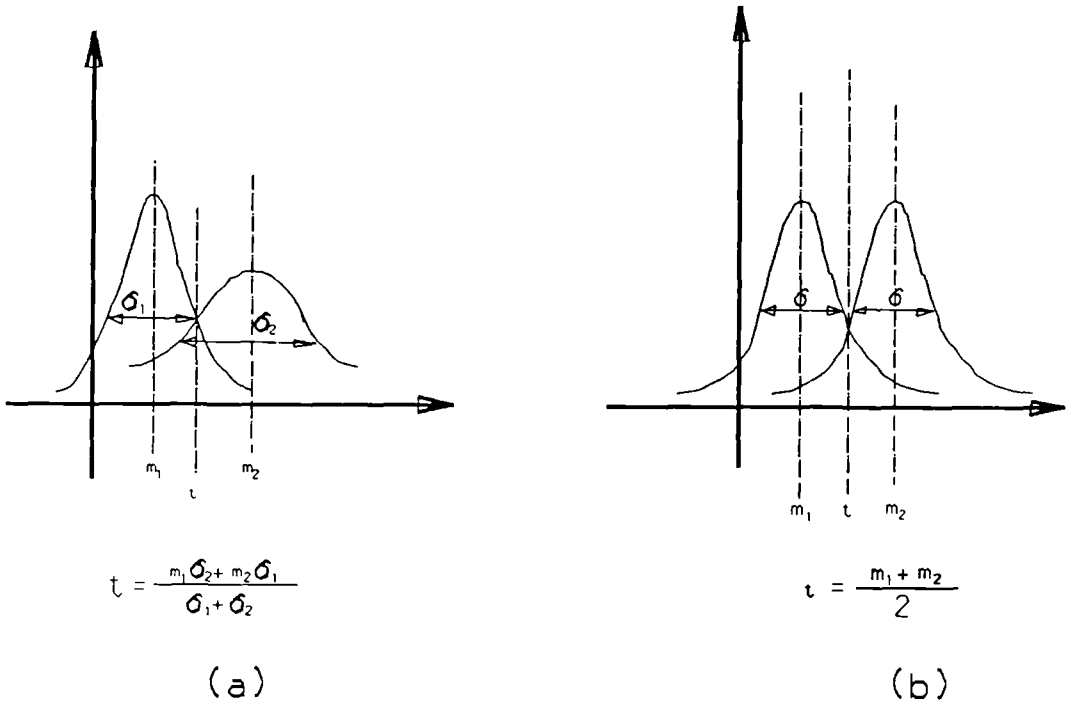


Figure 8.3. Selection of Optimal Intensity Thresholds.

It is also interesting to note that when the means are very close to each other, the effect of standard deviations on the selection of optimal thresholds is negligible as shown in the following. Let $m_2 = m_1 + \Delta m$ with $\Delta m \ll m_1, \sigma_1, \sigma_2$, then from (8.1), we have

$$t = \frac{m_1\sigma_2 + m_2\sigma_1}{\sigma_1+\sigma_2} = \frac{m_1\sigma_2 + m_1\sigma_1 + \Delta m\sigma_1}{\sigma_1+\sigma_2}$$
$$= m_1 + \Delta m\frac{\sigma_1}{\sigma_1+\sigma_2} \approx m_1 \tag{8.3}$$

At an extreme, when $m_1 = m_2$, t is then equal to m_1 (m_2) and is truly independent of the standard deviations. This is obviously consistent with our intuition. Automatic selection of thresholds for images containing more than two textures may also be possible. It should be pointed out that the Gaussian assumption on the first-order pdfs of LVMed images is reasonable for linear local vector mappings as discussed in Chapter 3 (See Fig.3.5 and Fig.3.6 in particular).

C. Optimal Low-Pass Filtering

Although the binary image resulted from intensity thresholding shows different densities of white or black pixels for differently textured regions, they are apparently not the final segmentation of the input image into uniform texture regions for

1. After segmentation, uniform texture regions should be distinctly labeled or represented by different gray intensities rather than different densities;
2. The extracted uniform texture regions should be connected;
3. The clear boundaries between differently textured regions should have been drawn in the final segmentation.

If the binary image can be converted into a gray image in which the intensity of a uniform texture region is proportional to the density of the white or black pixels in the region, then the converted image will show different average intensities from one texture region to another, and the application of some gray image segmentation techniques on this gray image may enable one to satisfy the above requirements (1),(2) and (3). Clearly, this image conversion process can be viewed as a filtering process and can be executed by means of low-pass filtering. To ease the task of the subsequent gray image segmentation and to extract correct texture regions, the underlying conversion filter should ideally satisfy the following:

1. *The filter should be such that uniform texture regions have intensities as smooth as possible, i.e., the filter should respond only to DC component or frequency components very close to*

DC component; and

2. *The filter should be such that different texture regions have different intensities and the transition regions from one texture type to another in the filtered image should be kept as small as possible.*

The first requirement is concerned with the resolution in the spectral domain and is related to the frequency response of the filter. It says that the filter should be a low pass filter with a very narrow bandwidth. On the other hand, the second requirement is concerned with the resolution in the spatial domain and is related to the impulse response of the filter. It requires that the filter should have a very narrow impulse response. From the theory of linear system, however, one knows that these two requirements are conflicting and can not be satisfied simultaneously. For 2-D linear filters, they are related by the following so-called **Uncertainty Principle** [2]:

$$\begin{cases} (\Delta x)(\Delta u) \geq \dfrac{1}{4\pi} \\ (\Delta y)(\Delta v) \geq \dfrac{1}{4\pi} \\ (\Delta x)(\Delta u)(\Delta y)(\Delta v) \geq \dfrac{1}{16\pi^2} \end{cases} \tag{8.4}$$

where

- x,y: time coordinates in the spatial domain;
- u,v: frequency coordinates in the spectral domain;
- $\Delta x, \Delta y$: effective widths in the spatial domain;
- $\Delta u, \Delta v$: effective widths in the spectral domain;

For a proof of (8.4), we refer to **Papoulis** [1]. For a compromise, one should choose filters which achieve equalities in (8.4). a 2D Gaussian low-pass filter of the form

$$f(x,y) = Ce^{-(\frac{x^2}{a^2}+\frac{y^2}{b^2})}$$

is shown to be a member of these *optimal* filters [1,2], where C is some constant. For simplicity, we have adopted a symmetrical 2-D Gaussian low-pass filter of the form

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-(\frac{x^2+y^2}{2\sigma^2})} \tag{8.5a}$$

or equivalently,

$$g(r) = \frac{1}{2\pi\sigma^2} e^{-(\frac{r^2}{2\sigma^2})} \tag{8.5b}$$

where σ is the standard deviation and $r = (x^2+y^2)^{1/2}$ is the polar distance from the origin of the filter to the point (x,y). In practice, recalling the exponential characteristic of Gaussian functions and for computational simplicity, the optimal low-pass filtering in Fig.8.1 has been confined to a window of size WxW, where W is usually specified by the user.

D. Gray Level Segmentation

After the first three steps in the proposed UNITRE algorithm, namely local vector mapping, intensity thresholding and optimal Gaussian low-pass filtering, the input textured image is step by step transformed into a non-textured or gray image. Further more, regions of identical texture will have similar intensities whereas regions of distinct texture will exhibit different intensity values in such a gray image. Therefore, segmentation of this gray image by means of some gray image segmentation methods may give one a partition of the input image into uniform texture regions. Nevertheless, because of the conflicting nature of requirements (1) and (2) given in the preceding paragraph, the method to be adopted for gray image segmentation should possess the following:

1. It is robust at the presence of noise or local intensity variations;
2. It is capable of extracting connected regions; and
3. It can locate accurate boundaries.

The RSST (Recursive Shortest Spanning Tree) segmentation algorithm recently developed by **Morris** *et al.* [3] seems to be a good candidate, and is employed in this thesis.

This completes the description of the principles of the basic LVM algorithm.

Before concluding the discussions on the basic LVM algorithm, the following is worth mentioning. A problem with the RSST technique is concerned with the determination of the

number of regions into which the input image is to be partitioned. The RSST can not automatically detect the number of major uniform regions appearing in the input image, and as a result, the number of regions has to be specified by the user. In our case, this problem may be overcome by analyzing the histogram of the image from optimal Gaussian low-pass filtering. Since after optimal low-pass filtering, pixels in a uniform texture region have similar intensities, each texture pattern may give a peak in the histogram of the smoothed image. So, the number of peaks in the histogram may correspond to the number of regions the RSST should work with. Note, this correspondence may not be true if some disjoint regions contain identical textures.

8.2.2 The Successive LVM Algorithm

In the preceding sub-section, we have discussed step by step the principles of the basic LVM algorithm shown in Fig.8.1. The algorithm, in its present form, is prone to some problems which are mainly due to the inclusion of an intensity thresholding step. It is realized that the selection of a good threshold is usually not easy and may be done only by a trial-and-error method. Meanwhile, the number of distinct textures which can appear in the input image may be necessarily limited to a small number for the successful application of the algorithm. This is because when some of the regions in a LVMed image have average intensities far away from the selected threshold, the resultant binary image after thresholding may have very similar densities of white (or black) pixels in these regions, and as a consequence these regions may have very similar intensity values after optimal Gaussian low-pass filtering, thus causing false segmentation. In addition, the simple intensity thresholding operation is always sensitive to noise, and the basic algorithm may completely fail in noisy environments. These problems may be overcome if

1. The intensity thresholding step is precluded from the algorithm; and
2. Only those LVMs are used which produce LVMed images exhibiting very different means for distinct textures.

Note, Condition (1) may be satisfied as a consequence of (2). Instead of searching for LVMs which are capable of mapping textural differences into large mean value differences on their own, we attempt to realize Condition (2) by means of repeated applications of some LVMs.

Since a single LVM operation may transform high-order statistic differences into lower-order differences, repeated applications of some LVMs may ultimately transform higher-order statistic differences (say second-order statistics differences) into very large mean value (the lowest-order statistic) differences. In other words, to obtain LVMed images with very different means for distinct texture regions, one first transforms an original image into a LVMed image using a LVM, then this LVMed image is taken as the input image to another LVM producing a LVM-LVMed image. This chain of LVMs is continued until the differently textured regions in the original image exhibit very different mean values in the LVM-...-LVMed image. Then optimal low-pass filtering on this successively LVMed image will generate a non textured image in which distinct texture regions have sufficiently different intensity values, and at the same time in which there is little intensity variation within uniform texture regions. Finally, the segmentation of the filtered image on the basis of intensity will give one a segmentation of the input image based on texture. The procedure just outlined will be termed as the *Successive LVM algorithm* or simply the SLVM algorithm. Fig.8.4 shows its block diagram.

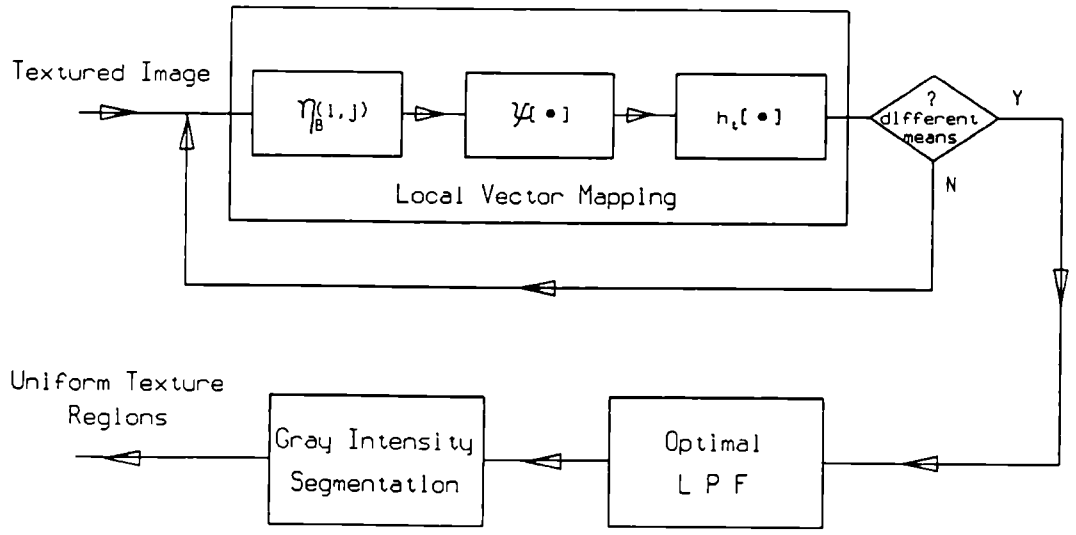


Figure 8.4. Block Diagram of the Successive LVM Algorithm for UNITRE.

The above description of the SLVM algorithm has been primarily at the abstract level. The remaining question, as far as practical implementation is concerned, is how such successive LVMs can be designed. To answer the question and for simplicity, let us focus our attention on LVM-LVM cases. It is shown in Chapter 3 that the variance features from LLVMed (linear

local vector mapped) images contain sufficient textural information for texture discrimination. Thus, if the intensity of a pixel in a LVM-LVMed image is related to the variance characteristic of a LLVMed image, then such a LVM-LVMed image is expected to possess very different means in distinct texture regions, and can be directly taken as the input to the optimal low-pass filter as shown in Fig.8.5.

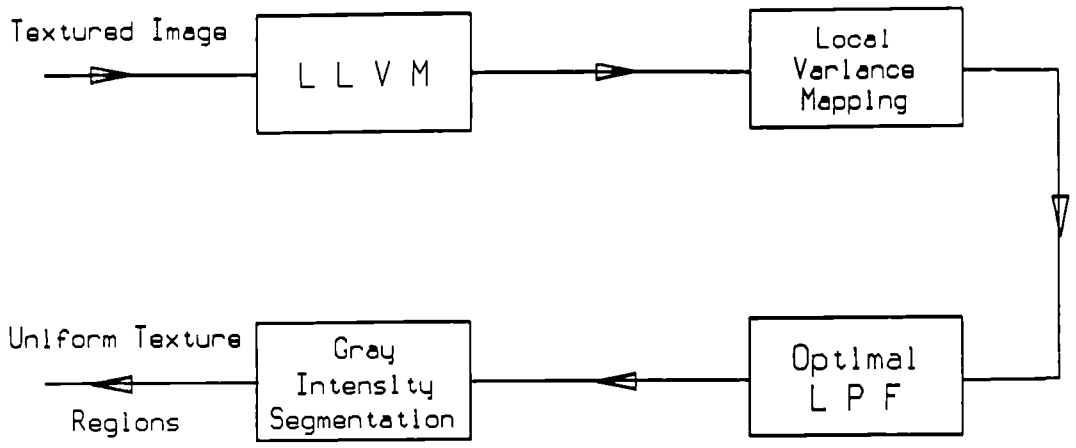


Figure 8.5. An Example of the Successive LVM Algorithm for UNITRE.

Clearly, Fig.8.5 is a special case of the general SLVM algorithm shown in Fig.8.4. It is also the specific SLVM algorithm to be implemented in the next sub-section. Note, according to our definitions in Chapter 3, local variance mapping is a NLVM, i.e., a non-linear local vector mapping (See Section 3.3.2). Thus what Fig.8.5 presents is an application of the combination of a LLVM and a NLVM.

8.2.3 Experimental Results

For the demonstration of the effectiveness of the proposed UNITRE algorithms, they are implemented under a wide variety of textures ranging from artificial regular textures, noisy textures to natural textures. Meanwhile, to show the accuracy of uniform texture region extraction of the algorithms, both artificial and natural texture boundaries are included. The following lists the relevant information required for the implementation:

- *Image Dimension:* 128x128;

- *Image Pre-Processing:* histogram equalization;
- *Local Neighborhood* $\eta_B(i,j)$: a 3x3 square window with central pixel (i,j);
- *Vector Formation Scheme* Ψ: row concatenation;
- *LLVM Used:* local discrete cosine mapping/transform defined in Section 3.3.1;
- *NLVM Used:* local variance mapping defined Section 3.3.2;
- *Optimal Low-Pass Filter:* Gaussian low-pass filter specified in (8.5) with filter parameter σ;
- *Filtering Window:* a WxW square region;
- *Gray Intensity Segmentation:* The RSST segmentation technique developed in [3];
- *Successive LVM:* local discrete cosine mapping + local variance mapping;

Note, the operation of histogram equalization has been taken to compensate for the lighting and other image acquisition distortions. It also makes all texture samples have identical first-order statistics.

8.2.3.1 Implementation of the Basic LVM Algorithm

In this sub-section, we shall present the implementation results of the basic LVM algorithm. The results are grouped according to the types of textures used.

A. Segmentation of Artificial Textures

Fig.8.6 to Fig.8.8 are some examples of segmentation of artificial textures. Fig.8.6 shows the first example. The original image (Fig.8.6(a)) is composed of two uniform texture regions. The background texture region consists of regularly arranged U elements while the other (the central region) contains $-90^0-rotated$ Us. The two regions have the same first order statistics but different second-order statistics because of rotation. Fig.8.6(b) is the LVMed or texture feature image $g_2(i,j)$ (For the definition of $g_2(i,j)$, see (3.7) or (3.10)). It is clear that the second-order or higher-order difference in Fig.8.6(a) has been to some extent transformed into the first-order difference (mean value difference) in Fig.8.6(b). Fig.8.6(c) is the result of thresholding $g_2(i,j)$ at threshold t=210, while Fig.8.6(d)-(f) show the Gaussian low-pass filtering results of Fig.8.6(c) under different filter parameters and window sizes respectively.

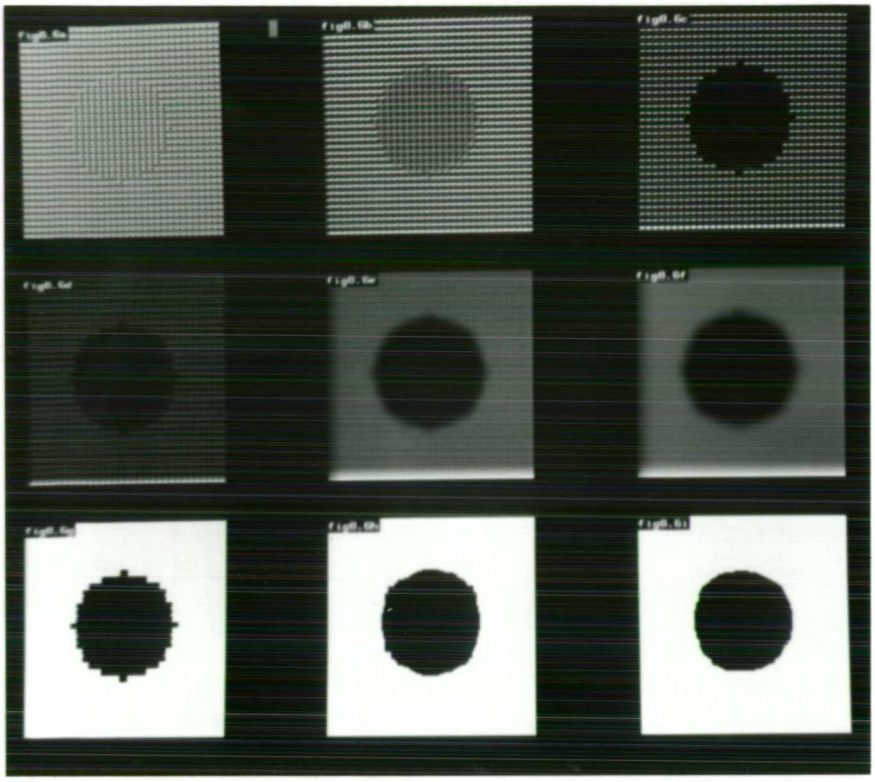


Figure 8.6. Segmentation of Artificial Textures by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_2(i,j)$; (c) Thresholding of (b) at t=210; (d) GLPF of (c) with W=5 and σ=6.0; (e) GLPF of (c) with W=13 and σ=4.0; (f) GLPF of (c) with W=19 and σ=4.0; (g) Intensity segmentation of (d); (h) Intensity segmentation of (e) and (i) Intensity segmentation of (f).

From Fig.8.6(d)-(f), one can see that with the increase of window size, the intensities in uniform texture regions are getting smoother whereas the "transition regions" between distinct texture regions are widening. Fig.8.6(g)-(i) present the RSST segmentations of Fig.8.6(d)-(f) respectively. Fig.8.6(g) is the best result in this case since the texture boundary detected matches exactly the actual one in Fig.8.6(a). Fig.8.7 gives the second example of artificial texture segmentation. The original image Fig.8.7(a) is composed of four uniform texture

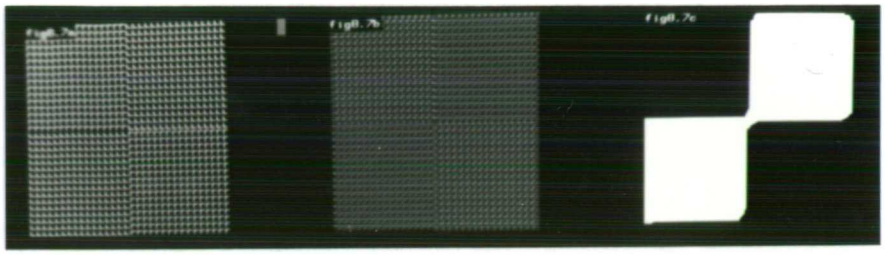


Figure 8.7. Segmentation of Artificial Textures by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_9(i,j)$ and (c) Final segmentation obtained with t=180, W=19 and $\sigma=4.0$.

regions containing either regular Ls or $-90^0-rotated$ Ls. Similar operations to those in Fig.8.6 have been performed but only the LVMed image $g_9(i,j)$ (Fig.8.7(b)) and the final segmentation (Fig.8.7(c) are given because of space limitation. Note, segmentation errors only occur near texture boundaries. This is reasonable as texture is a regional property and is undefinable at a single boundary pixel.

So far, texture boundaries have been artificial. They are either circle-shaped or plus-shaped. To test the general applicability of the algorithm, we give a more complicated example in Fig.8.8. Fig.8.8(a) shows an object-background type binary image. The object with dark intensity is a space shuttle. The background is filled with regular Us and the object area with $180^0-rotated$ Us generating a textured image with space-shuttle-like texture boundary in Fig.8.8(b). According to Julesz's 4-disc texture generation procedure [10], one knows that the two textures in Fig.8.8(b) have identical both first-order and second-order statistics but distinct third-order statistics. Therefore, methods based on first-order or second-order statistics such as [11-14] may not be capable of segmenting Fig.8.8(b) into uniform texture regions. Without pixel-by-pixel scrutinization, it may take seconds for one to outline the shape of the space shuttle in the textured image Fig.8(b). However, this task is much easier in the LVMed image $g_6(i,j)$ as illustrated in Fig.8.8(c). Additional operations of intensity thresholding (Fig.8.8(d)), filtering (Fig.8.8(e)) and RSST segmentation achieve the final segmentation of Fig.8.8(b) into

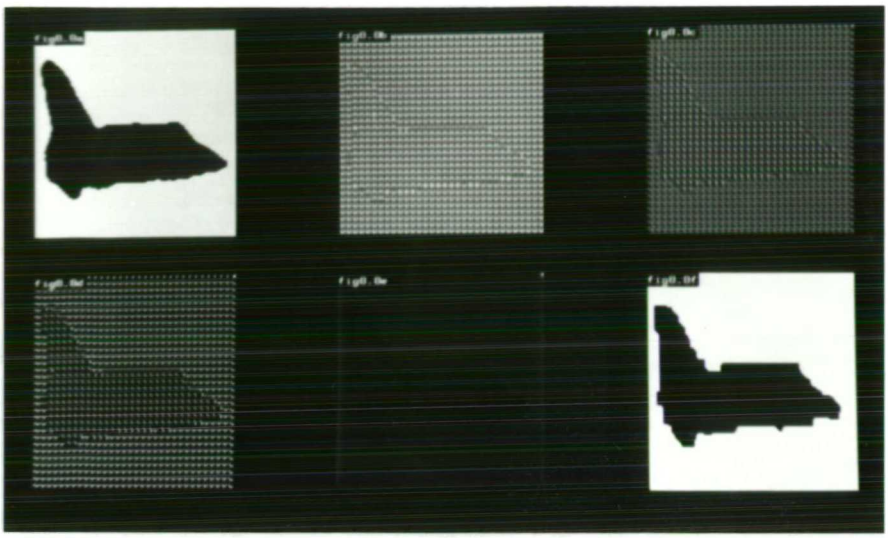


Figure 8.8. Segmentation of Artificial Textures by the Basic LVM Algorithm. (a) A space shuttle binary image; (b) A textured version of (a); (c) The LVMed image $g_6(i,j)$; (d) Thresholding of (c) at t=170; (e) GLPF of (d) with W=13 and $\sigma=2.0$ and (f) Intensity segmentation of (e).

two regions of object and background respectively (Fig.8.8(f)).

The results presented above demonstrated that

1. The basic LVM algorithm is workable with regular artificial textures under both artificial and natural texture boundaries;
2. LVM is capable of transforming second-order statistic differences into first-order differences;
3. LVM is capable of transforming third-order statistic differences into first-order differences; and
4. The algorithm is able to segment images which may be difficult or impossible for visual preattentive segmentation.

B. Segmentation of Noisy Textures

In part A, all textured images used contain regularly arranged primitives such as Us and Ls. In practice, however, images are irregular either because they come from irregular sources or because the regularity of the original images has been distorted by noise. In this part, we will investigate the ability of the algorithm to texturally segment noisy, irregular textured images. To do so, we have added different amounts of noise to the regular image of Fig.8.6(a). The resultant noisy images are shown respectively in Fig.8.9(a)-(c) at signal-to-noise ratio (SNR) 8db, 6db and 4db.

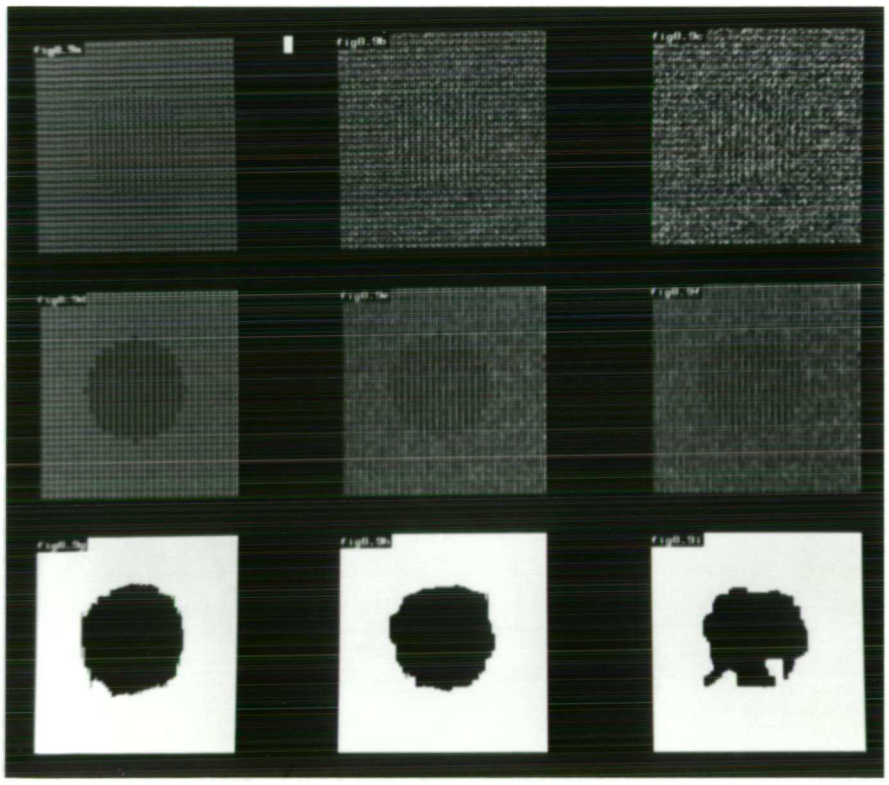


Figure 8.9. Segmentation of Noisy Textures by the Basic LVM Algorithm. (a)-(c) Noisy versions of Fig.8.6(a) at SNR=8db, 6db and 4db respectively; (d)-(f) The LVMed images $g_7(i,j)$ of (a)-(c) respectively and (g)-(i) Final segmentations of (a)-(c) respectively, obtained with t=160, W=13 and $\sigma=4.0$.

Note, with the increase of noise, the two texture regions become visually less discriminable. Fig.8.9(d)-(f) display the corresponding LVMed images $g_7(i,j)$s of Fig.8.9(a)-(c). Fig.8.9(g)-(i) are the final segmentations. As one may expect, the segmentation results are getting poorer with the increasing noise level.

To end this part, we want to point out that $g_7(i,j)$ has been applied in partitioning the three textured images Fig.8.9(a)-(c), while $g_2(i,j)$ has been used to segment their noise-free version of Fig.8.6(a). This is not surprising since the addition of noise certainly changes the statistics or local features. The results in this part showed that:

1. Different LVMs are sensitive to different local features or textons; and
2. The algorithm is workable with irregular, noisy images.

C. Segmentation of Brodatz (Natural) Textures

Having established that the algorithm is applicable in artificial textures, the question of whether it is also workable under natural textures remains to be answered. In this part, we shall test the algorithm with images composed of natural textures taken from **Brodatz** [4]. Fig.8.10(a) displays a composite image of Cotton Canvas and Wood Grain with the texture boundary being two diagonal lines. Because of histogram equalization, the two textures are not discriminable solely based on the first-order statistics as can be seen in Fig.8.10(a). The LVMed image $g_3(i,j)$, however, shows different first-order statistics in the two texture regions as given in Fig.8.10(b). Thresholding of Fig.8.10(b) at t=160 gives the binary density image Fig.8.10(c). Fig.8.10(d) is the filtered version of Fig.8.10(c) at window size W=13 and filter parameter $\sigma=4.0$. Finally, Fig.8.10(e) presents the final segmentation, which is quite satisfactory. One more example of natural texture segmentation by the basic algorithm is illustrated in Fig.8.11(a)-(e). These figures are analogous to those in Fig.8.10 but with different parameters as given in the caption. The original textured image in this case contains two visually similar textures namely Hand-made Paper and Calf.

A conclusion which can be drawn upon the results given in this part is

1. The basic LVM algorithm is capable of segmenting images containing natural textures.

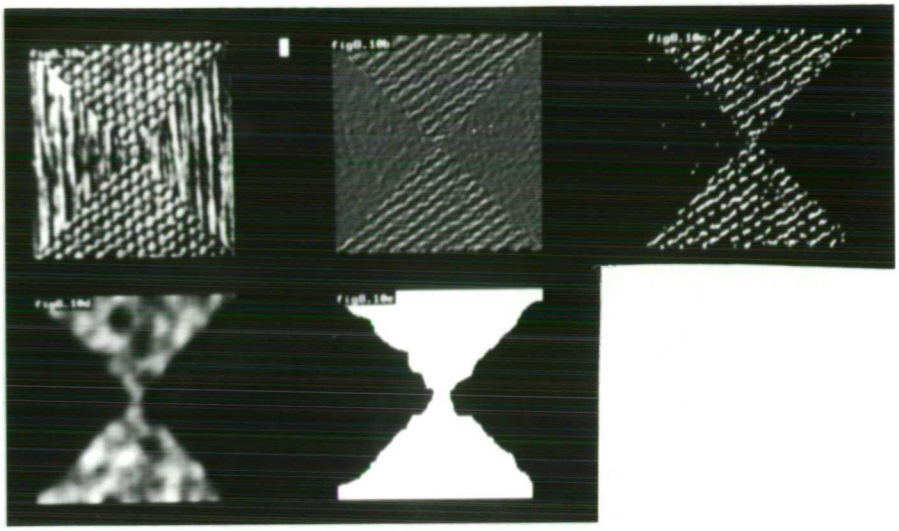


Figure 8.10. Segmentation of Natural Textures (Cotton Canvas and Wood Grain) by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_3(i,j)$; (c) Thresholding of (b) at t=160; (d) GLPF of (c) with W=13 and σ = 4.0 and (e) Intensity segmentation of (d).

8.2.3.2 Implementation of the Successive LVM Algorithm

Now, let us turn our attention to the implementation of the SLVM algorithm. Results are also arranged according to the type of textures involved.

A. Segmentation of Artificial Textures

Fig.8.12 is the first example. Fig.8.12(a) is the original textured image also used in Fig.8.6. Fig.8.12(b) is the LVMed image $g_2(i,j)$. Instead of taking intensity thresholding on $g_2(i,j)$, we now perform a LVM operation on $g_2(i,j)$. This second-stage LVM, as we discussed, is a NLVM, i.e., the local variance mapping defined in Section 3.3.2. The LLVM-NLVMed image is given in Fig.8.12(c). It is clear the second-order or higher-order statistic differences in Fig.8.12(a), which may have caused the visual texture segmentation of Fig.8.12(a), have been mapped into very large mean value differences in Fig.8.12(c) after two stage LVM operations. Fig.8.12(d)-(i) are then analogous to Fig.8.6(d)-(i) except that the input image to the Gaussian

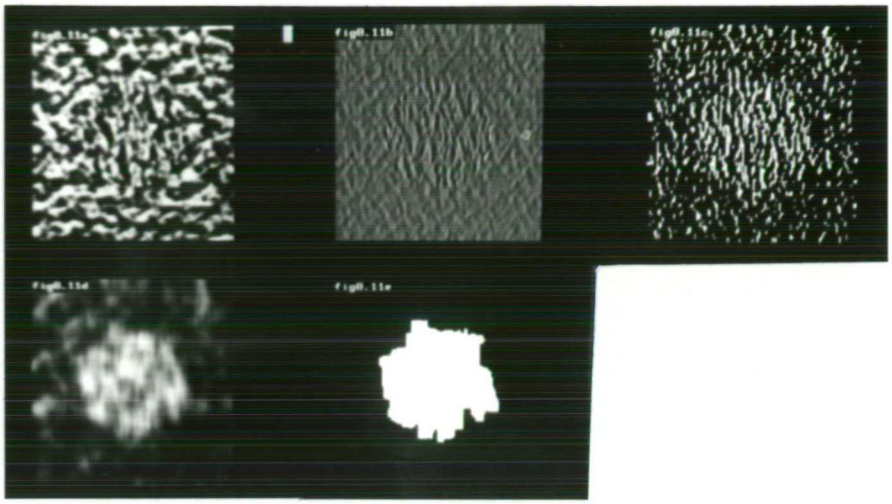


Figure 8.11. Segmentation of Natural Textures (Hand-made Paper and Calf) by the Basic LVM Algorithm. (a) Original image; (b) The LVMed image $g_7(i,j)$; (c) Thresholding of (b) at t=145; (d) GLPF of (c) with W=11 and σ = 4.0 and (e) Intensity segmentation of (d).

filter is the LLVM-NLVMed image Fig.8.12(c). Note, the segmentation results shown in Fig.8.12(g)-(i) are quite comparable to those of Fig.8.6(g)-(i).

B. Segmentation of Natural Textures

Fig.8.13 shows the results of successive LVM texture segmentation of Fig.8.11(a) which is re-displayed in Fig.8.13(a). After LLVM $g_7(i,j)$ and local variance mapping, the original image has been mapped into an image in which the two texture regions have very different mean values. Then, the same set of operations is performed on this LLVM-NLVMed image as on the binary image in Fig.8.11(c) giving the final texture segmentation result in Fig.8.13(e). The comparison between Fig.8.11(e) and Fig.8.13(e) clearly indicates that the successive LVM texture segmentation algorithm performs much better than the basic algorithm in this case. Fig.8.14 is an example of successive LVM segmentation of images with more than two texture patterns. The original composite in Fig.8.14(a) contains four uniform texture regions with four

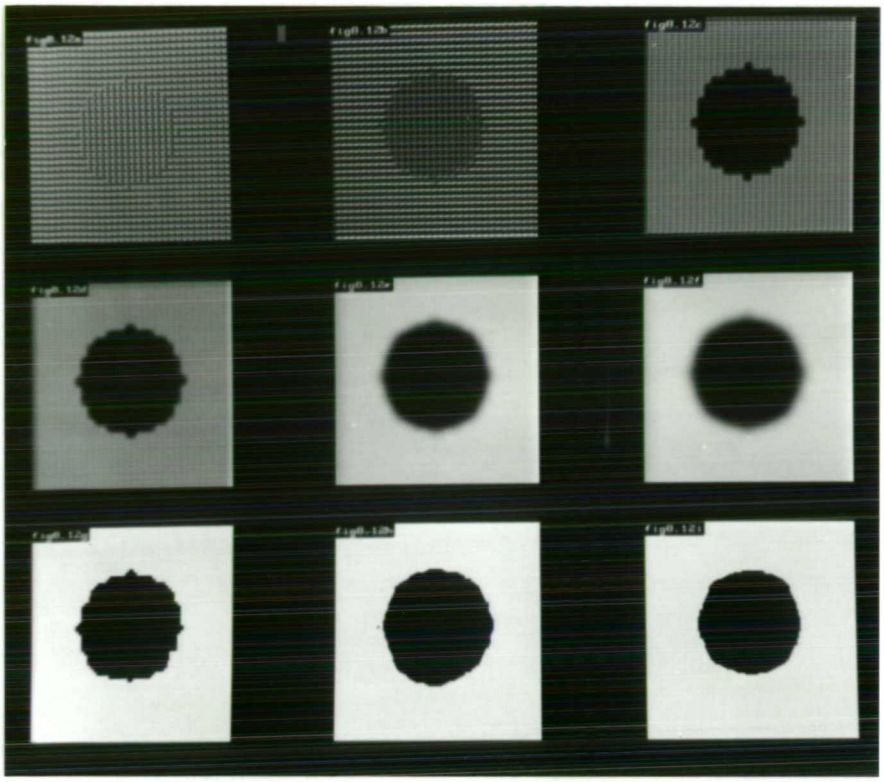


Figure 8.12. Segmentation of Artificial Textures by the SLVM Algorithm. (a) Original image; (b) The LVMed image $g_2(i,j)$; (c) The SLVMed image; (d) GLPF of (c) with W=13 and σ = 2.0; (e) GLPF of (c) with W=13 and σ = 4.0; (f) GLPF of (c) with W=19 and σ = 4.0 and (g)-(i) Intensity segmentations of (d)-(f) respectively.

natural textures taken from [37] namely Wood Grain, Calf, Hand-made Paper and Pressed Cork. Note, the last three texture samples are visually very similar. Fig.8.14(b) shows the LLVMed texture feature image $g_3(i,j)$ while Fig.8.14(c) is its NLVMed version by local variance mapping. Fig.8.14(d) and (e) are the smoothed images of Fig.8.14(c) at different filter parameters or window sizes. Fig.8.14(f) and (g) then present their respective segmentation by the RSST technique. The four segmented regions (top-left, top-right, bottom-left and bottom-right) in Fig.8.14(f) are labeled with intensity 88, 16, 13 and 150 respectively, while the four

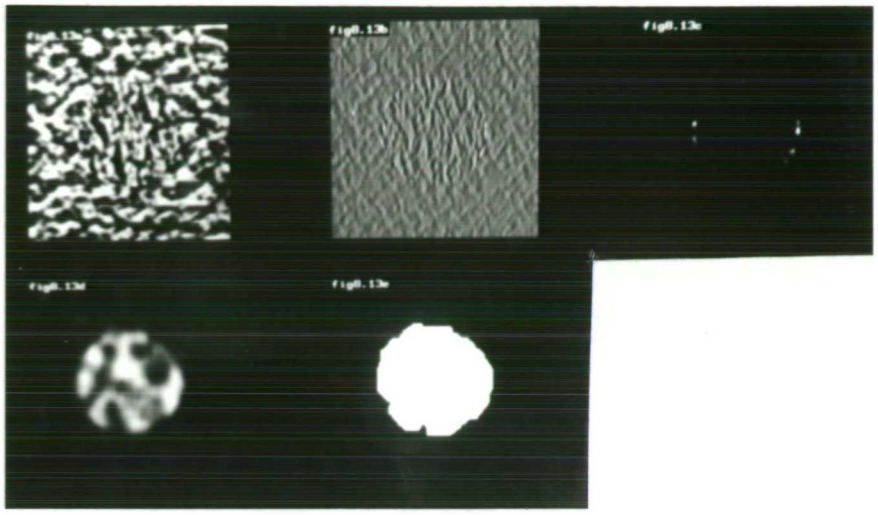


Figure 8.13. Segmentation of Natural Textures (Hand-made Paper and Calf) by the SLVM Algorithm. (a) Original image; (b) The LVMed image $g_7(i,j)$; (c) The SLVMed image; (d) GLPF of (c) with W=11 and σ = 4.0 and (e) Intensity segmentation of (d).

extracted regions in Fig.8.14(g) have intensities 95, 16, 3 and 154 respectively. Because of display limitations, the intensity differences between the top-right and bottom-left regions of Fig.8.14(f) and (g) are not easily visible in the taken pictures. Considering the similarity between the component textures, these segmentation results are quite acceptable.

The final example of successive LVM texture segmentation is given in Fig.8.15. Fig.8.15(a) is the original non-textured image which contains a white space shuttle on a dark background. This binary image is then converted into a textured image in Fig.8.15(b) by filling the object with Cotton Canvas texture and the background with Wood Grain texture. LLVM $g_3(i,j)$ followed by local variance mapping of this textured image gives subsequently the images in Fig.8.15(c) and (d). Optimal low-pass filtering of (d) produces the non-textured image Fig.8.15(e). The RSST segmentation of (e) enables one to identify the space shuttle under the textural environment in Fig.8.15(b). The final result shown in Fig.8.15(f) is very impressive.

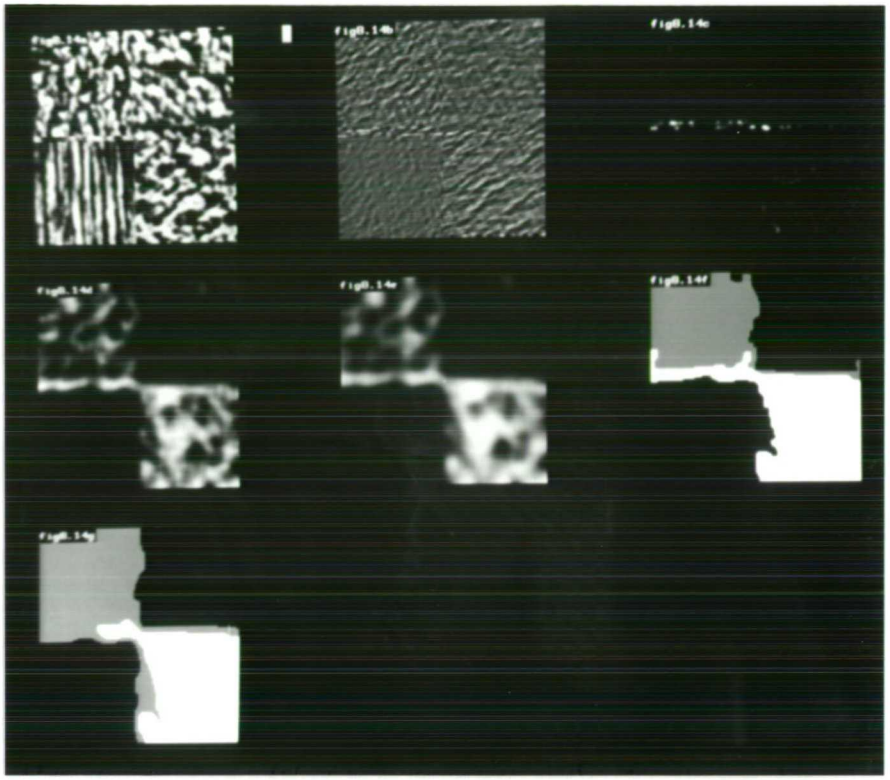


Figure 8.14. Segmentation of Natural Textures (Wood Grain, Calf, Hand-made Paper and Pressed Cork) by the SLVM Algorithm. (a) Original image; (b) The LVMed image $g_3(i,j)$; (c) The SLVMed image; (d) GLPF of (c) with W=9 and σ = 4.0; (e) GLPF of (c) with W=13 and σ = 4.0 and (f)-(g) Intensity segmentations of (d) and (e) respectively.

The preceding discussions demonstrated that

1. The SLVM algorithm performs generally better than the basic LVM algorithm; and
2. The SLVM algorithm is workable with natural textures and natural texture boundaries.

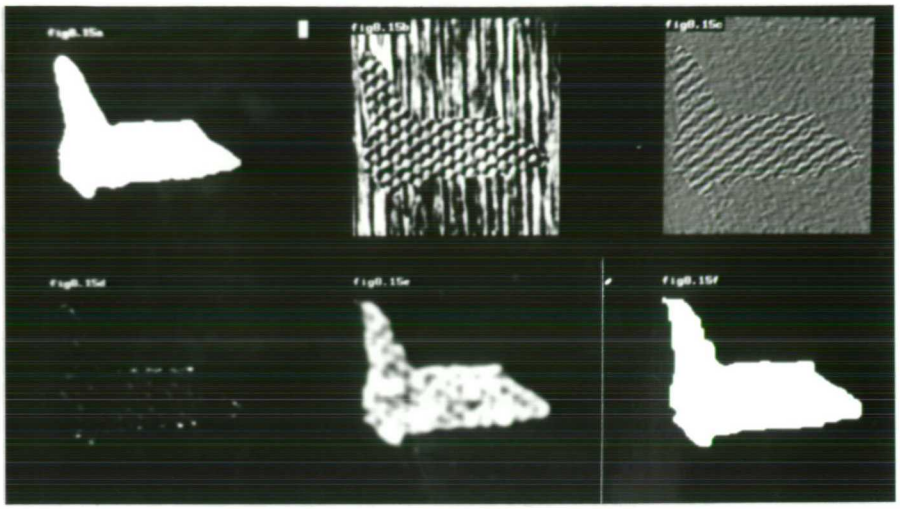


Figure 8.15. Segmentation of Natural Textures (Cotton Canvas and Wood Grain) by the SLVM Algorithm. (a) A binary space shuttle image; (b) A textured version of (a); (c) The LVMed image $g_3(i,j)$; (d) The SLVMed image; (e) GLPF of (d) with W=13 and σ = 2.0 and (f) Final segmentation.

8.3 Texture Edge Detection Based on LVM

Texture edge detection (TEXTED), also referred to as *edge-based texture segmentation*, aims to establish the boundaries separating uniform texture regions. A fundamental requirement on TEXTED techniques is the connectedness of the detected texture edges. This requirement is, however, not complied with by most of the existing TEXTED techniques, especially those belonging to TEA and TBM (See Chapter 7). The criterion directing the segmentation process in TEXTED is the discontinuity of textural properties from region to region. In this section, we shall describe a multi-channel-like TEXTED algorithm capable of detecting connected texture edges of a single pixel width.

8.3.1 The Algorithm

One of the models for texture perception in the Human Visual System (HVS) is to hypothesize that the processing of afferent pictorial information involves a set of parallel spatial

filtering operations (See Chapter 6). Each of these spatial filters can be regarded as being tuned to a specific texture attribute. In reality, there are few textures which differ from one another in only one particular texture attribute. In [5] and also the last section, it is demonstrated that a LVM or a combination of some LVMs is only particularly sensitive to a specific texture attribute. Thus the simultaneous application of a set of LVM operations is usually requisite in more realistic situations. These considerations motivate us to propose a LVM-based multi-channel algorithm for texture edge detection. Each channel employs a distinct LVM operation. The algorithm will be hereafter called the *multi-channel LVM texture edge detection* algorithm or the *MUCTED* algorithm for brevity. Fig.8.16 shows the schematic diagram of the algorithm which can be envisaged as being composed of four major operations namely textural transform, textural gradient calculation, channel grouping, and intensity thresholding and post-processing. The principles of the algorithm are explained in the following paragraphs.

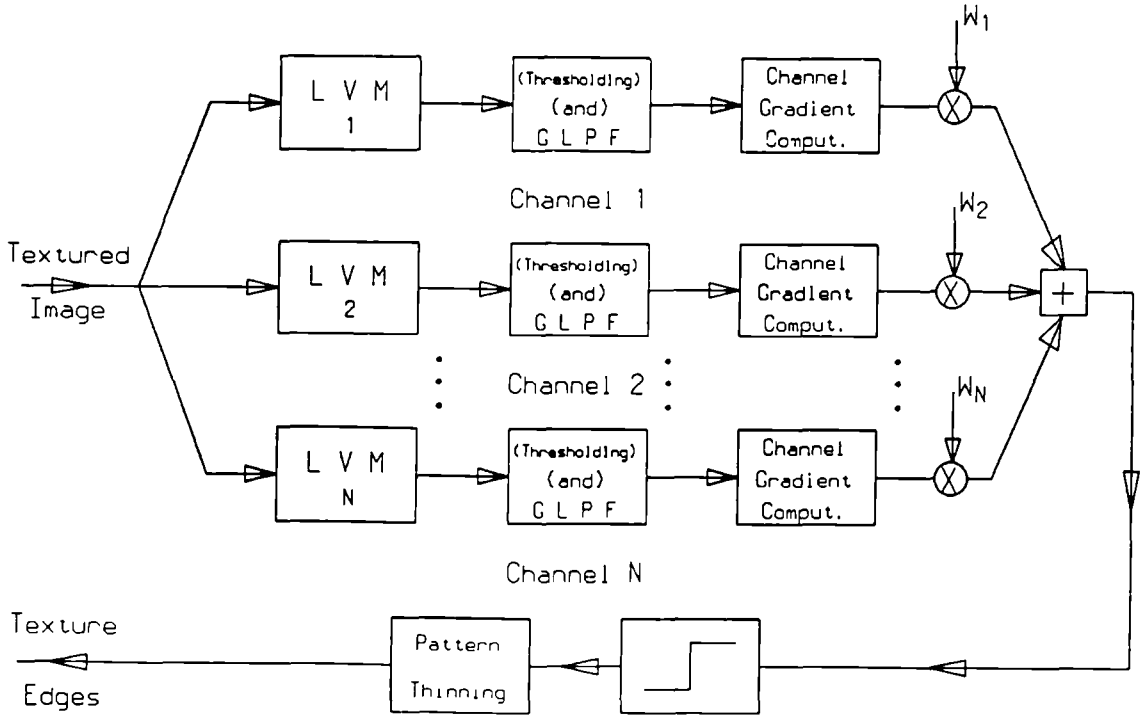


Figure 8.16. Block Diagram of the Proposed MUCTED Algorithm.

8.3.1.1 Textural Transform

The operation of textural transform attempts to effectively map textural differences into gray intensity differences. Textural differences are the differences in some texture attributes such as coarseness, directionality and regularity. As shown in Fig.8.16, textural transform in the proposed algorithm is achieved in each channel via a LVM operation followed by intensity smoothing and/or low-pass filtering. Each channel is assumed to be tuned to some specific texture attributes by applying a specific LVM operation.

For the first step in textural transform, only those LVM operations are chosen which respond to distinct textures very differently. The ultimate goal of performing LVM operations is to transform the differences in higher-order statistics (specifically the second-order statistics) into the differences in first-order statistics (specifically the average intensity). It is shown in [5] that some LVM operations such as those defined in Chapter 3 are capable of performing this kind of statistical complexity reduction operation. It is also argued in [6] that in order to achieve maximum amount of difference in mean values, successive applications of LVMs are necessary forming the so-called *successive local vector mapping (SLVM)* operations. If a single LVM operation is used in each channel, then an intensity thresholding operation on the LVMed image may be required prior to low-pass filtering as explained in the preceding section and [6].

After LVM or SLVM operations, the resultant LVMed or SLVMed images, i.e., the texture feature images, exhibit different average intensities in distinct texture regions. Nevertheless, because of local structural variations in texture, these texture feature images are necessarily noisy and intensity smoothing or low-pass filtering is essential in order to extract only the mean values. For the same considerations as stated in Section 8.2.1, Gaussian filters of (8.5) are employed for intensity smoothing.

8.3.1.2 Textural Gradient Calculation

The operation of textural transform transforms the original textured image into an intensity image in which uniform texture regions have smooth intensities whereas regions with distinct textures have different intensities. This implies that the texture edges in the original image can now be observed as intensity edges in the transformed image, and that the intensity gradients in the transformed image measure the textural gradients in the original image. Thus to compute the textural gradients of the original input image, one only needs to calculate the

intensity gradients in the transformed intensity image. In principle, any gradient operators useful for enhancing intensity edges are applicable here. Examples are Roberts, Sobel and Laplacian operators [7]. The Sobel operator has been chosen here. In other words, for the calculation of textural gradients, one computes:

$$\begin{aligned} G_n(i,j) &= \left[G_{nx}^2(i,j)+G_{ny}^2(i,j)\right]^{\frac{1}{2}} \\ G_{nx}(i,j) &= I_n(i+1,j-1)+2I_n(i+1,j)+I_n(i+1,j+1) \\ &\quad -I_n(i-1,j-1)-2I_n(i-1,j)-I_n(i-1,j+1) \\ G_{ny}(i,j) &= I_n(i-1,i+1)+2I_n(i,j+1)+I_n(i+1,j+1) \\ &\quad -I_n(i-1,j-1)-2I_n(i,j-1)-I_n(i+1,j-1) \end{aligned} \tag{8.6}$$

where $I_n(i,j)$ is the transformed intensity image from the n^{th} channel and $G_n(i,j)$ the resultant gradient magnitude image from the channel.

8.3.1.3 Channel Grouping

As each channel is supposed to be tuned to some specific texture attributes, a gradient image $G_n(i,j)$ obtained from a single channel is unlikely to represent the total amount of textural gradients in the original image. Channel outputs should be integrated in some way to produce a gradient image $G(i,j)$ measuring the total amount of textural gradients in the original image. Since different texture attributes may contribute differently to visual texture discrimination, channel outputs should be appropriately weighted before integration. For simplicity, the channel grouping operation outlined above is defined by the following linear model:

$$G(i,j) = \sum_{n=1}^{N} W_n G_n(i,j) \tag{8.7}$$

where N is the total number of channels and W_n, $n=1,2,\ldots,N$, are the weighting coefficients.

8.3.1.4 Intensity Thresholding and Post-Processing

Image $G(i,j)$ specified in (8.7) can be considered as the texturally enhanced version of the original image. Image energy in $G(i,j)$ is concentrated in areas of textural discontinuity. Therefore, simple intensity thresholding on $G(i,j)$ may be able to mark texture edges. Nevertheless, texture edges thus obtained are found very thick. So, an additional edge/line or pattern thinning operation (post-processing) is necessitated to get texture edges of single pixel width. This is achieved by the SPTA thinning algorithm developed in [8]. Since the SPTA

assumes that the input pattern is a smoothed pattern in the sense that all pattern irregularities and all salt-and-pepper noise (small holes) have been removed, majority filtering (median filtering) in 3x3 windows has been performed on the thresholded image (binary image) prior to the application of the SPTA algorithm. Further more, in order to reduce the total number of passes in the SPTA algorithm (thus reduce the computational time), and to eliminate isolated small edges, another operation has been undertaken between majority filtering and the SPTA. The operation works as follows. If all pixels in a sliding window of size MxM are edge points, then only the central pixel is retained to be an edge point. Otherwise if the window contains one or more non-edge pixels, then no edge point is defined in the window.

In the last few paragraphs, the principles of the MUCTED algorithm have been detailed. To summarize, let us denote the original input image by $T(i,j)$, the LVMed or SLVMed image (texture feature image) in the n^{th} channel by $F_n(i,j)$, the Gaussian low-pass filtered image (texturally transformed intensity image) by $I_n(i,j)$, the channel gradient image by $G_n(i,j)$, the total gradient image by $G(i,j)$, the thresholded binary image by $B(i,j)$, and the thinned image (texture edge map) by $E(i,j)$, then Fig.8.17 illustrates the major operations involved in the MUCTED algorithm.

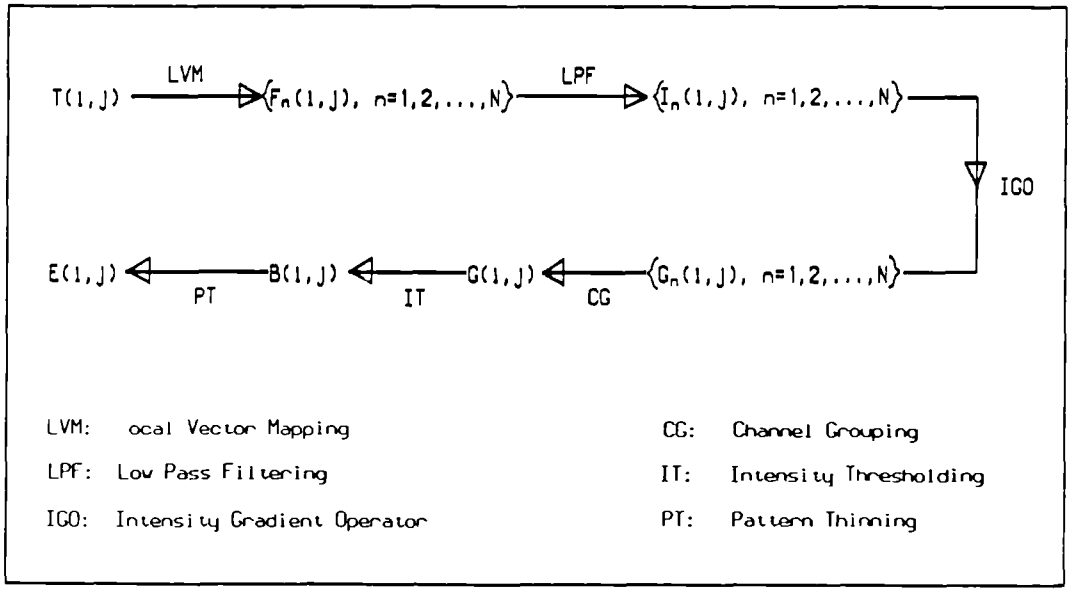


Figure 8.17. Major Operations Involved in the Proposed MUCTED Algorithm.

8.3.2 Experimental Results

To test the effectiveness of the proposed TEXTED algorithm, numerous experiments have been conducted. In the following, experimental results are presented and discussed. Before the presentation of the results, the relevant information for the implementation is listed as follows:

- *LVM Operations:* a LLVM defined in (3.7) + a NLVM (local variance mapping) defined in Section 3.3.2 with local neighborhood $\eta_B(i,j)$ = a 3x3 square window and vector formation scheme $\Psi[\bullet]$ = row concatenation;
- *Intensity Smoothing:* a symmetrical Gaussian low-pass filter defined in (8.5);
- *Gradient Operator:* a Sobel operator specified in (8.6);
- *Channel Grouping:* a linear grouping model defined in (8.7);
- *Post-Processing:* majority filtering in 3x3 sliding windows + small isolated edges elimination in *MxM* sliding windows + the SPTA described in [8];
- *Test Images:* natural textures and synthetic textures.

Fig.8.18 shows the first example of texture edge detection by the MUCTED. The original textured image Fig.8.18(a) is a composite of two natural textures Wood Grain and Cotton Canvas taken from **Brodatz** [4]. The textural boundary is the contour of a space shuttle. The total number of channels implemented is four. Fig.8.18(b)-(e) are the textural gradient images from Channels 1, 2, 3 and 4 respectively. Channel 1 consists of the LLVM $g_2(i,j)$ defined in (3.7), the local variance mapping $h_{var}[\vec{V}(i,j)]$, a Gaussian low-pass filter with parameter σ = 4.0, and a Sobel Operator. Channels 2, 3 and 4 are composed of similar operations except that the LLVMs used are $g_3(i,j)$, $g_5(i,j)$ and $g_9(i,j)$ defined in (3.7) respectively. Fig.8.18(f) is the image of total textural gradients, i.e., the textural gradient image corresponding to the original image. The weighting coefficients for Channels 1, 2, 3 and 4 are 1.0, 1.0, 1.0 and 0.5 respectively. Fig.8.18(g) shows the binary image obtained by thresholding the textural gradient image Fig.8.18(f) at threshold=90. Majority filtering and small isolated edges elimination operations give an image similar to Fig.8.18(g). Fig.8.18(h) is the final result of edges detected after applying the SPTA thinning algorithm on Fig.8.18(g). Fig.8.18(h) clearly shows that the edges detected are connected and of single pixel width. To show the positional accuracy of the

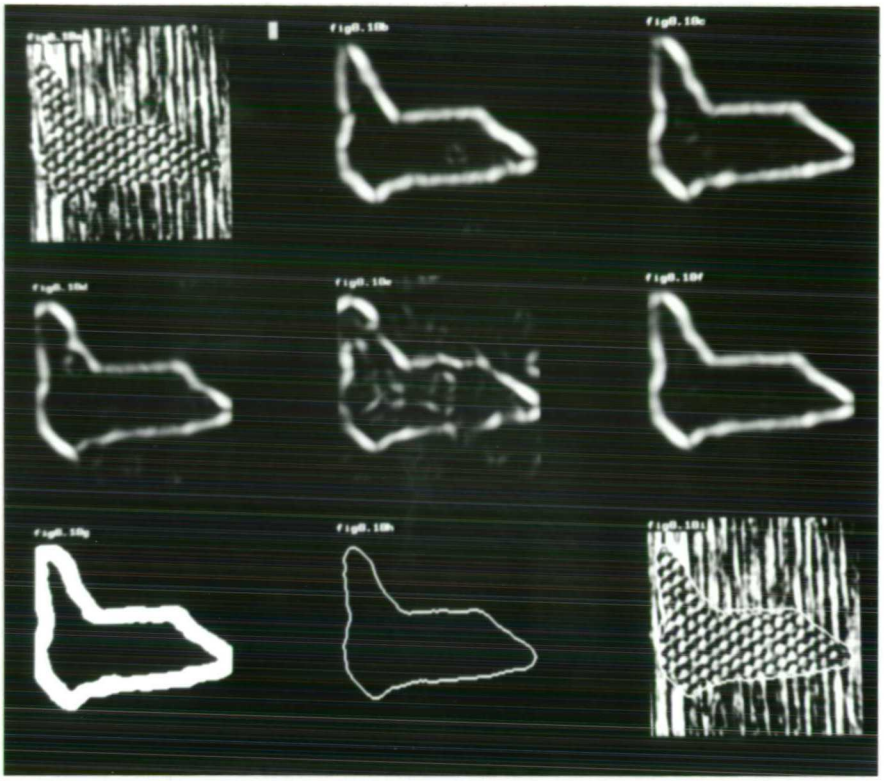


Figure 8.18. Texture Edge Detection in Natural Textures (Cotton Canvas and Wood Grain) by the MUCTED Algorithm. (a) Original image; (b)-(e) The four channel textural gradient images; (f) The total textural gradient image; (g) After thresholding (f) at t=90; (h) After thinning (g) and (i) The detected texture edges superimposed on (a).

boundary recovered, the detected edges are overlaid onto the original image as illustrated in Fig.8.18(i). It is evident that the edges detected are very accurate. A similar example to Fig.8.18 is given in Fig.8.19. The original image consists of a background Wood Grain texture and a circular central region of Hand-made Paper also taken from [4]. Then, similar operations to those in Fig.8.18 are performed except that the four channel LLVMs are now $g_2(i,j)$, $g_3(i,j)$, $g_5(i,j)$ and $g_7(i,j)$ respectively, and equal weights have been assigned to all

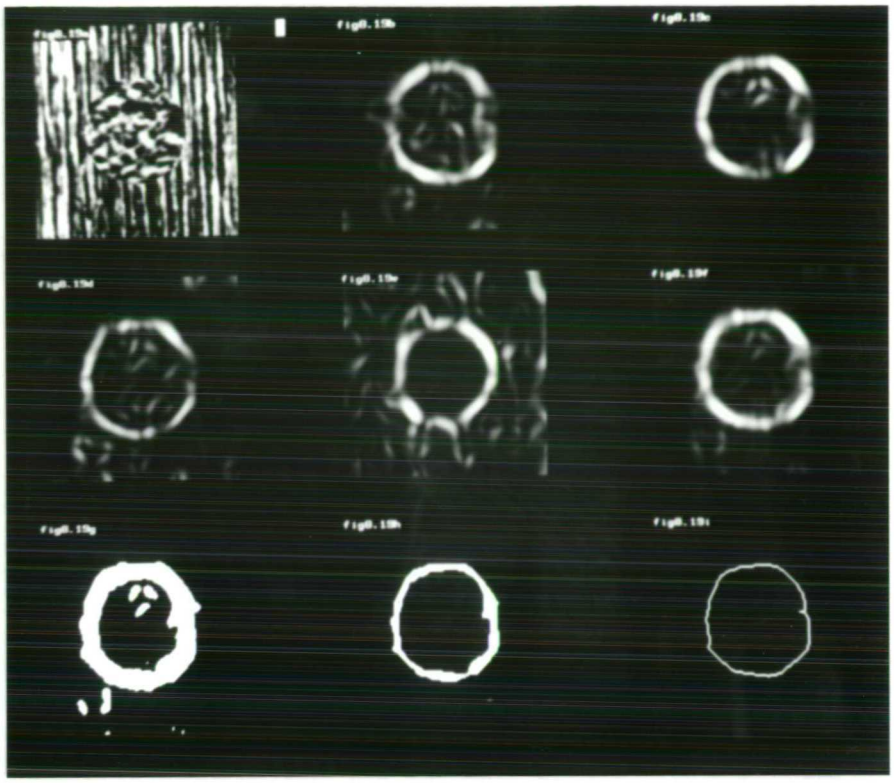


Figure 8.19. Texture Edge Detection in Natural Textures (Wood Grain and Hand-made Paper) by the MUCTED Algorithm. (a)-g) are analogous to Fig.8.18(a)-(g); (h) After majority filtering and small-isolated-edges elimination operations on (g) and (i) Texture edges detected.

channels in channel grouping. Fig.8.19(g) shows the binary image obtained by thresholding the textural gradient image Fig.8.19(f) at threshold=90. It is clear that this binary image contains some undesirable isolated edges and also some "holes". Thus, majority filtering and small-isolated-edges elimination operations are necessary. Fig.8.19(h) shows the result of these operations, where the small-edge-elimination operation is taken in a 7x7 sliding window. Fig.8.19(i) is the texture edge map produced by the algorithm.

In addition to natural textures, the MUCTED algorithm has also been tested on synthetic textures. Fig.8.20 shows 8 textured images containing artificial textures. The underlying component textures in each textured image are all visually quite similar, and in some cases, preattentive segmentation may be impossible.

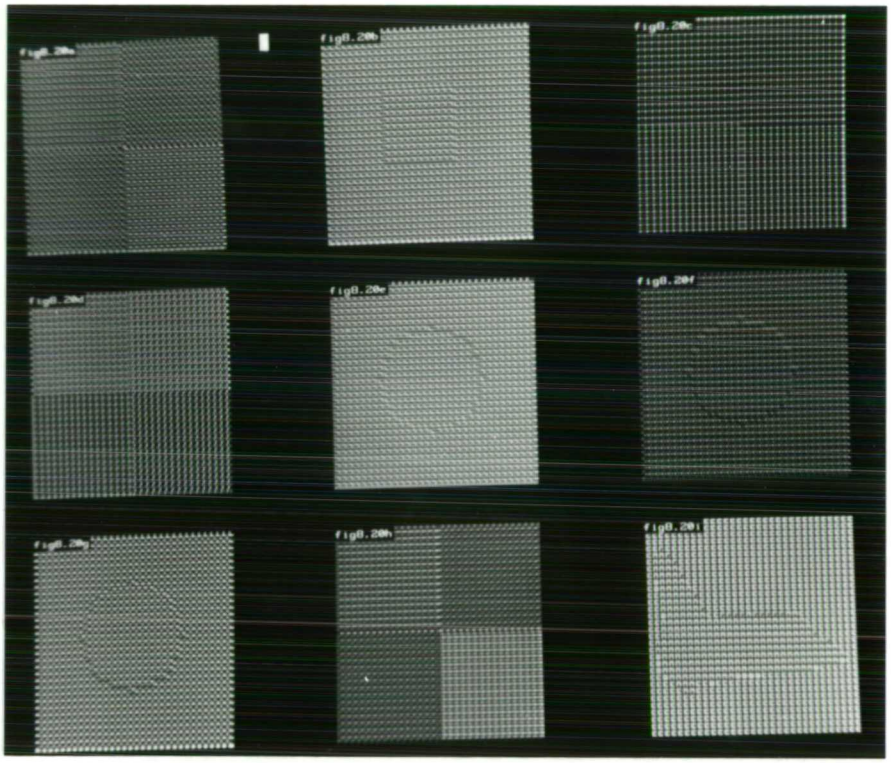


Figure 8.20. 8 Synthetic Textured Images Shown in Fig.8.20(a)-(i) Respectively.

In fact, if one glances at Fig.8.20(c)n without pixel-by-pixel scrutinization, one may only be able to identify a T-shaped texture edge separating the top half from the bottom half and the left half from the right half in the bottom half. However, a pixel-by-pixel scrutinization process will reveal the top half also contains two distinct texture patterns, which are 180^o-rotated to each other. Fig.8.21 illustrates the textural gradient images of the respective images in Fig.8.20. To obtain these gradient images, we have applied the same set of operations for each textured

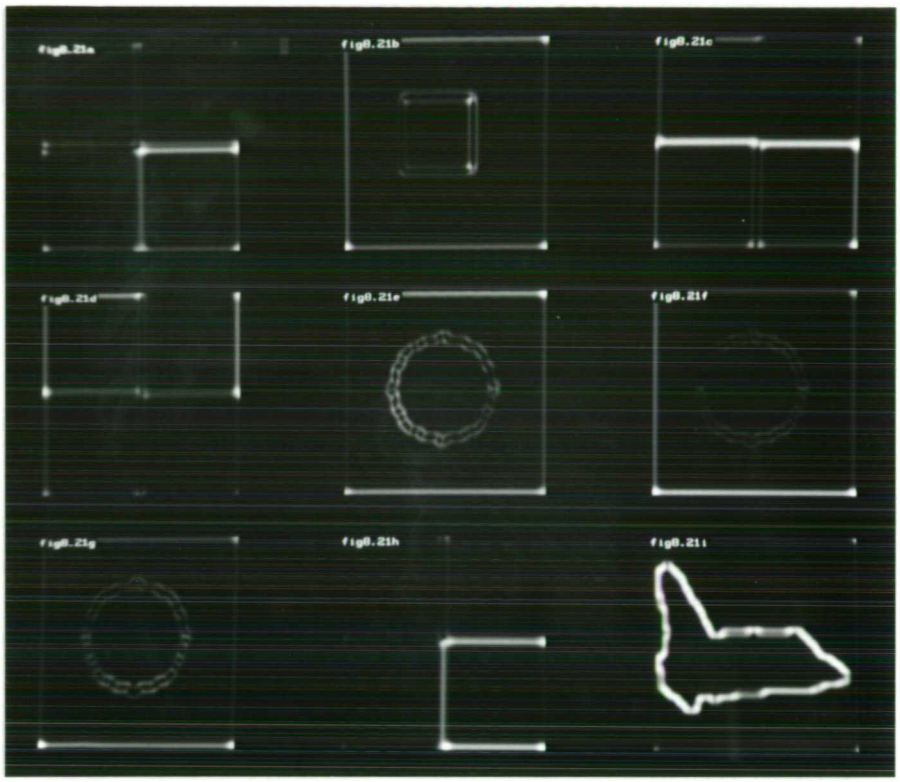


Figure 8.21. The 8 Total Textural Gradient Images Computed from Fig.8.20(a)-(i) Respectively.

image. The channel LLVMs used are $g_1(i,j)$ to $g_9(i,j)$ defined in (3.7) making the total number of channels be 9, and equal channel weights have been used. It is very interesting to note the intensity changes in these gradient images. In Fig.8.21(a), relatively high intensities occur around the boundary separating the bottom right quadrant from the other three quadrants. Visually, the texture in this quadrant can be more easily discriminated from the textures in other quadrants. The presence of the white blobs (distinct textons) in this quadrant is presumably the cause for its easier discrimination. In Fig.8.21(c), higher intensities are present in areas of a T-shaped boundary, i.e., a horizontal line between the top half and the bottom half and a vertical line dividing the left half and the right half in the bottom half. As already mentioned, these are

the edges one may preattentively be able to identify in the image. Similar phenomena can be observed in Fig.8.21(d) and (h). In Fig.8.21(d), the textures in the top half can be easily discriminated from the textures in the bottom half because of their slightly different average intensities. Similarly, the bottom right quadrant in Fig.8.21(h) can be easily separated from the other quadrants because of its slightly higher average intensity. A conclusion which can be drawn from these observations is that *the more the texture edges are visually identifiable, the more strongly the MUCTED algorithm responds to them*. This is obviously a desirable feature for a computational TEXTED algorithm. After intensity thresholding and post-processing being performed on these gradient images, Fig.8.22 shows the detected texture edges superimposed on the original images.

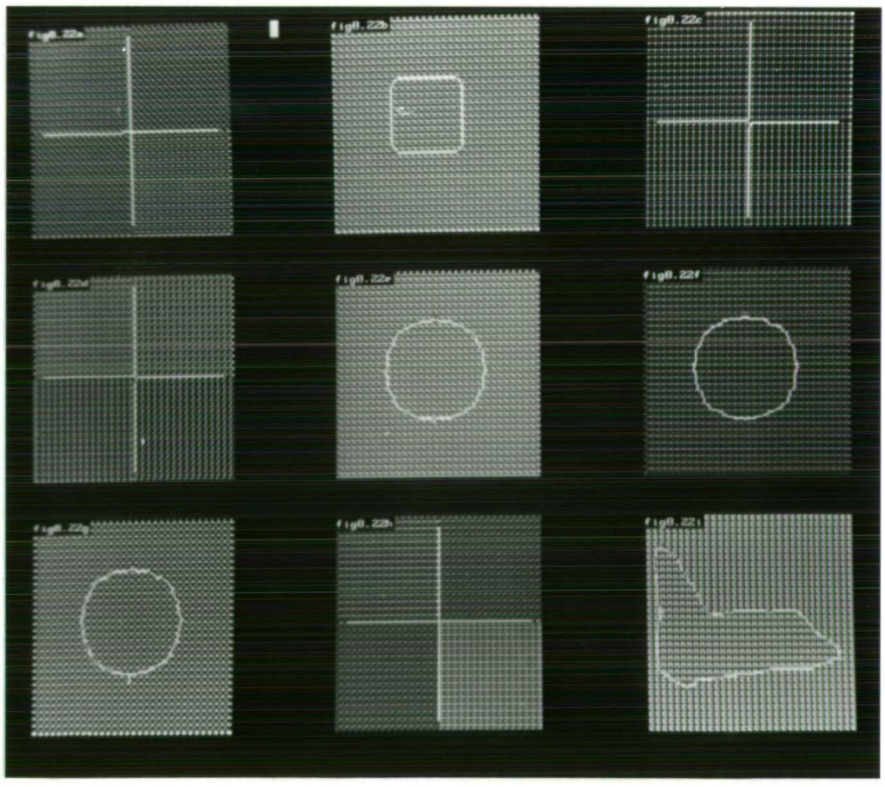


Figure 8.22. The 8 Detected Texture Edge Maps for Fig.8.20(a)-(i) Respectively.

They are all very satisfactory. In fact, for Fig.8.20(h), the identification of texture edges is perfect as shown in Fig.8.22(h).

8.4 Discussions and Conclusions

In this chapter, we have investigated how texture segmentation algorithms can be developed based on the LVM scheme elaborated in Chapter 3. In particular, two UNITRE algorithms and one TEXTED algorithm have been proposed and described.

In the UNITRE algorithms, the capability is assumed of a single LVM or a LVM-LVM (i.e., a SLVM) to reduce statistical complexity, e.g., to transform second-order statistic differences into mean value differences. The basic LVM algorithm involves a single LVM, intensity thresholding, Gaussian low-pass filtering, and gray image segmentation. The successive LVM algorithm is composed of a single SLVM, Gaussian low-pass filtering, and gray image segmentation. Although the two algorithms do not require any *a priori* information on what texture may appear in the input image, their implementation necessitates the pre-specification of the number of uniform texture regions into which the given image is intended to be partitioned. The requisition of the pre-specification of the number of regions is the direct product of the application of the RSST segmentation technique. Thus, the adoption of some other gray image segmentation techniques may enable one to avoid the problem. The major difficulty in the application of the two algorithms is the choice of an effective LVM or SLVM. The local discrete cosine mappings and the local variance mapping defined in Chapter 3 are seemingly good candidates.

In reality, textures can differ in a wide range of attributes, each of which may only partially contribute to texture discrimination. This is the motive for the development of the proposed TEXTED algorithm, the MUCTED algorithm. The algorithm consists of a number of parallel channels each of which differs in the underlying LVM or SLVM and is envisaged as being tuned to a specific textural attribute. Unlike the two UNITRE algorithms, the MUCTED algorithm requires no *a priori* knowledge on the type and number of uniform texture regions in the given image, and hence can be classified as an UTS (unsupervised texture segmentation, see Chapter 7) technique. The human intervention in the implementation of the MUCTED algorithm is mainly due to the determination of the weighting coefficients during channel grouping. For simplicity, a linear model for channel grouping has been assumed. It should also

be pointed out that channel grouping based on this model depends purely on the magnitudes of the channel gradients, and no consideration has been given to the spatial coherence of texture edges being reflected in the channel gradient orientations. Better channel grouping results and thus better texture edge detection can be expected by employing more sophisticated grouping models [9] which take into account both the channel gradient magnitude (describing edge strength) and the channel gradient orientation (describing edge coherence).

In summary, we have developed in this chapter two algorithms for uniform texture region extraction and one algorithm for texture edge detection. All these algorithms are based on the LVM scheme described in Chapter 3. The algorithms have been tested with a variety of textures. Experimental results demonstrate that the algorithms are capable of extracting connected uniform texture regions or detecting connected texture edges of single pixel width with moderate computational burden.

8.5 References

[1] **A.Papoulis**, *Systems and Transforms with Applications in Optics*, McGraw-Hill Book Company, 1968.

[2] **J.G.Daugman**, *Uncertainty Relations for Resolutions in Space, Spatial Frequency and Orientation Optimized by Two-Dimensional Visual Cortical Fields*, J.Opt.Soc.Am. A, vol.2, no.7, 1985, pp.1160-1169.

[3] **O.J.Morris, M.De.J.Lee** and **A.G.Constantinides**, *Graph Theory for Image Analysis: An Approach Based on Shortest Spanning Tree*, Proc. IEE, vol.133, Pt F, 1986, pp.146-152.

[4] **P.Brodatz**, *Textures: A Photographic Album for Artists and Designers*, New York: Reinhold, 1968.

[5] **T.N.Tan** and **A.G.Constantinides**, *LVM Image Texture Analysis: Texture Classification*, being submitted for publication.

[6] **T.N.Tan** and **A.G.Constantinides**, *LVM Image Texture Analysis: Uniform Texture Region Extraction*, being submitted for publication.

[7] **R.C.Gonzalez** and **P.Wintz**, *Digital Image Processing*, 2nd ed., Addison-Wesley Publishing Company, 1987.

[8] **N.J.Naccache** and **R.Shinghal**, *SPTA: A Proposed Algorithm for Thinning Binary Patterns*, IEEE Trans. on Syst. Man and Cybern., vol.SMC-14, 1984, pp.409-418.

[9] **E.E.Triendle** and **T.Henderson**, *A Model for Texture Edges*, Proc. of the 5th Inter. Conf. on Pattern Recognition, Miami Beach, Florida, Dec.1-4, 1980, pp.1100-1102.

[10] **B.Julesz**, *Experiments in The Visual Perception of Texture*, Scientific American, vol.212, 1965, pp.38-54.

[11] **R.Wang** *et al.*, *Texture Analysis Based on Local Standard Deviation of Intensity*, Proc. of the Inter. Conf. on Computer Vision and Pattern Recognition, FL, USA, June 22-26, 1986, pp.482-488.

[12] **P.C.Chen** and **T.Pavlidis**, *Segmentation by Texture Using a Co-occurrence Matrix and a Split-and-Merge Algorithm*, Comput. Graphics and Image Processing, vol.10, 1979, pp.172-182.

[13] **P.C.Chen** and **T.Pavlidis**, *Segmentation by Texture Using Correlation*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.5, 1983, pp.64-69.

[14] **M.Pietikainen** and **A.Rosenfeld**, *Image Segmentation by Texture Using Pyramid Node Linking*, IEEE Trans. on Syst. Man and Cybern., vol.11, 1981, pp.822-825.

9. GABOR FILTERS IN TEXTURE ANALYSIS II: TEXTURE EDGE DETECTION

9.1 Introduction

A fundamental problem in most of the existing texture segmentation approaches, particularly those belonging to FSC, STC, GIS and TGO (For these approaches, see Chapter 7), is the so-called *textural resolution* problem [1], that is the choice of windows of pertinent size within which texture measures are to be computed. When texture features are measured on small windows, they are unreliable, but if we use large windows, it is hard to find regions that are uniformly textured. On the other hand, if we want to segment a given image into differently textured regions, the windows will have to be as small as possible to enable accurate location of the borders between the regions without a blocky outlook. These considerations indicate that the reliability of texture features and the accuracy of texture edge detection are incompatible. This incompatibility can be examined under the concept of *uncertainty principle* in image processing elaborated in [2].

It is the concern of this chapter to develop a TEXTED algorithm which alleviates this incompatibility by means of global spatial filtering. An immediate question to be answered is what form should the underlying spatial filters take in order to produce a compromise (i.e., to minimize the incompatibility) with respect to the two conflicting requirements? Several desirable features can be hypothesized:

- The filters should have good frequency and orientation tunability in order to be able to be tuned to any texture;
- The filters should have narrow frequency and orientation bandwidths (i.e., good spectral localization) in order to be well tuned to a particular texture; and
- the filters should also have limited spatial support (i.e., good spatial localization) in order to cause minimum uncertainty around texture edges.

These three hypothesized filter requirements are satisfied by a family of filters called *Gabor filters* [3,4]. In addition to the features listed above, Gabor filters have also been found to be good models for the receptive field profiles of simple cortical cells as discussed in Chapter 6.

These peculiar properties of Gabor filters are the primary motives for the development of

the TEXTED algorithm to be presented. The algorithm is essentially based on the implementation of a number of cortical channels, a model of which was established in Chapter 6. The number and parameters of the channels to be implemented are automatically determined based on the analysis of the power spectrum of the input image. The efficacy of the proposed algorithm is illustrated using a variety of synthetic, noisy and natural textured images.

The chapter is organized as follows. The principles of the proposed algorithm are first detailed step by step in the next section, followed by the theoretical justification of the appropriateness of the proposed cortical channel model for texture edge detection, and the description of the method for the determination of the number of channels. Section 3 then focuses on the implementation of the algorithm. Numerous experimental results are included in this section. The chapter is concluded by Section 4.

9.2 Texture Edge Detection Via Gabor Filtering

In this section, our attention will be focused on the proposed algorithm itself. A block diagram of the algorithm will be given and the principles of the algorithm will be explained in some detail. The cortical channel model proposed in Chapter 6 on the basis of the analysis of simple cortical cells will be theoretically justified. Meanwhile, the problem of spectral peak identification, an important step in filter and channel design, will also be discussed.

9.2.1 The Principles of The Algorithm

Texture edge detection requires the discrimination of neighboring regions having distinct textures. Although texture edge detection is a very difficult problem in image analysis and processing [5], the human visual system (HVS) is believed to be both effective and efficient for the task. It is almost certain that biological evolution has created highly useful and effective texture discrimination and segmentation processes in the HVS. This suggests that computational texture segmentation including texture edge detection should mimic visual texture segmentation performed by the HVS. As we mentioned, the visual cortex is the most important unit in the HVS. Thus a good understanding of the mechanisms of the visual cortex on visual information processing may greatly facilitate the development of successful algorithms for texture edge detection.

Based on the discussions on the visual cortex and the simple cortical cells in particular, we have proposed a cortical channel model in Chapter 6, which can be described as follows:

$$\begin{aligned} a(x,y) &= h_c(x,y)**p(x,y) \\ b(x,y) &= h_s(x,y)**p(x,y) \\ q(x,y) &= \left[a^2(x,y)+b^2(x,y)\right]^{\frac{1}{2}} \end{aligned} \tag{9.1}$$

where ** denotes 2D linear convolution, $p(x,y)$ and $q(x,y)$ are the channel input and output images, and $h_c(x,y)$ and $h_s(x,y)$ are the symmetrical and anti-symmetrical Gabor filters specified in (6.7) and (6.8). Each channel is associated with two Gabor filters $h_c(x,y)$ and $h_s(x,y)$. These two filters are bandpass in nature, and have very narrow bandwidths corresponding to the very narrow tuning curves of the simple cells in the visual cortex. They are tuned (sensitive) to the same spatial frequency, orientation and direction but differ in phase by approximately 90^0, i.e., form a quadrature filter pair as we discussed in Chapter 6.

In Chapter 6, the visual cortex is regarded as being composed of multiple channels with respect to afferent visual information processing. Each of these channels is described by model (9.1), and is tuned to a different band of spatial frequency, orientation and direction such that the entire set of these channels covers the whole spatial frequency domain. Since distinct texture features on the spatial time domain are certainly reflected in different spatial frequencies, orientations and directions on the spatial frequency domain [6], each of these channels can be said to be sensitive to certain texture characteristics or *textons* [7,8]. As there are hundreds of thousands of cortical cells and thus cortical channels in the visual cortex, the complete implementation of all these channels is obviously out of the question. However, as far as texture edge detection is concerned, one only need to implement those cortical channels which are tuned to some prominent characteristics of the textures in the input image and respond distinctly to different textures. Since a spectral peak on the spatial frequency domain certainly signifies a visible texture feature on the spatial time domain such as structural periodicity and orientation, a channel can be designed for and tuned to each spectral peak. Thus, the number of channels to be implemented can be cut down to the number of spectral peaks. These considerations have motivated the suggestion of a new algorithm for texture edge detection whose block diagram is sketched in Fig.9.1.

The algorithm can be in succession divided into five main stages: Spectral Peak

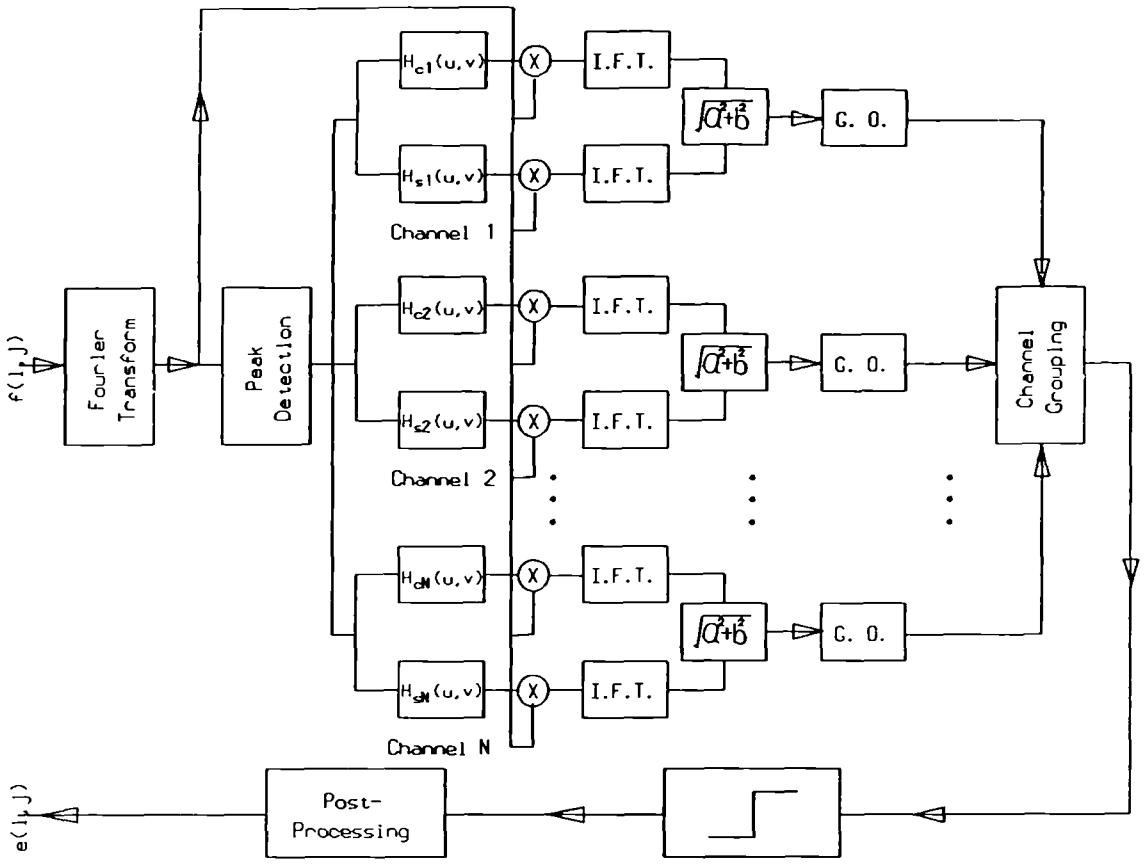


Figure 9.1. Block Diagram of Texture Edge Detection via Gabor Filtering.

Identification, Gabor Filtering, Gradient Calculation, Channel Grouping, and Thresholding and Edge Thinning. In the following, the principles of the proposed algorithm will be explained on a stage-by-stage basis.

☞ Spectral Peak Identification

Since similarity and/or periodicity on the spatial time domain is directly related to the spectral peak distributions on the spatial frequency domain, spectral peak identification is equivalent to the identification of prominent texture features on the spatial time domain such as global structural similarity and regularity. Let $p(x,y)$ be the real input image to be texturally segmented and $P(u,v)$ its Fourier transform. Then the power spectrum $E(u,v)$ of $p(x,y)$ is defined as

$$E(u,v) - P(u,v)|^2 \tag{9.2}$$

where $|\bullet$ denotes taking the modulus of a complex function. It is well known that $E(u,v) = E(-u,-v)$. This implies that if (u_i,v_i) is a peak of $E(u,v)$, $(-u_i,-v_i)$ is also a peak of the same amplitude. These two peaks are apparently associated with the same spatial textural characteristics. Therefore, the identification of independent spectral peaks only needs to be carried out on half of the entire power spectral plane.

As we mentioned, the number of spectral peaks identified determines the number of cortical channels to be implemented. Each channel is accordingly tuned to a specific spectral peak. In particular, suppose the i^{th} channel is tuned to the i^{th} spectral peak located at (u_i,v_i), then the channel will be implemented on the basis of the channel model (9.1) with the two filters $h_c(x,y)$ and $h_s(x,y)$ being specified by

$$h_{ci}(x,y) = g(x,y)\cos[2\pi(u_i x + v_i y)] \tag{9.3}$$

for the symmetrical Gabor filter and

$$h_{si}(x,y) = g(x,y)\sin[2\pi(u_i x + v_i y)] \tag{9.4}$$

for the anti-symmetrical Gabor filter, where $g(x,y)$ in subsequent discussion is assumed to be a 2D Gaussian of unity aspect ratio, i.e.,

$$g(x,y) = e^{-(x^2+y^2)/2\sigma^2} \tag{9.5}$$

The specific method adopted here for spectral peak identification is detailed in Section 9.2.3.

☞ Gabor Filtering

After all perceivable spectral peaks are located, the number of channels and the filter central frequencies are determined, and we can advance to the Gabor Filtering stage, i.e, the implementation of the channel model (9.1) under the determined parameters. To speed up computation, the implementation is conducted in the spatial frequency domain by means of FFT techniques. Let us consider the i^{th} channel. Suppose $p(x,y)$ and $q_i(x,y)$ are the channel input and output images, and $a_i(u,v)$ and $b_i(u,v)$ are the symmetrical and anti-symmetrical filter outputs respectively, then from the channel model (9.1) and by recalling the convolution theorem of linear systems, we have on the spatial frequency domain:

$$\begin{aligned} a_i(x,y) &= F^{-1}[H_{ci}(u,v)P(u,v)] \\ b_i(x,y) &= F^{-1}[H_{si}(u,v)P(u,v)] \\ q_i(x,y) &= \left[a_i^2(x,y)+b_i^2(x,y)\right]^2 \end{aligned} \tag{9.6}$$

where $F^{-1}[\bullet]$ denotes inverse Fourier transform, $P(u,v)$ is the Fourier transform of the input image $p(x,y)$, and $H_{ci}(u,v)$ and $H_{si}(u,v)$ are the Fourier transforms of $h_{ci}(x,y)$ and $h_{si}(x,y)$ defined in (9.3) and (9.4).

Since the i^{th} channel is assumed to be tuned to spectral peak (u_i,v_i), the channel output image $q_i(x,y)$ is expected to exhibit large values in the regions with texture $p_i(x,y)$ which contributes mostly to the spectral peak (u_i,v_i) on the spatial frequency domain. This suggests that after Gabor filtering in the i^{th} channel, the input textured image $p(x,y)$ is transformed into a non-textured gray image $q_i(x,y)$ in which the boundaries separating the regions with texture $p_i(x,y)$ from other regions of the input image are detectable based on first-order intensity statistics such as local intensity gradients. These heuristic arguments will be theoretically justified in Section 9.2.2.

From equations (9.3) to (9.5), it is clear that to uniquely specify a cortical channel, three channel parameters have to be specified: the central frequencies u_0 and v_0, and the space constant σ. The central frequencies are automatically determined at the stage of spectral peak identification whereas the space constant has to be predefined. In this chapter, σ is set to $\sigma = 0.045$ unless otherwise stated. This is the value at which the difference is 20db between the frequency response at the central frequency of $h_c(x,y)$ and $h_s(x,y)$ defined in (9.3) and (9.4) and that at the frequency which is five pixels away from the central frequency.

☞ Gradient Calculation

Let us again focus our attention on the i^{th} channel. After Gabor Filtering, the input textured image $p(x,y)$ is transformed into a gray image $q_i(x,y)$. Thus the texture boundaries between the regions with texture $p_i(x,y)$ and their neighboring regions can be measured by the intensity edges of $q_i(x,y)$. We therefore proceed with intensity gradient calculation on $q_i(x,y)$ to obtain a channel textural gradient image $d_i(x,y)$. This is achieved by applying the conventional Sobel operator given in (8.6) on the channel output image $q_i(x,y)$.

☞ Channel Grouping

Since $d_i(x,y)$ only partially marks the texture edges perceived in the input image $p(x,y)$, channel outputs have to be combined somehow in order to identify all possible texture edges. Let $d(x,y)$ represent the total gradient image, then to compute $d(x,y)$ we assume the following linear model for channel grouping as we did in Chapter 8:

$$d(x,y) = \sum_{i=1}^{N} w_i d_i(x,y) \tag{9.7}$$

where N is the total number of channels, and w_i, $i=1,2,\ldots,N$, are the weighting coefficients. In the experiments presented in Section 9.3, these coefficients are usually set to 1.0 unless otherwise specified.

☞ Thresholding and Edge Thinning

After channel grouping, we proceed with thresholding on the gradient image $d(x,y)$. The output is a binary image $b(x,y)$ which marks all texture edges. However, a simple thresholding operation on $d(x,y)$ usually results in very thick edges. To obtain texture edges of single-pixel width, we post-process the binary image $b(x,y)$ by means of edge thinning. The thinning algorithm used here is the same as used in the preceding chapter, i.e., the SPTA algorithm described in [13]. Since the SPTA requires the input pattern to be smooth, which is not guaranteed by a simple intensity thresholding operation, the binary image $b(x,y)$ is median filtered before edge thinning.

This completes the presentation of the principles of the algorithm.

9.2.2 Theoretical Justification of The Cortical Channel Model

Now, let us turn our attention to the appropriateness of the cortical channel model (9.1). Specifically, we want to show that for a given input textured image $p(x,y)$, the channel output image $q(x,y)$ is a non-textured gray image, in which regions having the texture to which the underlying channel is tuned, have higher values than any other regions. For simplicity, let us assume that all underlying variables are continuous. Suppose the image $p(x,y)$ to be texturally segmented is composed of N texture types $p_1(x,y)$, $p_2(x,y)$, $\cdots$, $p_i(x,y)$, $\cdots$, $p_N(x,y)$. The presence or absence of a particular texture type $p_i(x,y)$ on $p(x,y)$ is indicated by $s_i(x,y)$. If we define the indicator function $s_i(x,y)$ as follows:

$$s_i(x,y) = \begin{cases} 1, & \text{if } (x,y) \in p_i(x,y); \\ 0, & \textit{otherwise}; \end{cases} \tag{9.8}$$

then $p(x,y)$ can be conveniently represented by

$$p(x,y) = \sum_{i=1}^{N} s_i(x,y) p_i(x,y) \tag{9.9}$$

For further simplification, let us assume that texture $p_i(x,y)$ is a pure sinusoidal function located at (u_i,v_i) on the frequency domain. In other words, we assume

$$p_i(x,y) = C_i \cos 2\pi (u_i x + v_i y), \quad i = 1,2,\ldots,N; \tag{9.10}$$

where C_i is a constant. By substituting (9.10) into (9.9), one obtains

$$p(x,y) = \sum_{i=1}^{N} s_i(x,y) C_i \cos 2\pi (u_i x + v_i y) \tag{9.11a}$$

or equivalently,

$$p(x,y) = \frac{1}{2}\sum_{i=1}^{N} C_i s_i(x,y) e^{2\pi j(u_i x + v_i y)} + \frac{1}{2}\sum_{i=1}^{N} C_i s_i(x,y) e^{-2\pi j(u_i x + v_i y)} \tag{9.11b}$$

Let the Fourier transform of the indicator function $s_i(x,y)$ be $S_i(u,v)$, then by recalling (9.11b) and the shifting property of Fourier transform, one can easily show that the Fourier transform $P(u,v)$ of $p(x,y)$ is given by

$$\begin{aligned} P(u,v) &= \frac{1}{2}\sum_{i=1}^{N} C_i S_i(u-u_i, v-v_i) + \frac{1}{2}\sum_{i=1}^{N} C_i S_i(u+u_i, v+v_i) \\ &= \frac{1}{2}\sum_{i=1}^{N} C_i [S_i(u-u_i, v-v_i) + S_i(u+u_i, v+v_i)] \end{aligned} \tag{9.12}$$

which is the sum of the shifted Fourier transforms of the N indicator functions. Since $s_i(x,y)$ is basically a low-pass signal, one can expect N spectral peaks on $P(u,v)$ corresponding to the N texture types $p_i(x,y)$, $i = 1,2,\ldots,N$ respectively. Suppose a spectral peak is identified at (u_k,v_k) on $P(u,v)$ at the peak identification stage, then the underlying texture is $p_k(x,y) = C_k \cos 2\pi (u_k x + v_k y)$, and the corresponding Gabor filter pair is (see equations (9.3) and (9.4)):

$$h_{ck}(x,y) = g(x,y)\cos 2\pi(u_k x + v_k y) \tag{9.13a}$$

and

$$h_{sk}(x,y) = g(x,y)\sin 2\pi(u_k x + v_k y) \tag{9.13b}$$

On the frequency domain, this filter pair is

$$H_{ck}(u,v) = \frac{1}{2}\left[G(u-u_k,v-v_k)+G(u+u_k,v+v_k)\right] \tag{9.14a}$$

and

$$H_{sk}(u,v) = \frac{1}{2j}\left[G(u-u_k,v-v_k)-G(u+u_k,v+v_k)\right] \tag{9.14b}$$

where $G(u,v)$ is the Fourier transform of $g(x,y)$. The outputs from the two filters on the spatial frequency domain are therefore

$$\begin{aligned} A_k(u,v) &= P(u,v)H_{ck}(u,v) \\ &= \frac{1}{4}\left[G(u-u_k,v-v_k)+G(u+u_k,v+v_k)\right]\left[\sum_{i=1}^{N} C_i S_i(u-u_i,v-v_i)+\sum_{i=1}^{N} C_i S_i(u+u_i,v+v_i)\right] \\ &= \frac{1}{4}\Big[\sum_{i=1}^{N} C_i S_i(u-u_i,v-v_i)G(u-u_k,v-v_k)+\sum_{i=1}^{N} C_i S_i(u-u_i,v-v_i)G(u+u_k,v+v_k) \\ &\quad +\sum_{i=1}^{N} C_i S_i(u+u_i,v+v_i)G(u-u_k,v-v_k)+\sum_{i=1}^{N} C_i S_i(u+u_i,v+v_i)G(u+u_k,v+v_k)\Big] \end{aligned} \tag{9.15a}$$

and

$$\begin{aligned} B_k(u,v) &= P(u,v)H_{sk}(u,v) \\ &= \frac{1}{4j}\left[G(u-u_k,v-v_k)-G(u+u_k,v+v_k)\right]\left[\sum_{i=1}^{N} C_i S_i(u-u_i,v-v_i)+\sum_{i=1}^{N} C_i S_i(u+u_i,v+v_i)\right] \\ &= \frac{1}{4j}\Big[\sum_{i=1}^{N} C_i S_i(u-u_i,v-v_i)G(u-u_k,v-v_k)-\sum_{i=1}^{N} C_i S_i(u-u_i,v-v_i)G(u+u_k,v+v_k) \\ &\quad +\sum_{i=1}^{N} C_i S_i(u+u_i,v+v_i)G(u-u_k,v-v_k)-\sum_{i=1}^{N} C_i S_i(u+u_i,v+v_i)G(u+u_k,v+v_k)\Big] \end{aligned} \tag{9.15b}$$

respectively.

As we discussed in Chapter 6 (see equation (6.16)), the overlap between $G(u-u_k,v-v_k)$ and $G(u+u_k,v+v_k)$ is negligible under certain conditions which are typically true in texture edge detection. If the two spatial frequencies (u_i,v_i) and (u_j,v_j) are sufficiently separated from each other on the spatial frequency domain, then the overlaps between $S_i(u\pm u_i,v\pm v_i)$ and

$S_j(u \pm u_j, v \pm v_j)$ would also be negligible for $i \neq j$. Under these conditions, we have

$$\begin{aligned} &G(u \pm u_k, v \pm v_k)S_i(u \pm u_i, v \pm v_i) \approx 0, \quad i \neq k; \\ &G(u \pm u_k, v \pm v_k)S_i(u \mp u_i, v \mp v_i) \approx 0, \end{aligned} \tag{9.16}$$

By substituting (9.16) into (9.15a) and (9.15b), we can simplify the filter outputs as

$$\begin{aligned} A_k(u,v) \approx \frac{C_k}{4}\big[&S_k(u-u_k, v-v_k)G(u-u_k, v-v_k) \\ &+ S_k(u+u_k, v+v_k)G(u+u_k, v+v_k)\big] \end{aligned} \tag{9.17a}$$

and

$$\begin{aligned} B_k(u,v) \approx \frac{C_k}{4j}\big[&S_k(u-u_k, v-v_k)G(u-u_k, v-v_k) \\ &- S_k(u+u_k, v+v_k)G(u+u_k, v+v_k)\big] \end{aligned} \tag{9.17b}$$

Since $S_k(u,v)G(u,v)$ and $s_k(x,y)*g(x,y)$ form a Fourier transform pair, then by recalling the shifting property of the Fourier transform and taking the inverse Fourier transform upon (9.17a) and (9.17b), one can immediately write the filter outputs on the spatial time domain as:

$$a_k(x,y) \approx \frac{C_k}{2}\big[s_k(x,y)*g(x,y)\big]\cos 2\pi(u_k x + v_k y) \tag{9.18a}$$

and

$$b_k(x,y) \approx \frac{C_k}{2}\big[s_k(x,y)*g(x,y)\big]\sin 2\pi(u_k x + v_k y) \tag{9.18b}$$

Therefore the channel output image $q_k(x,y)$ associated with the spectral peak (u_k, v_k) can be calculated as

$$\begin{aligned} q_k(x,y) &= \big[a_k^2(x,y) + b_k^2(x,y)\big]^{1/2} \\ &\approx \Big[\frac{C_k^2}{4}\big[s_k(x,y)*g(x,y)\big]^2 \cos^2 2\pi(u_k x + v_k y) \\ &\quad + \frac{C_k^2}{4}\big[s_k(x,y)*g(x,y)\big]^2 \sin^2 2\pi(u_k x + v_k y)\Big]^{1/2} \\ &\approx \frac{C_k}{2} s_k(x,y)*g(x,y) \end{aligned} \tag{9.19}$$

which is a smoothed and scaled version of the indicator signal $s_k(x,y)$ defined in (9.8) since $g(x,y)$ is a low-pass Gaussian filter. Clearly, $q_k(x,y)$ will take large values (high intensities)

within $p_k(x,y)$ regions whereas small values elsewhere. If a gradient operator such as the Sobel operator is applied on $q_k(x,y)$, the gradient image $d_k(x,y)$ will have large values around the boundaries between $p_k(x,y)$ regions and other regions. Based on (9.19), we can now conclude that the cortical channel model (9.1) is appropriate for texture edge detection.

(9.19) also tells us that in order to accurately identify texture edges, $g(x,y)$ should have a small support on the spatial time domain corresponding to small space constants σ_x and σ_y. In other words, it should cause only a small amount of filtering on $s_k(x,y)$ and have a good spatial localization. On the other hand, $g(x,y)$ should have a good spectral localization corresponding to large space constants σ_x and σ_y in order to assure each pair of $h_{ck}(x,y)$ and $h_{sk}(x,y)$ defined in (9.13a) and (9.13b) is well tuned to the underlying texture $p_k(x,y)$. It is well known, however, that the spatial localization and the spectral localization are constrainted by the *uncertainty principle* described in (6.24). A Gaussian function $g(x,y)$, being a special Gabor function located at the origin (0,0) on the spatial frequency domain, optimizes the uncertainty principle.

This concludes the justification of the cortical channel model given in (9.1).

9.2.3 Spectral Peak Identification

From Fig.9.1, one knows that the very first step in the proposed algorithm is the selection of cortical channels which requires the determination of the number of channels and the specification of channel parameters (i.e., channel tuning). This is achieved by means of spectral peak identification. Let us now focus our attention on the technique for 2D spectral peak identification. What is a peak in a 2-D data array? An immediate answer would say that a peak is a data point having the global maximum in the underlying 2-D data array. However, we shall define a two-dimensional peak as a point whose local neighborhood has the global maximum density (i.e., the sum of the data in the neighborhood divided by the number of data points). Hence, a spectral peak is a point on the spatial frequency plane whose local neighborhood possesses the global maximum spectral density. This kind of peak definition is desirable in our work as the Fourier spectrum of a real-world textured image usually does not show discrete peaks or spectral spikes. More importantly, such a definition minimizes the probability of misidentifying a peak due to noise as a true spectral peak.

Then how can we effectively locate all perceivable spectral peaks? Or what should the

spectral peak identification method be? Several desirable features can be hypothesized:

- The method should be capable of detecting global peaks as the definition of spectral peaks requires;
- The method should be capable of locating peaks very accurately as the peak identification accuracy determines the channel tuning accuracy; and
- The method should be insensitive to noise as the Fourier power spectra of realistic images are usually noisy.

The simple thresholding technique clearly does not satisfy these requirements. The *Converging Squares Algorithm (CSA)* proposed by **O'Gorman** and **Sanderson** [9], however, has proven to satisfy all these requirements. This algorithm has been therefore employed in our work. Nevertheless, the CSA is only capable of detecting one peak at a time. In order to identify all perceivable spectral peaks, the iterative applications of the CSA are necessary. Given a power spectrum array $E(u,v)$, we apply the CSA to identify the first spectral peak p_1. In order to facilitate the second iteration, the first peak p_1 should be erased from $E(u,v)$. So, within a small neighborhood of the first peak p_1, $E(u,v)$ is set to zero. The iteration is then continued until the amplitude of the current peak is less than the first peak by a pre-specified ratio. Fig.9.2 illustrates the iterative applications of the CSA in identifying all spectral peaks. In this chapter, the peak erasing windows and the peak ratio have been set as 7x7 windows and 80% respectively.

9.3 Experimental Results

In this section, experimental results will be described and discussed. To test the general applicability of the proposed algorithm, we have applied both artificial and natural textures. All testing images are of size 128x128 and represented by 8 bits. All component textures have been generated or pre-processed to have similar average gray intensities in order to avoid texture segmentation/discrimination based on gray intensity differences. The experimental results are presented according to the type of the testing images used.

9.3.1 Segmentation of Artificial Textures

Artificial textures have been extensively used in the human vision research community

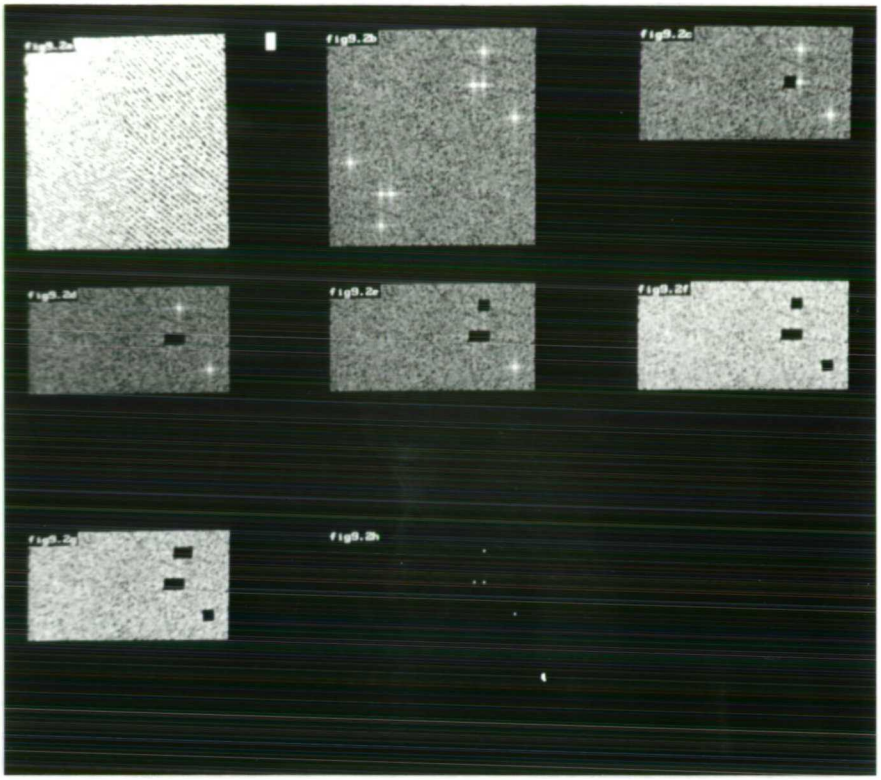
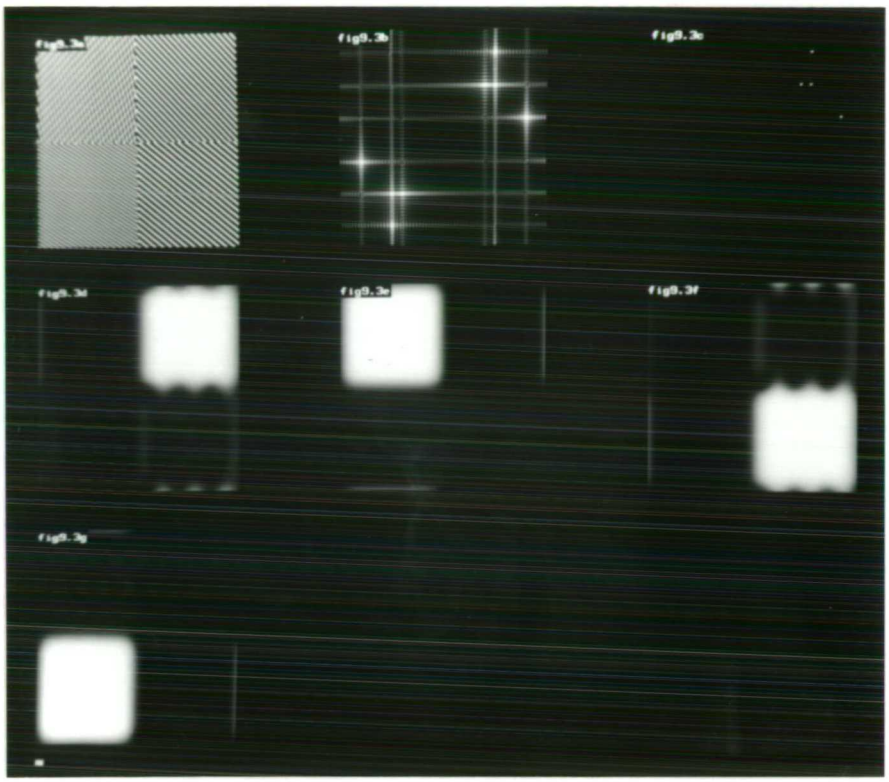


Figure 9.2. Illustration of Iterative Identification of Spectral Peaks. (a) Original Image; (b) Its Power Spectrum; (c) Identification of the First Spectral Peak (After the First Iteration); (d) After the Second Iteration; (e) After the Third Iteration; (f) After the Fourth Iteration; (g) After the Fifth Iteration and (h) the Four Peaks Identified (shown in white dots).

which is concerned with human visual perception and processing of textures. Although artificial textures are of course unrealistic, their applications are, however, much more illustrative for a particular theory or an algorithm. We start with texture edge detection in artificial textures.

Fig.9.3 presents the first example. The original textured image shown in Fig.9.3(a) is composed of four textures. Each component texture fills one quadrant of the 128x128 input image. Fig.9.3(b) is the power spectrum of the original image. Because of symmetry, peak



(To be continued on the next page)

identification is performed only on the top half of Fig.9.3(b) as we discussed. With the origin of the spectral plane being the center of Fig.9.3(c), four peaks have been identified located respectively at (-32,26), (-51,32), (-13,51) and (-32,32) as shown in Fig.9.3(c) (white dots). The correspondence of these peaks to the four component textures need not be known and is of no importance in the proposed algorithm whereas the knowledge about this correspondence is essential in similar approaches by others [10]. Fig.9.3(d) to (g) show the outputs from the underlying four channels after Gabor filtering (i.e., after the implementation of the channel model (9.1)). It is clear that each channel is only particularly sensitive to one quadrant, and the textural differences have been effectively mapped into large intensity differences. Fig.9.3(h) to (k) give the four channel gradient images by applying a Sobel operator on Fig.9.3(d) to (g) respectively. After channel grouping (i.e., after the implementation of equation (9.7)),

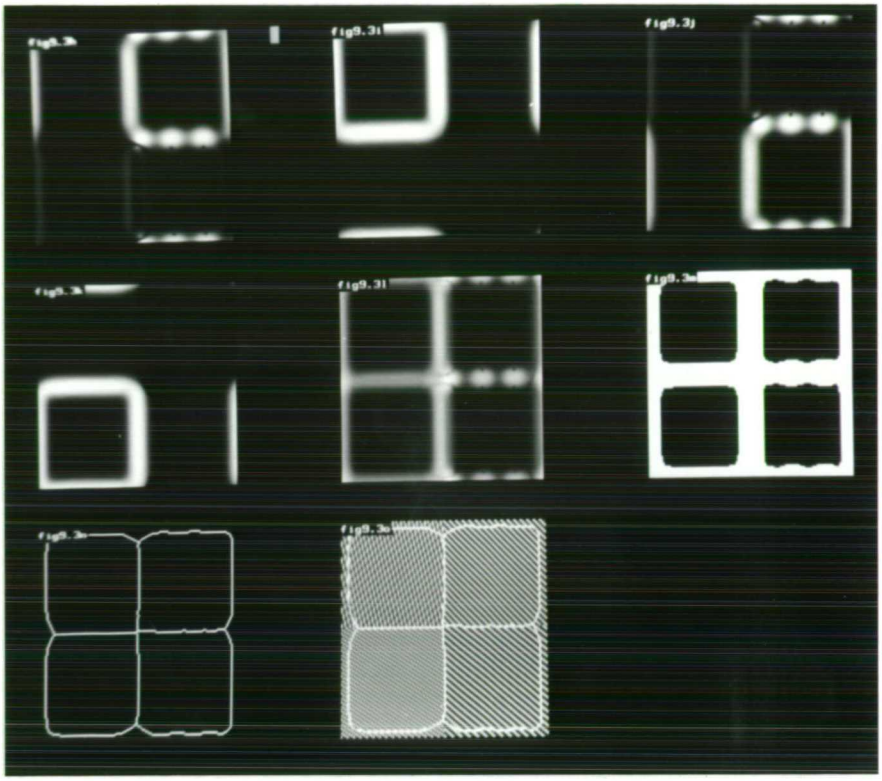


Figure 9.3. Segmentation of Artificial Textures. (a) Original image; (b) Its power spectrum; (c) The four spectral peaks detected; (d)-(g) The four channel output images after Gabor filtering; (h)-(k) The four channel texture gradient images; (l) Total gradient image; (m) After thresholding (l); (n) After thinning (m) and (o) Edge map superimposed on (a).

Fig.9.3(l) is obtained which measures all possible textural changes. Fig.9.3(m) shows the binary image by thresholding Fig.9.3(l) at 90 whereas Fig.9.3(n) is the final texture edge map generated from the algorithm. To show the accuracy of the detected edges, the edge map is overlaid onto the original image. This is shown in Fig.9.3(o). The accuracy of the edges detected are self-evident in Fig.9.3(o).

Fig.9.4 shows another example.

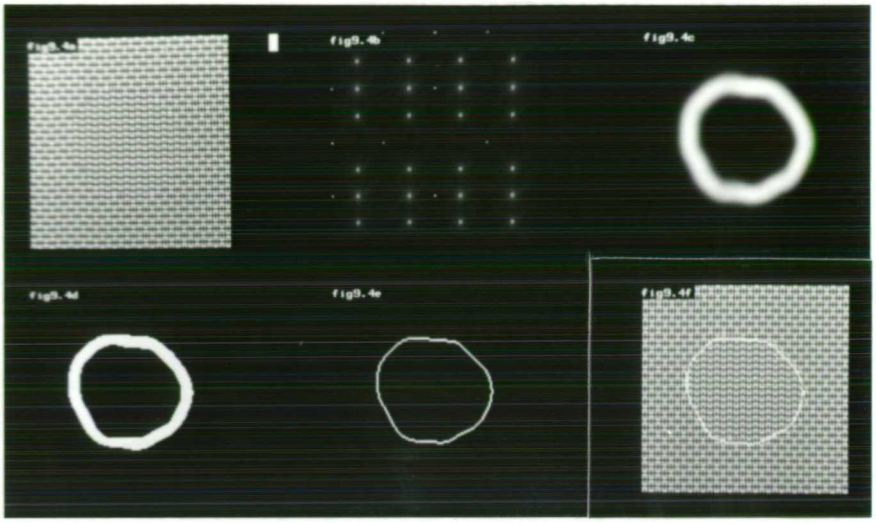


Figure 9.4. Segmentation of Artificial Textures. (a) Original image; (b) Its power spectrum; (c) The total texture gradient image; (d) After thresholding (c); (e) After thinning (d) and (f) Edge map superimposed on (a).

The original textured image Fig.9.4(a) consists of two texture regions: the central region and the background region. In the central region, image rows are identical and are composed of alternating up and down Us whereas in the background region, the Us in adjacent rows are oriented in opposite directions. This image has also been used by **Julesz** *et al* [8] and **Beck** *et al* [11] in their studies on human visual perception of textures. They suggested that preattentive texture discrimination occurs as a result of the differences in the first-order statistics of local features of texture elements. In particular, the discrimination of the two texture regions in Fig.9.4(a) is also a result of local linking by proximity and similarity such as closure and continuation as argued in [11]. However, the accurate identification of the textural boundary in Fig.9.4(a) by a human observer is possible only through element-by-element scrutinization, and may take him a few seconds. Fig.9.4(b) is the power spectrum of Fig.9.4(a). Note, even though only two distinct textures are presented in Fig.9.4(a), the power spectrum shows 12 spectral peaks. This implies that more than one channel would have been tuned to a single texture.

Fig.9.4(c) is the output (the total gradient image) from channel grouping. Fig.9.4(d) and (e) display the edge map before and after edge thinning respectively. In Fig.9.4(f), the edge map is superimposed on the original image. The texture edges detected are clearly consistent with the results of human scrutinization.

To show the segmentation resolution of the proposed algorithm, a textured image with very complicated texture edges is used as our third example, and is given in Fig.9.5.

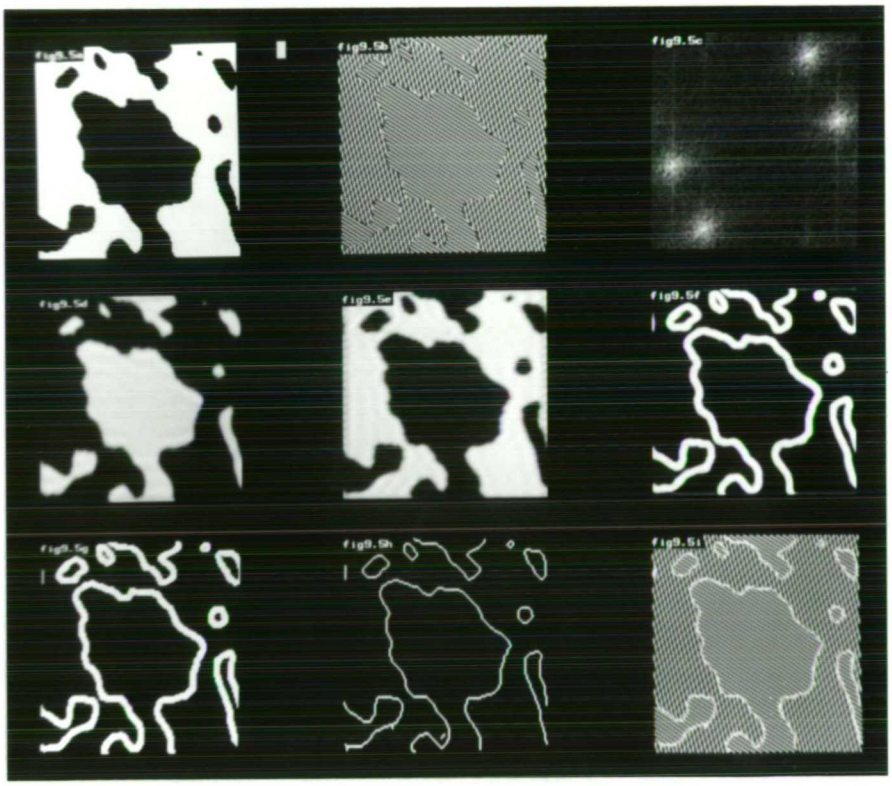


Figure 9.5. Illustration of the Segmentation Resolution of the Proposed Algorithm. (a) A binary image; (b) Its textured version; (c) Power spectrum of (b); (d)-(e) The two channel output images after Gabor filtering; (f) The total gradient image; (g) After thresholding (f); (h) After thinning (g) and (i) Edge map superimposed on (b).

The original textured image Fig.9.5(b) is generated from a binary image Fig.9.5(a). This binary image contains black regions of sizes ranging from only a few pixels to hundreds of pixels. To obtain a textured version, we fill the white and black regions with two different artificial textures resulting in Fig.9.5(b). The power spectrum of Fig.9.5(b) is given in Fig.9.5(c) where two spectral peaks can be observed. The output from the peak identification stage is the two peaks located at (-51,32) and (-13,51) on the spectrum plane. The outputs from the two underlying channels after Gabor filtering are displayed in Fig.9.5(d) and (e). The two texture regions in Fig.9.5(b) have clearly been represented by two different gray intensities. Fig.9.5(f) and (g) are the total gradient image before and after intensity thresholding. The thinned texture edge map is Fig.9.5(h). Except for a small surplus line segment on the upper left border, the edge map is remarkably accurate. The edges associated with the very small black region around the upper right corner in Fig.9.5(a) have also been accurately identified. The high segmentation accuracy and resolution of the proposed algorithm is self-evident in Fig.9.5(i).

A similar example to Fig.9.5 is presented in Fig.9.6. The original textured image Fig.9.6(c) is obtained by segmenting a gray head-and-shoulder type image Fig.9.6(a) into three regions (background, face and neck, and head) as shown in Fig.9.6(b), and by filling the three regions with three distinct textures. Three spectral peaks are detected on the power spectrum Fig.9.6(d). The three channel outputs after Gabor filtering are given in Fig.9.6(e) to (g) respectively. Fig.9.6(h) is the gradient image after channel grouping. Fig.9.6(i) is the final edge map, and Fig.9.6(j) is the edge map superimposed on the original textured image Fig.9.6(c).

9.3.2 Segmentation of Noisy Textures

To test the workability of the proposed algorithm under noisy environments, we have employed a textured image containing noisy textures. The results are summarized in Fig.9.7. The original textured image is Fig.9.7(a). It contains four noisy textures whose layout is similar to those in Fig.9.3. The visual differences between these textures are all very subtle. The power spectrum of the input image, as we expected, is extremely noisy as given in Fig.9.7(b). Nevertheless, four spectral peaks are still visually perceivable. The peak identification stage has successfully identified these four peaks with locations at (-32,26), (-51,32), (-13,51) and (-32,32) respectively. The outputs from the four channels after Gabor filtering are given in Fig.9.7(c) to (f) respectively. Each channel responds strongly only to one texture region.

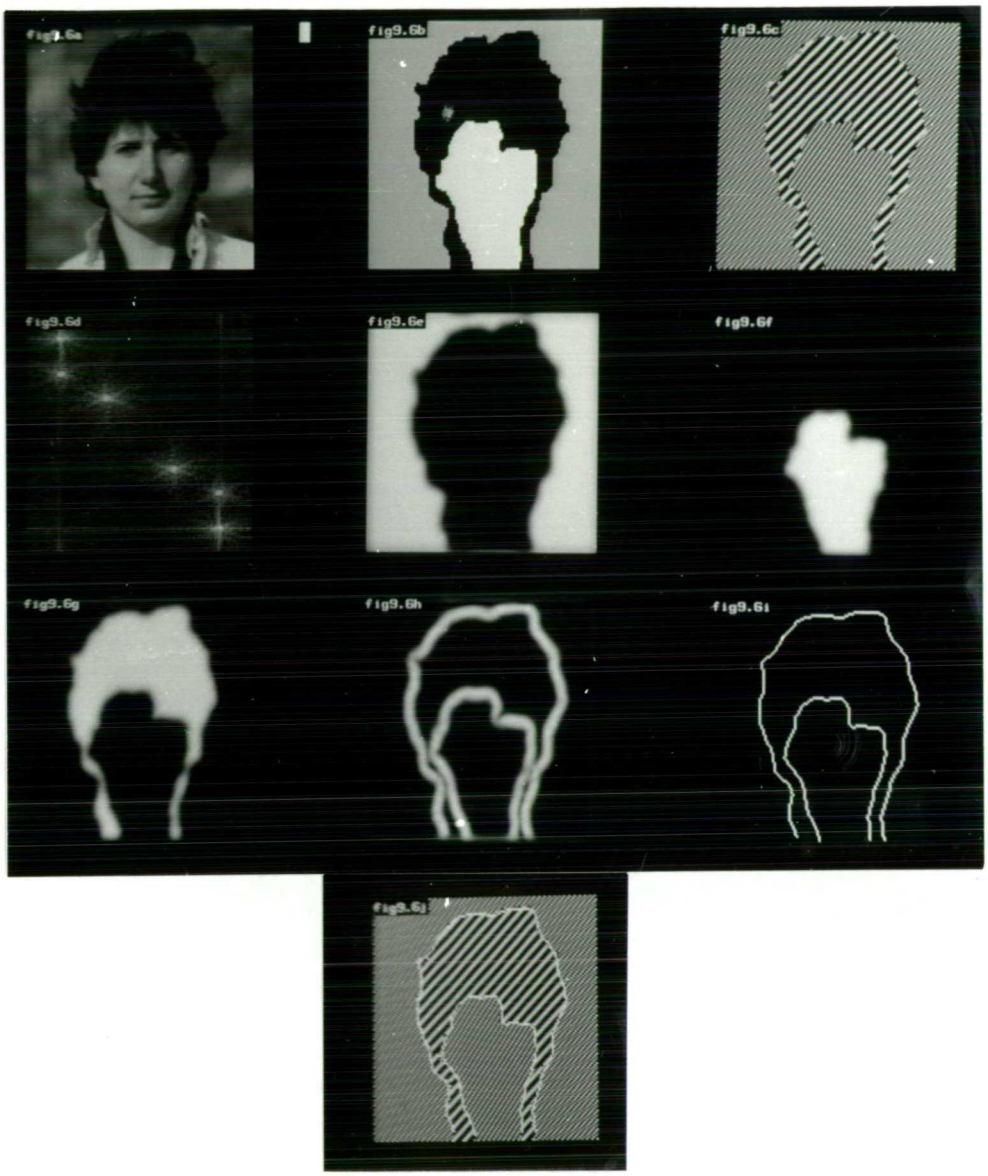


Figure 9.6. Segmentation of Artificial Textures. (a) A gray image; (b) Its three-region segmentation; (c) A textured version of (b); (d) Power spectrum of (c); (e)-(g) The three channel output images after Gabor filtering; (h) The total texture grdaient image; (i) Texture edges detected and (j) Edge map superimposed on (c).

Fig.9.7(g) is the total gradient image from channel grouping. Fig.9.7(h) and (i) show the texture edges detected before and after edge thinning. The edge map Fig.9.7(i) is

superimposed on the original image Fig.9.7(a) to obtain Fig.9.7(j). The workability of the algorithm under noisy environments is evident from Fig.9.7(j).

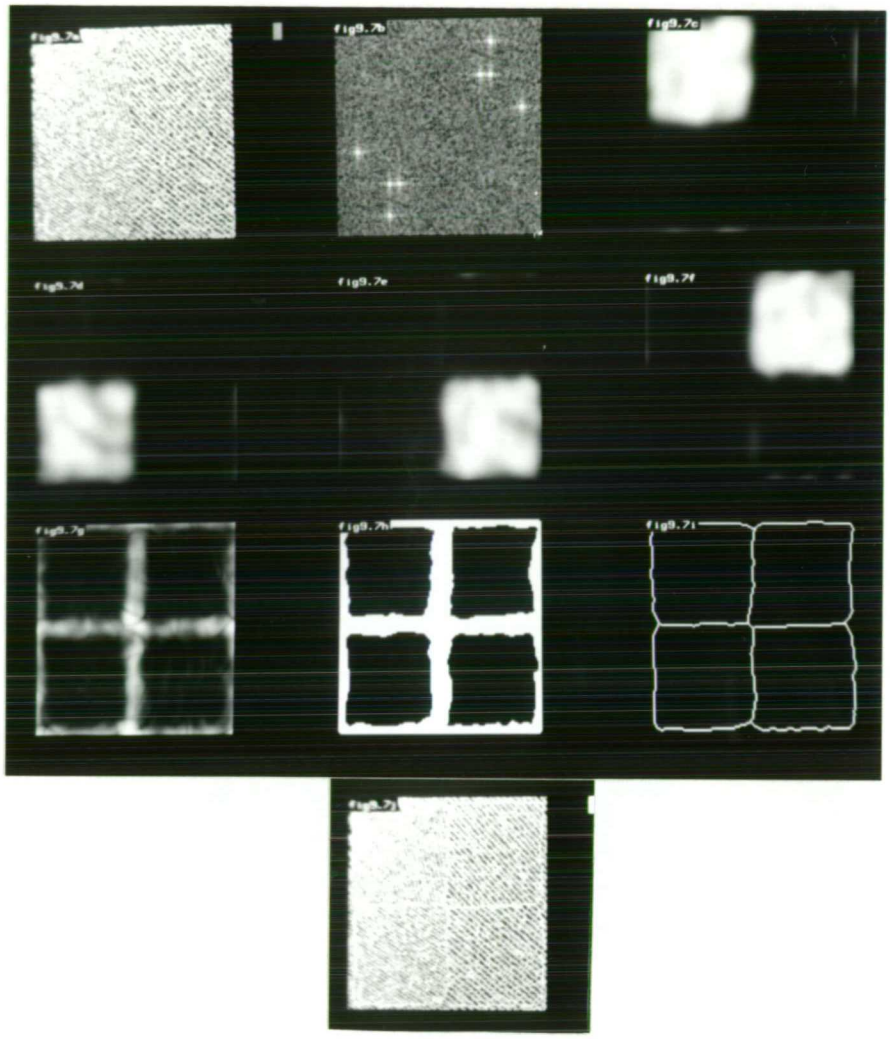


Figure 9.7. Segmentation of Noisy Textures. (a) Original image; (b) Its power spectrum; (c)-(f) The four channel output images after Gabor filtering; (g) The total grdaient image; (h) After thresholding (g); (i) After thinning (h) and (j) Edge map superimposed on (a).

9.3.3 Segmentation of Natural Textures

In the preceding two sub-sections, all textures used were artificial. In this sub-section, our attention will be focused on the applicability of the proposed algorithm under more realistic environments. All textures used can be seen in the real world, and are taken from the photographic album of natural textures by **Brodatz** [12].

Fig.9.8 presents the first example of natural texture segmentation. The original textured image Fig.9.8(a) is a composite of two natural textures: Oriental Straw Cloth (d53) in the left half and Netting (d34) in the right half. Its power spectrum is given in Fig.9.8(b). As expected, most of the image energy is concentrated in low frequency bands, and spectral peaks are not as easily definable as in the power spectra of artificial textures used in the preceding sub-sections. Four spectral peaks have been identified at the spectral peak identification stage. They are located at (-18,0), (0,-9), (0,9) and (-8,-1) respectively. Note, peak (0,-9) and peak (0,9) clearly correspond to the same textural characteristics on the spatial time domain. The channel outputs after Gabor Filtering are given in Fig.9.8(c) to (f). We know by visual inspection on Fig.9.8(a) that the peak at (-18,0) must correspond to texture d53 in the left half as d53 contains parallel horizontally oriented blobs. We thus expect that the underlying channel output will exhibit strong responses in the left half of the original image. This is indeed so as shown in Fig.9.8(c). On the other hand, Fig.9.8(d) to (f) tell us that the peaks at (0,-9), (0,9) and (-8,-1) are related to texture d34. After channel grouping, we obtain the gradient image shown in Fig.9.8(g). Fig.9.8(h) and (i) then present the texture edges detected before and after the edge thinning operation. Fig.9.8(j) illustrates the accuracy of the detected edges. It should be noticed that the left and right image borders have also been identified.

The second example is presented in Fig.9.9. Two texture regions can be seen in the original image Fig.9.9(a). The central region is filled with Herringbone Wave (d17) whereas the background with Cotton Canvas (d77). In this example, only one spectral peak located at (-16,-10) on the power spectrum plane Fig.9.9(b) has been identified at the peak identification stage. Thus, only one cortical channel has been selected and implemented. Fig.9.9(c) is the result after Gabor Filtering from which we know the spectral peak is associated with the background texture Cotton Canvas. Fig.9.9(d) is the output of the Sobel operator applied on the transformed image Fig.9.9(c). The final edge map is given in Fig.9.9(e). The output from the

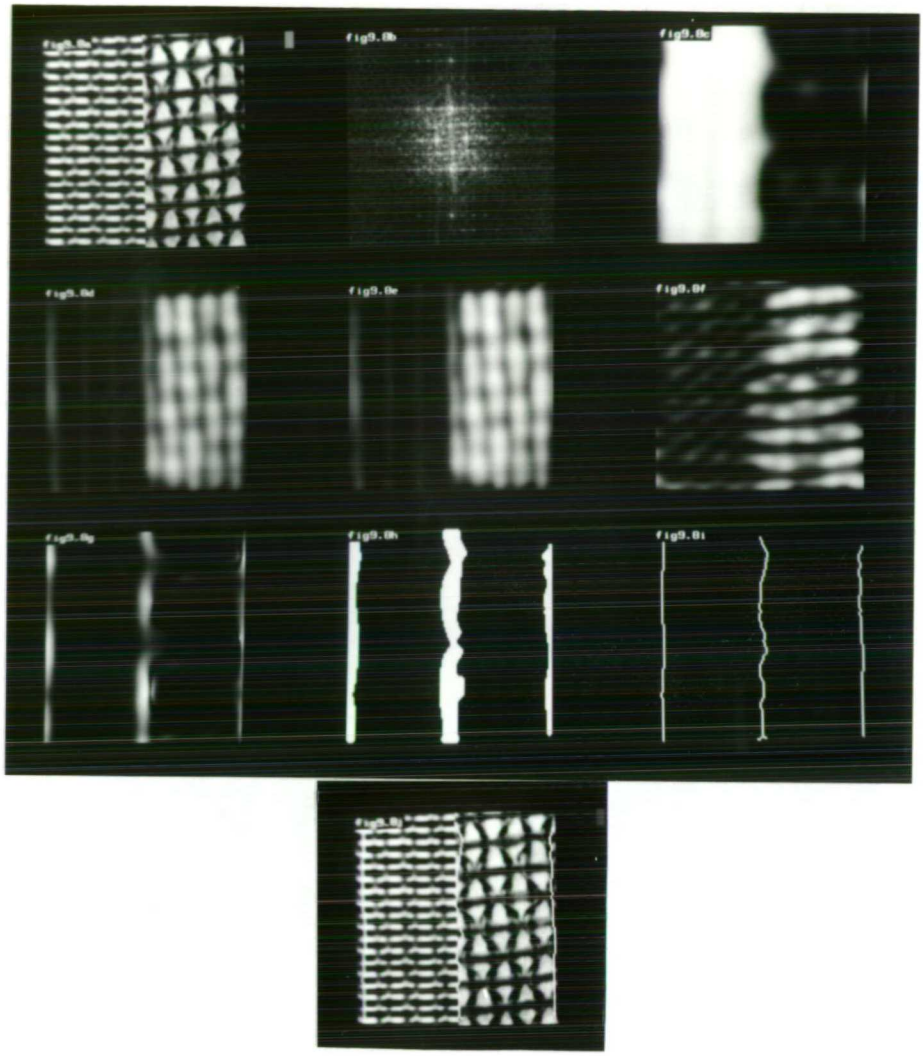


Figure 9.8. Segmentation of Natural Textures (Oriental Straw Cloth and Netting). (a)-(j) are analogous to Fig.9.7(a)-(j).

algorithm is acceptable as can be seen in Fig.9.9(f). Again, the left image border and part of the right image border are also identified.

Fig.9.10 presents the third example. The original textured image Fig.9.10(a) contains four textured regions filled with two natural textures: Cotton Canvas (d77) and Wood Grain (d68).

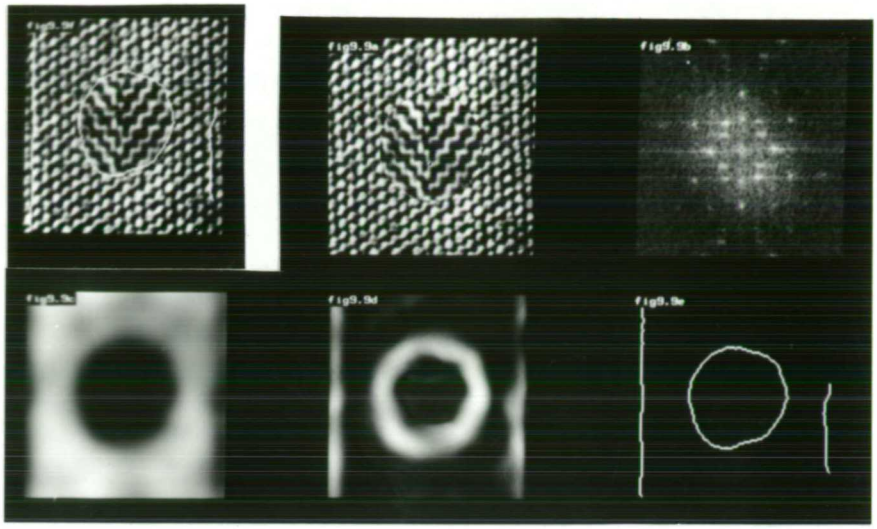


Figure 9.9. Segmentation of Natural Textures (Cotton Canvas and Herringbone Wave). (a) Original textured image; (b) Its power spectrum; (c) The channel output image after Gabor filtering; (d) The texture gradient image; (e) Texture edges detected and (f) Edge map superimposed on (a).

The power spectrum of the input image is shown in Fig.9.10(b). Three spectral peaks are identified. They are located at (-16,-10), (1,-16) and (0,-12) respectively. However, only the channel associated with peak (-16,-10) is found to show very distinct responses to the two different textures. The total gradient image from the three channels after channel grouping is Fig.9.10(c). After thresholding the gradient image, the binary image Fig.9.10(d) results. Fig.9.10(e) presents the final edge map. The accuracy of the edges detected is illustrated in Fig.9.10(f). Although the results are good enough, errors do occur especially around the center of the input image where sharp textural edge transitions exist.

The final example of texture edge detection in natural textures is displayed in Fig.9.11. Fig.9.11(a) to (f) are analogous to Fig.9.10(a) to (f). The textured image contains two natural textures: Oriental Straw Cloth (d53) in the space-shuttle region and Handwoven Oriental Rattan (d65) in the background. The accuracy of the detected texture edges (outline of the space-shuttle) is self-evident in Fig.9.11(f).

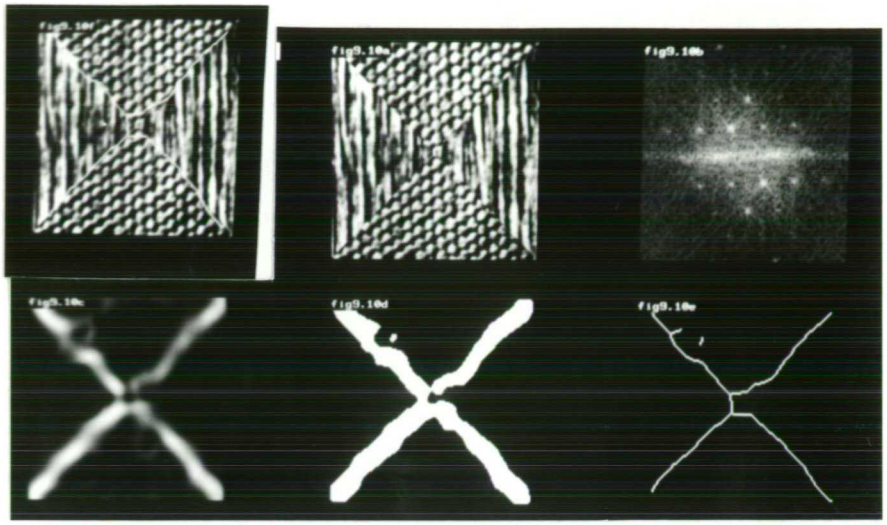


Figure 9.10. Segmentation of Natural Textures (Wood Grain and Cotton Canvas). (a) Original textured image; (b) Its power spectrum; (c) The total texture gradient image; (d) After thresholding (c); (e) After thinning (d) and (f) Edge map superimposed on (a).

This completes the presentation of some of the experimental results of the implementation of the proposed algorithm.

9.4 Conclusions

In this chapter, we have described a new algorithm for texture edge detection. The structure of the algorithm is multichannel-like. For a given input image, the number of channels implemented is set to be the number of spectral peaks identified on the spectral plane, and each channel is tuned to a specific spectral peak. The core operation in each channel is the implementation of the cortical channel model specified in (9.1). The overall algorithm can be functionally distinguished into five stages namely spectral peak identification, cortical channel model implementation (Gabor filtering), channel textural gradient calculation, channel grouping, and thresholding and edge thinning. The principles of the proposed algorithm have been detailed stage by stage. The effectiveness of the algorithm for texture edge detection has

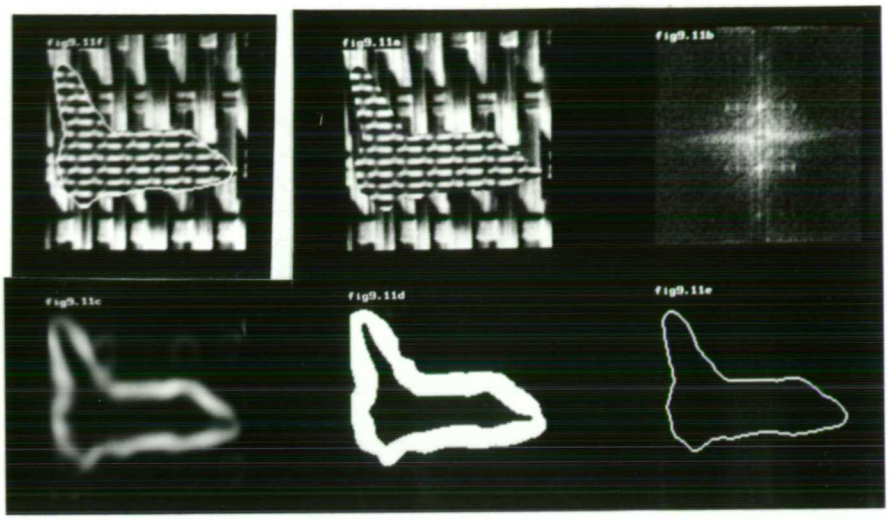


Figure 9.11. Segmentation of Natural Textures (Oriental Straw Cloth and Oriental Rattan). (a)-(f) are analogous to Fig.9.10(a)-(f).

been demonstrated by numerous successful implementation results.

However, it should be pointed out, before concluding this chapter, that the algorithm has implicitly assumed that the textures involved differ from each other in spectral peak distributions. If the power spectrum of the input image shows no perceivable peaks, then the algorithm may not be workable. Therefore, the algorithm may not be suitable for segmenting random and highly irregular textures as the spectral peaks are usually not observable under these circumstances. In addition, the phase information of the input image has been ignored during Gabor filter design, thus the algorithm may also be inappropriate for segmenting textures differing only in phase.

Although the algorithm requires no *a priori* knowledge on the number of uniformly textured regions and the characteristics of region textures, it is incomplete in the sense that there are several parameters which have to be pre-specified. Further improvement on the algorithm can be made in this direction.

9.5 References

[1] **W.B.Thompson,** *Textural Resolution*, Proc. of 3rd Int. Conf. on Pattern Recognition, Coronado, Calif., Nov.8-11, 1976, pp.283-285.

[2] **R.Wilson** and **G.H.Granlund,** *The Uncertainty Principles in Image Processing*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.6, 1984, pp.758-767.

[3] **D.Gabor,** *Theory of Communication*, J.Inst.Elec.Eng., vol.93, part III, 1946, pp.429-459.

[4] **J.G.Daugman,** *Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters*, J Opt Soc Am A, vol.2, 1985, pp.1160-1169.

[5] **L.Van Gool** *et al*, *SURVEY: Texture Analysis Anno: 1983*, Comput. Vision, Graphics and Image Proc., vol.29, 1985, pp.336-357.

[6] **H.C.Nothdurft,** *Texton Properties Associated with Differences in Spatial-Frequency Distribution*, Proc. of 11th European Conf. on Visual Perception, Bristol, England, Aug.31-Sept. 3, 1988.

[7] **M.R.Turner,** *Texture Discrimination By Gabor Functions* Biol Cybern, vol.55, 1986, pp.71-82.

[8] **B.Julesz** *et al*, *Textons, the Fundamental Elements in Preattentive Vision and Perception of Textures*, The Bell Syst Tech J, vol.62, 1983, pp.1619-1645.

[9] **L.O'Gorman** and **A.C.Sanderson,** *The Converging Squares Algorithm: An Efficient Method for Locating Peaks in Multidimensions*, IEEE Trans. on Pattern Anal. Mach. Intell., vol.6, 1984, pp.280-288.

[10] **M.Clark** *et al*, *Texture Segmentation Using Gabor Modulation/Demodulation*, Pattern Recognition Letters, vol.6, 1987, pp.261-267.

[11] **J.Beck** *et al*, *A Theory of Textural Segmentation*, in *Human and Machine Vision* edited by J.Beck et al., New York: Academic Press, 1983.

[12] **P.Brodatz,** *Textures: A Photographic Album for Artists and Designers*, Dover, New York, 1966.

[13] **N.J.Naccache** and **R.Shinghal**, *SPTA: A Proposed Algorithm for Thinning Binary Patterns*, IEEE Trans. on Syst. Man and Cybern., vol.SMC-14, 1984, pp.409-418.

10. CONCLUSION

10.1 Summary of The Thesis

Texture is an ubiquitous image characteristic. Texture analysis is often an indispensable part of low-level machine/computer vision and image understanding systems. Texture classification and segmentation are the two fundamental problems in texture analysis. This thesis has been concerned with the development of efficient and effective algorithms for texture feature extraction and classification, and texture segmentation. It is now commonly recognized that a very wide range of industry can benefit from texture analysis. Nevertheless, before any practical application is possible, dedicated and easily implementable algorithms for texture analysis should be readily available. This thesis is devoted to the progress in this direction.

In the past two decades, a considerable amount of attention has been directed towards texture analysis. Most of the effort, however, has been concentrated on texture feature extraction and classification. An updated survey on the various existing approaches towards texture feature extraction and classification was given in Chapter 2. These approaches, being mostly *ad hoc* in nature, have been reviewed and discussed under three major categories namely Classical Statistical Approaches, Structural Approaches and Model-Based Approaches. The first category includes gray level dependence matrix (GLDM), generalized spatial dependence matrix (GSDM), histogram and autocorrelation function analysis (HAFA), Fourier power spectrum analysis (FPSA), gray level difference approach (GLDA), gray level edge element analysis (GLEA), gray level run length (GLRL), gray level profile analysis (GLPA), and local linear convolution method (LLCM); the second group consists of strong texture analysis, weak texture analysis and syntactic texture analysis; while the approaches reviewed under the third category are autoregressive model (ARM), Gaussian Markov random field model (GMRFM), Gibbs random field model (GRFM), mosaic random field model (MRFM), random walk model (RWM), fractal surface model (FSM) and multichannel spatial filtering model (MSFM). The principles, merits and demerits of these approaches have been outlined. Among all these approaches, the GLDM is by far the most popular and widely employed approach. In fact, it has been for long the benchmark for texture analysis (texture feature extraction in particular) approaches. The appropriateness of the GLDM approach for texture discrimination is justified by the experimental finding that the human visual discrimination of textures is primarily

invoked by the differences in second-order statistics. Although the GLDM approach has been in general effective, its popularity is overshadowed by its high computational complexity.

Unlike intensity, texture is a local regional image characteristic in that it can not be perceived at a single pixel or only a few pixels. Studies in physiology and psychophysics suggest that early stage texture perception or preattentive texture perception is a parallel process based on some local conspicuous features or textons. These local features can be in turn best computed by means of local operations [6]. All these considerations have given rise to the development of a large number of existing texture analysis approaches (especially the LLCM group) which are rooted on local operations. A unification and generalization of these local-operation-based approaches was presented in Chapter 3 with the introduction of a conceptually new scheme for texture feature extraction. The scheme, being termed as local vector mapping (LVM), is described by three degrees of freedom namely local neighborhood specification, local vector formation and vector mapping. At the stage of local neighborhood specification, a local image region of some shape and size is defined. With local vector formation, the pixels in the chosen 2-D local neighborhood are sequentially scanned in some way and consequently formulated as a 1-D vector which is called a local vector. Then the obtained 1-D vector undergoes a linear or nonlinear mapping or transformation. The principles of these steps, i.e., local neighborhood specification, local vector formation and vector mapping, have been detailed, and some of the existing approaches to texture feature extraction such as the LLCM approaches were interpreted in terms of LVM operations. After an entire LVM operation, the input scalar image is mapped into another scalar image or several scalar images. For the purpose of texture classification, the first four central moments of these LVMed images have been defined as texture features for the original image. To be more illustrative of the general principles of the LVM scheme, concrete examples of LVM operations such as local discrete cosine mapping, local polynomial mapping, local variance mapping, etc., have been discussed. Texture features calculated from these LVMs have been applied in texture classification. Very high classification accuracies have been achieved. In particular, the second moments, i.e., the variances, of the LLVMed images are found to be extremely powerful in discriminating textures. Recalling that all testing samples have been preprocessed to possess identical first-order statistics, the high classification accuracy indicates the capability of LVM operations to map the differences in higher-order statistics into the differences in lower-order statistics such as

to map second-order statistic differences into first-order statistic differences.

The LVM scheme elaborated in Chapter 3 requires the mapping of the pixels in a 2-D local neighborhood into a 1-D vector. An obvious question to ask is can useful local operations be defined directly on a chosen 2-D local neighborhood? The answer is yes and was presented in Chapter 4. By visualizing a 2-D local neighborhood as a local discrete surface, a local surface fitting (LSF) procedure for texture feature extraction has been described in Chapter 4. A LSF operation involves local neighborhood specification, basis function selection and discrete surface fitting. The output of an entire LSF operation is a set of fitting coefficient images. Then for the purpose of texture feature extraction, the first four central moments of the resultant coefficient images were defined to be texture measures. The relationship between the LVM scheme and the LSF scheme was also established in this chapter.

A fundamental difficulty in implementing the LVM scheme of Chapter 3 or the LSF scheme of Chapter 4 is the specification of a local neighborhood of appropriate shape and size. As texture is a local regional image property, the chosen local neighborhood has to be large enough in order to extract intrinsic and reliable texture measures. For example, if a texture is composed of very large sub-patterns or primitives, then the chosen local neighborhood should intuitively be large. Nevertheless, large local neighborhoods necessitate an increase in computational complexity. A way out of this dilemma is seemingly to adopt a structural approach by first extracting texture primitives and then calculating texture features based on primitive analysis. This has been the concern of Chapter 5. In this chapter, three different techniques have been proposed for primitive extraction, the key step in most of structural approaches. The first technique, being an extension of an existing technique, involves Gaussian low-pass filtering followed by fixed percentile thresholding; the second consists of intensity segmentation and fixed percentile thresholding; while the third technique is composed of intensity segmentation and moment invariant thresholding. For the purpose of texture classification, texture features were defined to be the means and standard deviations of the attributes of the extracted primitives. The relative effectiveness of the three primitive extraction techniques was evaluated and compared by means of texture classification. Experimental results obtained in this chapter have confirmed the common assertion that structural approaches are particularly effective for structural textures but much less so for stochastic ones.

The criticisms of the approaches towards texture feature extraction presented in Chapters 3 and 4 are due to the difficulty in the specification of pertinent local neighborhoods, which is texture-dependent; and the criticisms of the approach described in Chapter 5 are linked to the ineffectiveness of the defined texture features in discriminating stochastic textures. All these criticisms indicate that the effectiveness and efficiency of the texture feature extraction approaches developed in Chapters 3, 4 and 5 are dependent on the texture samples under study. However, in realistic applications, texture samples concerned can result from a wide spectrum of sources, and furthermore *a priori* knowledge on these samples may be simply not available. This implies that in reality algorithms for texture feature extraction are desired which are generally applicable and whose effectiveness is independent of input texture samples. Such a need leads to the development of a novel algorithm for texture feature extraction and classification presented in Chapter 6.

The algorithm draws directly upon the mechanisms of the HVS in texture analysis as the HVS is no doubt the most effective and efficient system in performing texture feature extraction and classification. Thus the models for and the recent findings on the HVS in the context of pictorial information processing have been first reviewed. The two most instructive findings/ hypotheses are a) the HVS or specifically the visual cortex processes input pictorial signals via a set of parallel, quasi-independent channels each of which is tuned to a specific band of spatial frequency and orientation; and b) the profiles of the receptive fields of simple cortical cells can be well described by symmetrical or anti-symmetrical Gabor functions/filters. Based on these findings, a cortical channel model has been proposed which attempts to model the process of texture feature extraction in the hypothesized individual channels in the HVS. In order to determine the number and position of the channels to be implemented for effective feature extraction, two experiments were designed and conducted to investigate the frequency component resolution characteristics of the HVS. These experiments have led to the specification of the final filtering pattern which employs 16 channels tuned to 4 spatial frequencies and 4 orientations. For the purpose of feature extraction, the means and standard deviations of the 16 channel output images are defined as texture measures for a given input sample. These features were then used in texture classification experiments involving 14 distinct texture classes. Very high classification accuracies have been obtained by using a small number of features.

To further confirm the effectiveness of the procedure described in Chapter 6 for texture feature extraction, the procedure was compared with the conventional benchmark, the GLDM approach. The comparison was made in terms of classification performance, computational complexity, noise sensitivity and sample-invariance. The comparison results indicate that the proposed procedure is superior to the GLDM technique in all aspects.

In Chapters 3, 4, 5 and 6, concern lies in texture feature extraction and classification. Investigations on the most challenging issue in texture analysis, i.e., texture segmentation, started in Chapter 7, followed by Chapters 8 and 9. Unlike texture classification, texture segmentation has only received a limited amount of attention, and the progress in this area is far from satisfactory. Most of the existing approaches to texture segmentation have been reviewed in Chapter 7. Few surveys of this kind have been published in the open literature. In Chapter 7, the existing approaches were divided into two categories, i.e., UNITRE approaches and TEXTED approaches. The UNITRE approaches surveyed include feature space clustering (FSC), statistical classification (STC), the state estimation approach (SEA), generalization of intensity segmentation methods (GIS), and the spatial filtering approach (SFA); the TEXTED approaches reviewed include texture gradient operator (TGO), the spatial filtering technique (SFT), the theoretical estimation approach (TEA), and the texton-based method (TBM). The principles, merits and weaknesses of these approaches have been outlined.

The success of the LVM scheme in texture feature extraction and classification reported in Chapter 3 has inspired the development of a number of useful algorithms for texture segmentation elaborated in Chapter 8. All these algorithms rely on the capability of LVM operations to map textural differences into intensity differences, or equivalently to map higher-order statistic differences into first-order statistic differences, which has been confirmed in Chapter 3. The first two algorithms presented in Chapter 8 are devoted to uniform texture region extraction. The first algorithm is directly based on a single LVM operation, and is composed of local vector mapping, intensity thresholding, optimal low-pass filtering and gray level segmentation. The main difficulty in applying this algorithm is the selection of an appropriate threshold at the intensity thresholding step. Meanwhile, the inclusion of intensity thresholding makes the algorithm become necessarily sensitive to noise. In order to exclude the intensity thresholding step and thus improve this basic algorithm, the operation of local vector mapping has to be capable of mapping textural differences into very large intensity differences

as intensity thresholding basically intends to enhance the mapping capability of the preceding step, i.e., the step of local vector mapping. By realizing that a single LVM operation is unlikely to be capable of achieving this, successive LVM operations have been proposed to reach this goal leading to the development of the SLVM algorithm for UNITRE. It has been found that a SLVM composed of a LLVM followed by a local variance mapping suffices in segmenting a large number of textures.

In reality, textures usually differ from each other in many attributes, and differences in a subset of these attributes may only cause a partial discrimination. As a single LVM or SLVM operation is in general incapable of performing a complete mapping from textural differences to intensity differences, or in other words, is unlikely to be sensitive to all textural attributes, the application of multiple LVM or SLVM operations is essential to the accurate segmentation of a given input image. This has prompted the development of the third algorithm presented in Chapter 8, i.e., the MUCTED algorithm for texture edge detection. It is a multichannel algorithm. Each channel is assumed to be tuned to some specific textural attributes, and employs a distinct LVM or SLVM operation, followed by optimal low-pass filtering and gradient calculation. The channel outputs, i.e., the channel gradients, were integrated by means of channel grouping, the result of which is a gradient image indicating major texture transition regions. To get the final texture edge map, this total gradient image then undergoes intensity thresholding and edge thinning.

All the algorithms described in Chapter 8 have been tested under a variety of texture samples. The good implementation results obtained have illustrated the validity of these algorithms.

The work presented in Chapter 6 has been extended to texture segmentation in Chapter 9. The extension is by no means a trivial task as in texture segmentation the input image contains regions of distinct textures. Based on the proposed cortical channel model, a multichannel algorithm for texture edge detection has been developed. As in the case of texture feature extraction of Chapter 6, the decisive step in the proposed algorithm has been the determination of the number and position of the channels to be implemented. In Chapter 9, the number of channels to be implemented was chosen to be the number of perceivable spectral peaks as each spectral peak indicates one or more prominent texture feature in the spatial image domain. Each

channel was tuned to a specific spectral peak. Overall, the proposed algorithm consists of spectral peak detection, cortical channel model implementation, channel textural gradient calculation, channel grouping, intensity thresholding and pattern thinning. The most important steps have been the detection of spectral peaks and the implementation of the chosen channels.

The appropriateness of the cortical channel model proposed in Chapter 6 in the context of texture analysis and texture edge detection in particular has been theoretically justified in Chapter 9. It has been shown that if a channel is tuned to texture $T_k(i,j)$, then the channel output image exhibits very strong responses in regions with $T_k(i,j)$ whereas very weak responses elsewhere, thus achieving the mapping from textural differences into intensity differences. The effectiveness of the algorithm proposed in Chapter 9 has been confirmed by the satisfactory implementation results under a variety of textured images.

10.2 Suggestions for Further Research

In this thesis, a number of novel and successful algorithms have been developed for texture feature extraction and classification, and texture segmentation. However, this clearly is not the last word on the challenging issues, and we have merely provided some promising alternatives. Furthermore, the algorithms developed in this thesis can be improved in terms of less human intervention and better applicability. Above all, we believe that many fruitful areas of research can stem from the investigations presented in this thesis. To mention but a few examples, the following areas are believed to be particularly worthy of further research.

The investigation of the LVM scheme of Chapter 3 is far from complete. First of all, the specification of a pertinent local neighborhood is texture-dependent, and has to be carried out by means of human intervention. To make the step of local neighborhood specification be adaptive and automatic, research is required to develop techniques for automatically estimating the basic local neighborhood within which reliable and intrinsic texture measures can be extracted. This is basically the commonly termed *textural resolution* problem [1], which has only received very limited attention [2]. Secondly, in most of the texture classification experiments presented in Chapter 3, texture features were defined with the chosen local neighborhood being a 3x3 square region, and the local vector formation procedure being row concatenation. It is predicted that better performances can be achieved by adopting larger neighborhoods and hence local vectors of higher dimension and more sophisticated local vector formation procedures

(e.g., Peano scanning). However, larger neighborhoods and more complex vector formation schemes undoubtedly require more computation. Further research is needed to examine whether the improvement on performance is worthy of the increase in computational complexity.

Similarly, more experimental work can be carried out on texture feature extraction and classification using the LSF scheme described in Chapter 4. We have only applied second-order polynomial functions to fit a 3x3 local neighborhood. Higher-order polynomials or other basis functions could have been employed in conjunction with larger local neighborhoods, and comparisons between the applications of different basis functions and local neighborhoods could have been made in terms of classification performances.

The procedure for texture feature extraction developed in Chapter 6 is essentially rooted on the proposed cortical channel model, thus any refinement on the model could improve the validity of the overall procedure. The cortical channel model in its present form has only attempted to mimic the visual cortex in pictorial information processing. Nevertheless, before reaching the visual cortex, the scene or gray image received by the eyes has experienced some pre-processing operations such as low-pass and high-pass filtering at the stage of the retina and LGBs (lateral geneculate bodies). These effects have been completely ignored in the current cortical channel model. It would be very interesting to integrate models for the retina and LGBs into the current cortical channel model, and to study whether any improvement on the performance of the overall procedure for texture feature extraction could be consequently gained.

The filtering pattern adopted in Chapter 6 consists of 16 channels tuned to 4 frequencies and 4 orientations, and was designed under the assumption that all underlying image samples are of size 128x128. If image samples under study are of some other size, then the filtering pattern may not be suitable. In order to enhance the applicability of the proposed procedure for feature extraction, further investigation is worthwhile on the general design of the filtering pattern based on a possibly refined cortical channel model.

The texture classification results using the features proposed in Chapter 6 and the comparison between these features and the conventional GLDM features have confirmed the effectiveness, the general applicability and the superiority of the texture feature extraction procedure elaborated in Chapter 6. Meanwhile, the overall structure of the procedure is highly

parallel and is apt to real-time implementation. Research can be pursued to seek suitable hardware architectures for the procedure. Another area of research initiated in Chapter 6 is the investigation of the noise characteristics of algorithms for texture feature extraction. Although there are some comparison studies on texture feature extraction algorithms [3,4], they are based on the tacit assumption that all texture samples involved are noise-free. In reality, however, the texture samples concerned are rarely noise-free, and the knowledge about the noise sensitivity of various algorithms for texture feature extraction is valuable to the choice of a particular algorithm. The noise characteristics of the GF and GLDM features have been investigated and compared only empirically in terms of classification accuracy. More research can be carried out to investigate the noise characteristics of the GF, GLDM and some other conventional features both theoretically and empirically.

The texture segmentation algorithms developed in Chapter 8 are based on LVM operations and have been shown to be generally applicable. The primary problem is again the choice of pertinent local neighborhoods. This problem is more troublesome in texture segmentation than in texture feature extraction and classification. On one hand, the local neighborhood has to be large enough to extract reliable features; but on the other hand, it should be small enough to enable accurate identification of texture edges. Further research in this area can be concentrated on finding ways to overcome the incompatibility between feature reliability and edge identification accuracy. A possible solution may be to adopt a multi-resolution approach or a pyramid approach. For instance, the MUCTED algorithm can be applied at different levels of resolution and the outputs from all resolutions are integrated to obtain a final texture edge map. This kind of *multi-resolution multi-channel* algorithm is likely to be able to give better results than the proposed MUCTED algorithm applied at a single resolution level. Another way in which the MUCTED algorithm can be improved is to employ channel grouping models which take into account both the textural gradient magnitude information and the gradient phase information. In Chapter 8, the assumed linear model for channel grouping does not consider the textural gradient phase information thus ignoring the spatial coherence of texture edges.

For the texture edge detection algorithm described in Chapter 9, further research can be directed in the same directions as for the MUCTED algorithm of Chapter 8, i.e., the application of more realistic channel grouping models, and the adoption of a multi-resolution approach, in order to get better segmentations. The proposed algorithm in its present form is found not

suitable for segmenting stochastic textures as the power spectra of these textures do not exhibit perceivable peaks thus preventing the determination of the number of channels needed. However, the algorithm can be generalized to accommodate stochastic as well as structural textures. For example, one can partition the frequency plane into a certain number of bands, and find out those bands in which the energy of different textures differs prominently. Then, the number of channels can be set to the number of these bands, with each channel being tuned to one of the bands. Further research can be carried out to implement these ideas.

The algorithms developed in this thesis can prove useful in many other areas in image processing and analysis. For example, the texture segmentation algorithms presented in Chapters 8 and 9 can be incorporated into advanced techniques for data compression and image coding. While the compression ratio with the current intensity-based coding techniques is in the range of 10:1, compression ratio as high as 100:1 may be achieved by means of contour and texture coding [5]. These latter techniques are called the *second-generation* image coding techniques. A fundamental step in these techniques is texture segmentation to extract and model uniform texture regions. Because of the limited success in texture segmentation, progress in second-generation coding techniques has been insignificant. Research seems worthwhile on the application of the proposed texture segmentation algorithms in developing second-generation image coding techniques.

10.3 References

[1] **W.B.Thompson**, *Textural Resolution*, Proc. of 3th Int. Conf. on Pattern Recognition, Coronado, Calif., Nov.8-11, 1976, pp.283-285.

[2] **L.Van Gool** *et al.*, *Texture Analysis Anno 1983*, Comput. Vision, Graphics and Image Proc., vol.29, 1985, pp.336-357.

[3] **J.S.Weszka** *et al.*, *A Comparative Study of Texture Measures for Terrain Classification*, IEEE Trans. Syst. Man and Cybern., vol.SMC-6, 1976, pp.269-285.

[4] **R.W.Conners** *et al.*, *A Theoretical Comparison of Texture Algorithms*, IEEE Trans. Pattern Anal. Machine Intell., vol.PAMI-2, 1980, pp.204-222.

[5] **M.Kunt** *et al.*, *Second-Generation Image-Coding Techniques*, Proc. of IEEE, vol.73, 1985, pp.549-574.

[6] **B.Julesz** and **B.Krose**, *Features and Spatial Filters*, Nature, vol.333, 1988, pp.302-303.