

Performance Evaluation of Image Segmentation and Texture Extraction Methods in Scene Analysis

Submitted by

Mona Sharma

to the University of Exeter as a thesis for the degree of

Master of Philosophy in Computer Science

January 2001

Declaration

This thesis is available for Library use on the understanding that it is copyright material and that no quotation from this thesis may be published without proper acknowledgement.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of this degree by this or any other university.

Mona Sharma

Dedicated to my mother

Abstract

The main aim of this thesis is to evaluate the performance of image segmentation and texture analysis algorithms on synthetic and real images. As a part of this study, two popular texture benchmarks called MeasTex and VisTex have been used. A new scene analysis benchmark, called PANN database, has been generated as a part of this study for the evaluation of image analysis tools on natural object recognition tasks. The thesis demonstrates the considerable variability in an image understanding system performance based on different choices of image segmentation and texture analysis algorithms used. Hence, it is proposed that optimising each image analysis tool in a chain of processes is not enough for deriving optimal performances. The effect of a preceding process on its successor is both important and significant. So for example, how well we segment images, can have a dramatic impact on the quality of texture features we extract from such regions and subsequently this impacts our object recognition ability. This thesis includes results of exhaustive experimentation with four image segmentation algorithms, five texture analysis algorithms, and two classifiers, to demonstrate this variability. The thesis also discusses the similarity and differences in performances highlighting different patterns of mistakes made by different combinations. In addition to being a comparative study, it also shows results on object recognition in natural scene images. A complete system starting from image acquisition to generating ground truth data and classifying it is described for the analysis of natural scenes.

Acknowledgements

I am eternally indebted to my supervisor for all the help, guidance and expertise provided by him throughout this study. I really appreciate all the valuable time he has spent in helping me with programming and new ideas about this research. I have been very fortunate to have him as my supervisor and it would take more than a few words to express my sincere gratitude. I would like to thank my peers from the PANN lab for their help and support with programming and conceptual problems at all times. Many thanks to Keir for the excellent Linux support provided by him. Thanks to Jonathan for his care and friendship. Many thanks to Markos for his moral support, friendship, humour, and company. I would also like to thank Matthew Glennen for his help with ground truth labelling software that has been of great help in the completion of this project.

I would sincerely like to thank my parents for being there at all times and for their support and encouragement with my research. Last, but not the least, I would like to thank my sister for being my best friend and a wonderful companion.

List of Contents

	Pages
List of tables	9
List of figures	11
List of appendices	15
1. Introduction	16
1.1. Problem definition	16
1.2. Building a chain	17
1.3. Hypothesis	20
1.4. Thesis structure	21
2. Literature review	24
2.1 Image segmentation studies	24
2.1.1 Segmentation techniques	24
2.1.2 Segmentation evaluation	41
2.1.3 Comparative studies and reviews	43
2.2 Texture analysis studies	43
2.2.1 Texture techniques	44
2.2.2 Comparative studies and reviews	56
2.2.3 Texture similarity and salience	63
2.1. Classification	65
2.2. Scene analysis studies	70
3. Methodology	77
3.1 Data details	77
3.1.1 Texture benchmarks	77
3.1.2 PANN database	84
3.2 Plan of work	87
3.2.1 Texture benchmark analysis	87
3.2.2 PANN database analysis	88
4. Algorithms for image analysis and pattern recognition	99
4.1 Algorithms for image enhancement	99
4.1.1 Filters in spatial domain	99
4.1.2 Filters in frequency domain	100
4.2 Algorithms for image segmentation	101
4.2.1 Fuzzy c-means clustering based segmentation	101
4.2.2 Split and merge segmentation	103
4.2.3 Region growing segmentation	105
4.2.4 Histogram thresholding based segmentation	106
4.3 Autofill	107
4.4 SUSAN	108
4.5 Texture based feature extraction	108
4.5.1 Autocorrelation	109
4.5.2 Co-occurrence matrices	109

4.5.3 Edge frequency	112
4.5.4 Law's method	112
4.5.5 Primitive length (run length) encoding	112
4.6 Classifiers	113
4.6.1 Linear Discriminant Analysis (LDA)	114
4.6.2 Nearest neighbour classifier	114
4.6.3 Data dimensionality reduction	117
5. Texture benchmark analysis	119
5.1 Texture benchmarks	119
5.2 Benchmark analysis	119
5.3 MeasTex analysis	119
5.3.1 Linear classifier results	120
5.3.2 Nearest neighbour classifier results	123
5.4 VisTex analysis	125
5.4.1 Linear classifier results	125
5.4.2 Nearest neighbour classifier results	129
6. Experimental design for PANN benchmark	132
6.1 Scene analysis benchmarks	132
6.2 Problem definition	133
6.3 Experimental design	136
6.3.1 Experimental design based on two-phase classification	136
6.3.2 Experimental design based on multi-stage classification	137
7. Vegetation data analysis	139
7.1 Analysis of vegetation data	139
7.2 Linear classification	139
7.2.1 Fuzzy c-means clustering segmentation	140
7.2.2 Histogram thresholding segmentation	144
7.2.3 Region growing segmentation	147
7.2.4 Split and merge segmentation	151
7.2.5 Summarising linear classification results on vegetation data	154
7.3 Nearest neighbour classification	157
7.3.1 Fuzzy c-means clustering segmentation	157
7.3.2 Histogram thresholding segmentation	159
7.3.3 Region growing segmentation	161
7.3.4 Split and merge segmentation	164
7.3.5 Summarising nearest neighbour classification results on vegetation data	166
8. Natural object data analysis	169
8.1 Linear classification	169
8.1.1 Fuzzy c-means clustering segmentation	169
8.1.2 Histogram thresholding segmentation	173
8.1.3 Region growing segmentation	177
8.1.4 Split and merge segmentation	182
8.1.5 Summarising linear classification results on natural object data	185

8.2 Nearest neighbour classification	189
8.2.1 Fuzzy c-means clustering segmentation	190
8.2.2 Histogram thresholding segmentation	192
8.2.3 Region growing segmentation	194
8.2.4 Split and merge segmentation	197
8.2.5 Summarising nearest neighbour classification results on vegetation data	199
8.3 Conclusions	203
9. Conclusions	205
9.1 Scene analysis in perspective	205
9.2 Further improvement on recognition rates	207
9.3 Key results	213
9.4 Summary	214
References	215
Appendices	

List of Tables

Table	Title	Page
Table 2.1	Review of outdoor scene analysis techniques by Batlle et al.[13].	75
Table 3.1	MeasTex database composition and samples used.	78
Table 3.2	VisTex database composition and samples used.	83
Table 3.3	PANN benchmark data composition in terms of regions generated by different segmentation methods.	87
Table 3.4	Structure of the generated feature file	97
Table 3.5	Structure of the generated feature file with labelling.	98
Table 4.1	Haralick's 14 texture measures based on co-occurrence matrices.	110
Table 5.1	Details of feature extraction methods used on MeasTex data.	119
Table 5.2	Details of feature extraction methods used on VisTex data.	125
Table 7.1	Summary of linear classifier performance with FCM.	143
Table 7.2	Description of mistakes made by the linear classifier with FCM.	143
Table 7.3	Summary of linear classifier performance with Histogram Thresholding.	146
Table 7.4	Description of mistakes made by the linear classifier with Histogram Thresholding.	147
Table 7.5	Summary of linear classifier performance with Region Growing.	150
Table 7.6	Description of mistakes made by the linear classifier with Region Growing.	150
Table 7.7	Summary of linear classifier performance with Split and Merge.	154
Table 7.8	Description of mistakes made by the linear classifier with Split and Merge.	154
Table 7.9	Linear classifier mistakes for <i>autocorrelation</i> features.	155
Table 7.10	Linear classifier mistakes for <i>co-occurrence</i> features.	155
Table 7.11	Linear classifier mistakes for <i>edge frequency</i> features.	155
Table 7.12	Linear classifier mistakes for <i>Law's</i> features.	155
Table 7.13	Linear classifier mistakes for <i>primitive length</i> features.	155
Table 7.14	The different classifier performance depending on which data set is used for vegetation analysis.	156
Table 7.15	Summary of nearest neighbour classifier performance with FCM.	159
Table 7.16	Description of mistakes made by the nearest neighbour classifier with FCM.	159
Table 7.17	Summary of nearest neighbour classifier performance with Histogram Thresholding.	161
Table 7.18	Description of mistakes made by the nearest neighbour classifier with Histogram Thresholding.	161
Table 7.19	Summary of nearest neighbour classifier performance with Region Growing.	163
Table 7.20	Description of mistakes made by the nearest neighbour classifier with Region Growing.	163
Table 7.21	Summary of nearest neighbour classifier performance with Split and Merge.	165
Table 7.22	Description of mistakes made by the nearest neighbour classifier with Split and Merge.	165
Table 7.23	kNN classifier mistakes for <i>autocorrelation</i> features.	167
Table 7.24	kNN classifier mistakes for <i>co-occurrence</i> features.	167
Table 7.25	kNN classifier mistakes for <i>edge frequency</i> features.	167
Table 7.26	kNN classifier mistakes for <i>Law's</i> features.	167
Table 7.27	kNN classifier mistakes for <i>primitive length</i> features.	167
Table 7.28	The different nearest neighbour classifier performance depending on which data set is used for vegetation analysis.	168
Table 8.1	Summary of linear classifier performance with FCM.	172
Table 8.2	Description of mistakes made by the linear classifier with FCM.	173
Table 8.3	Summary of linear classifier performance for Histogram Thresholding.	176
Table 8.4	Description of mistakes made by the linear classifier with Histogram Thresholding.	177
Table 8.5	Summary of linear classifier performance with Region Growing.	180
Table 8.6	Description of mistakes made by the linear classifier.	181
Table 8.7	Summary of linear classifier performance with Split and Merge.	184
Table 8.8	Description of mistakes made by the linear classifier with Split and Merge.	185
Table 8.9	Mistakes made by the linear classifier for <i>autocorrelation</i> features.	186
Table 8.10	Mistakes made by the linear classifier for <i>co-occurrence</i> features.	186
Table 8.11	Mistakes made by the linear classifier for <i>edge frequency</i> features.	187
Table 8.12	Mistakes made by the linear classifier for <i>Law's</i> features.	187
Table 8.13	Mistakes made by the linear classifier for <i>primitive length</i> features.	188

Table 8.14	The different linear classifier performance depending on which data set is used for natural object data analysis.	189
Table 8.15	Summary of nearest neighbour classifier performance for FCM.	191
Table 8.16	Description of mistakes made by the nearest neighbour classifier with FCM.	191
Table 8.17	Summary of nearest neighbour classifier performance for Histogram Thresholding.	193
Table 8.18	Description of mistakes made by the linear classifier for Histogram segmentation.	194
Table 8.19	Summary of nearest neighbour classifier performance for Region Growing.	196
Table 8.20	Description of mistakes made by the nearest neighbour classifier for Region Growing.	196
Table 8.21	Summary of the nearest neighbour classifier performance for Split and Merge.	198
Table 8.22	Description of mistakes made by the nearest neighbour classifier for Split and Merge.	198
Table 8.23	Mistakes made by the nearest neighbour classifier for <i>autocorrelation</i> features.	199
Table 8.24	Mistakes made by the nearest neighbour classifier for <i>co-occurrence</i> features.	200
Table 8.25	Mistakes made by the nearest neighbour classifier for <i>edge frequency</i> features.	200
Table 8.26	Mistakes made by the nearest neighbour classifier for <i>Law's</i> features.	201
Table 8.27	Mistakes made by the nearest neighbour classifier for <i>primitive length</i> features.	201
Table 8.28	The different nearest neighbour classifier performance depending on which data set is used for natural object data analysis.	202
Table 8.29	Ranking different segmentation programs based on different feature sets f_1 (autocorrelation) to f_5 (primitive length).	203
Table 9.1	MeasTex PCA data classification using nearest neighbour classifier.	208
Table 9.2	VisTex PCA data classification using nearest neighbour classifier.	208
Table 9.3	PCA data classification using nearest neighbour classifier on vegetation data recognition with FCM.	209
Table 9.4	PCA data classification using nearest neighbour classifier on vegetation data recognition with Histogram Thresholding.	209
Table 9.5	PCA data classification using nearest neighbour classifier on vegetation data recognition with Region Growing.	210
Table 9.6	PCA data classification using nearest neighbour classifier on vegetation data recognition with Split and Merge.	210
Table 9.7	PCA data classification using nearest neighbour classifier on natural object data recognition with FCM.	211
Table 9.8	PCA data classification using nearest neighbour classifier on natural object data recognition with Histogram Thresholding.	211
Table 9.9	PCA data classification using nearest neighbour classifier on natural object data recognition with Region Growing.	211
Table 9.10	PCA data classification using nearest neighbour classifier on natural object data recognition with Split and Merge.	212
Table 9.11	The best results on different data sets.	213

List of Figures

Figure	Title	Page
Figure 1.1	Comparison of recognition performances of different combinations of image segmentation and texture analysis methods.	21
Figure 3.1	Sample images from MeasTex database.	79
Figure 3.2	Sample images used from the VisTex database.	81
Figure 3.3	Sample images not used from the VisTex database.	82
Figure 3.4	Sample images from the Brodatz album.	85
Figure 3.5	Samples images from PANN Scene Analysis Benchmark.	86
Figure 3.6	Flowchart for the methodology of benchmark analysis.	88
Figure 3.7	Flowchart showing various image processing steps for PANN benchmark analysis.	92
Figure 3.8	(a) Original segmented image; (b) median filtered output with 3x3 pixel mask; (c) median filtered output with 5x5 pixel mask; (d) median filtered output with 5x5 pixel mask.	93
Figure 3.9	The effect of brightness parameter on detecting edges in SUSAN; (a) Equalised map file; (b) output with brightness parameter $b=20$; (c) output with $b=10$; (d) output with $b=5$.	94
Figure 3.10	(a) SUSAN edge detected image; (b) the same edges overlaid on the colour original.	94
Figure 3.11	File selection for tagging images.	96
Figure 3.12	Region tagging for generating ground truth data.	96
Figure 4.1	(a) The pixel neighbourhood; (b) mask; (c) convolved values that are averaged.	100
Figure 4.2	(a) Original image; (b) FCM segmented image.	102
Figure 4.3	The partitioned image and its corresponding quadtree representation.	103
Figure 4.4	Split and merge methodology.	104
Figure 4.5	(a) Original image; (b) Split and merge segmented image.	104
Figure 4.6	A simple example showing region growing.	105
Figure 4.7	(a) Original image; (b) Region growing segmented image.	105
Figure 4.8	Selection of peaks to use for thresholding in histogram based segmentation.	106
Figure 4.9	(a) Original image; (b) Histogram thresholded image.	107
Figure 4.10	A simple example showing how Autofill works.	108
Figure 4.11	Five 1-D arrays identified by Laws.	112
Figure 4.12	Decision boundaries for LDA: (a) linearly separable data; (b) non-linearly separable data.	114
Figure 4.13	Nearest neighbour classifier in action on a two class problem.	115
Figure 4.14	k-nearest neighbour models and strategies: (a) Traditional k-nearest neighbour model; (b) Conflict resolution; (c) Closest average distance model; and (d) Hypersphere size effect.	117
Figure 5.1	(a) Plot of the first two principal components for the <i>autocorrelation</i> features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>autocorrelation</i> features on MeasTex data.	120
Figure 5.2	(a) Plot of the first two principal components for the <i>co-occurrence</i> features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>co-occurrence</i> features on MeasTex data.	121
Figure 5.3	(a) Plot of the first two principal components for the <i>edge frequency</i> features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>edge frequency</i> features on MeasTex data.	122
Figure 5.4	(a) Plot of the first two principal components for the <i>Law's</i> features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>Law's</i> features on MeasTex data.	122
Figure 5.5	PCA plot for the <i>primitive length</i> features on MeasTex data.	123
Figure 5.6	(a) Plot of the first two principal components for the <i>combined</i> features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>combined</i> features on MeasTex data.	123
Figure 5.7	(a) Plot of the first two principal components for the <i>autocorrelation</i> features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>autocorrelation</i> features on VisTex data.	126
Figure 5.8	(a) Plot of the first two principal components for the <i>co-occurrence</i> features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>co-occurrence</i> features on VisTex data.	127

Figure 5.9	(a) Plot of the first two principal components for the <i>edge frequency</i> features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>edge frequency</i> features on VisTex data.	127
Figure 5.10	(a) Plot of the first two principal components for the <i>Law's</i> features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>Law's</i> features on VisTex data.	128
Figure 5.11	Plot of the first two principal components for the <i>primitive length</i> features on VisTex data.	128
Figure 5.12	(a) Plot of the first two principal components for the <i>combined</i> features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on <i>combined</i> features on VisTex data	129
Figure 6.1	Data distribution overlaps during linear classification of natural scene analysis data containing 8 classes: vegetation (trees, grass, leaves) and natural objects (sky, clouds, bricks, pebbles, road). For all methods, FCM segmentation has been used.	134
Figure 6.2	Data distribution overlaps for the natural object recognition problem: a) vegetation and natural object data overlaps showing difficulty in linear discrimination; b) Natural object data after vegetation has been removed; c) vegetation after natural object data is removed.	135
Figure 6.3	Colour classification for separating vegetation data from other natural objects.	136
Figure 6.4	Multistage classification process where classifiers are used in a hierarchy.	137
Figure 7.1	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>autocorrelation</i> features for the discrimination of vegetation data.	140
Figure 7.2	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>co-occurrence</i> features for the discrimination of vegetation data.	141
Figure 7.3	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>edge frequency</i> features for the discrimination of vegetation data.	141
Figure 7.4	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>Law's</i> features for the discrimination of vegetation data.	142
Figure 7.5	PCA plot for the linear analysis using FCM and <i>primitive length</i> features for the discrimination of vegetation data.	142
Figure 7.6	PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and <i>autocorrelation</i> features for the discrimination of vegetation data.	144
Figure 7.7	PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and <i>co-occurrence</i> features for the discrimination of vegetation data.	144
Figure 7.8	PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and <i>edge frequency</i> features for the discrimination of vegetation data.	145
Figure 7.9	PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and <i>Law's features</i> for the discrimination of vegetation data.	145
Figure 7.10	PCA plot for the linear analysis using Histogram Thresholding and <i>primitive length</i> features for the discrimination of vegetation data.	146
Figure 7.11	PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and <i>autocorrelation</i> features for the discrimination of vegetation data.	147
Figure 7.12	PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and <i>co-occurrence</i> features for the discrimination of vegetation data.	148
Figure 7.13	PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and <i>edge frequency</i> features for the discrimination of vegetation data.	148
Figure 7.14	PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and <i>Law's</i> features for the discrimination of vegetation data.	149
Figure 7.15	PCA plot for the linear analysis using Region Growing and <i>primitive length</i> features for the discrimination of vegetation data.	149
Figure 7.16	PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge and <i>autocorrelation</i> features for the discrimination of vegetation data.	151
Figure 7.17	PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge and <i>co-occurrence</i> features for the discrimination of vegetation data.	151

Figure 7.18	PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge segmentation and <i>edge frequency</i> features for the discrimination of vegetation data.	152
Figure 7.19	PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge and <i>Law's</i> features for the discrimination of vegetation data.	153
Figure 7.20	PCA plot for the linear analysis using Split and Merge and <i>primitive length</i> features for the discrimination of vegetation data.	153
Figure 7.21	A graphical comparison of recognition accuracy obtained using different segmentation method and texture method combinations for vegetation data analysis using linear classifier.	157
Figure 7.22	A graphical comparison of recognition accuracy obtained using different segmentation method and texture method combinations for vegetation data analysis.	168
Figure 8.1	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>autocorrelation</i> features for the discrimination of natural object data.	169
Figure 8.2	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>co-occurrence</i> features for the discrimination of natural object data.	170
Figure 8.3	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>edge frequency</i> features for the discrimination of natural object data.	171
Figure 8.4	PCA plot and canonical discriminant function plot for the linear analysis using FCM and <i>Law's</i> features for the discrimination of natural object data.	171
Figure 8.5	PCA plot using FCM and <i>primitive length</i> features for discrimination of natural object data.	172
Figure 8.6	PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and <i>autocorrelation</i> features for the discrimination of natural object data.	174
Figure 8.7	PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding segmentation and <i>co-occurrence</i> features for the discrimination of natural object data.	174
Figure 8.8	PCA plot and canonical discriminant function plots for the linear analysis using Histogram Thresholding segmentation and edge frequency features for the discrimination of natural object data.	175
Figure 8.9	PCA plot and canonical discriminant function plots for the linear analysis using Histogram Thresholding and <i>Law's</i> features for the discrimination of natural object data.	175
Figure 8.10	PCA plot using Histogram Thresholding and <i>primitive length</i> features for the discrimination of natural object data.	176
Figure 8.11	PCA plot and canonical discriminant function plots for the linear analysis using Region Growing and <i>autocorrelation</i> features for the discrimination of natural object data.	178
Figure 8.12	PCA plot and canonical discriminant function plots for the linear analysis using Region Growing segmentation and <i>co-occurrence</i> features for the discrimination of natural object data.	178
Figure 8.13	PCA plot and canonical discriminant function plots for the linear analysis using Region Growing and edge frequency features for the discrimination of natural object data.	179
Figure 8.14	PCA plot and canonical discriminant function plots for the linear analysis using Region Growing segmentation and <i>Law's</i> features for the discrimination of natural object data.	180
Figure 8.15	PCA plot for the linear analysis using Region Growing segmentation and <i>primitive length</i> features for the discrimination of natural object data.	180
Figure 8.16	PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge and <i>autocorrelation</i> features for the discrimination of natural object data.	182
Figure 8.17	PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge and <i>co-occurrence</i> features for the discrimination of natural object data.	182
Figure 8.18	PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge segmentation and <i>edge frequency</i> features for the discrimination of natural object data.	183

Figure 8.19	PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge and <i>Law's</i> features for the discrimination of natural object data.	184
Figure 8.20	PCA plot the linear analysis using Split and Merge and <i>primitive length</i> features for the discrimination of natural object data.	184
Figure 8.21	A graphical comparison of linear classifier recognition accuracy obtained using different segmentation method and texture method combinations for natural data analysis.	189
Figure 8.22	A graphical comparison of recognition accuracy obtained using different segmentation method and texture method combinations for natural data analysis.	203

List of appendices

Appendix A	MeasTex data: Confusion matrices for LDA analysis
Appendix B	MeasTex data: Confusion matrices for kNN analysis
Appendix C	VisTex data: Confusion matrices for LDA analysis
Appendix D	VisTex data: Confusion matrices for kNN analysis
Appendix E	Vegetation data: Confusion matrices for LDA analysis for Fuzzy c-means clustering
Appendix F	Vegetation data: Confusion matrices for kNN analysis for Fuzzy c-means clustering
Appendix G	Vegetation data: Confusion matrices for LDA analysis for Histogram Thresholding
Appendix H	Vegetation data: Confusion matrices for kNN analysis for Histogram Thresholding
Appendix I	Vegetation data: Confusion matrices for LDA analysis for Region Growing
Appendix J	Vegetation data: Confusion matrices for kNN analysis for Region Growing
Appendix K	Vegetation data: Confusion matrices for LDA analysis for Split and Merge
Appendix L	Vegetation data: Confusion matrices for kNN analysis for Split and Merge
Appendix M	Natural object data: Confusion matrices for LDA analysis for Fuzzy c-means clustering
Appendix N	Natural object data: Confusion matrices for kNN analysis for Fuzzy c-means clustering
Appendix O	Natural object data: Confusion matrices for LDA analysis for Histogram Thresholding
Appendix P	Natural object data: Confusion matrices for kNN analysis for Histogram Thresholding
Appendix Q	Natural object data: Confusion matrices for LDA analysis for Region Growing
Appendix R	Natural object data: Confusion matrices for kNN analysis for Region Growing
Appendix S	Natural object data: Confusion matrices for LDA analysis for Split and Merge
Appendix T	Natural object data: Confusion matrices for kNN analysis for Split and Merge

Chapter 1

Introduction

The most beautiful thing we can experience is the mysterious. It is the source of all true art and science. He to whom this emotion is a stranger, who can no longer pause to wonder and stand rapt in awe, is as good as dead: his eyes are closed.



(Albert Einstein)

When someone suggested thousands of years ago that “a picture speaks a thousand words”, probably the idea of computing was limited to basic number crunching. In the modern age, the above adage still has significance to computing with images. In computer vision and image processing research, we aim to derive better tools that give us different perspectives on the same image allowing us to understand not only its content, but its meaning, and significance. Image processing can not compete with the human eye in terms of accuracy but it can outperform it easily on observational consistency, and ability to carry out detailed mathematical operations. Also on simple or structured tasks, computing solutions can be reliable, consistent and cheap. With time, image processing research has broadened from basic pixel based low-level operations to high-level analysis that now includes the use of artificially intelligent techniques for image interpretation and understanding. These new technologies are being developed to gain a better semantic understanding of images based on the relationship between its components, its context, its history if it is a part of a sequence, and *a priori* knowledge gained from a range of sources.

1.1 Problem definition

Image understanding is one of the key areas of research where the main objective is to understand the components of an image and interpret its semantic meaning. Image component recognition is the basic building block of most image processing based research that involves image understanding. What comprises an image must be first identified before we can analyse the image any further. In some applications, the recognition of image objects may be enough in itself. For example, in medical images, the identification of a tumour is enough and no further processing is needed. On the other hand, in some cases we need to identify image objects as a basis of a more detailed understanding of the image. For example, consider the problem of archiving video clips automatically on the basis of their content or context. In such cases the image object recognition only acts as a basis of a more detailed analysis of context or content relationship. Also, in some applications the knowledge of object category may be defined in term of their properties. For example, in the case of content based image retrieval, it is the content that defines the object category.

In the majority of image analysis applications, object recognition in images is a fundamental step. Such research can be easily categorised as those dealing with well-defined objects and those dealing with natural objects. Well-defined objects are mostly found in research dealing with industrial applications of machine vision and automated target recognition. Natural objects are found mostly in research dealing with remote sensing and scene analysis. In this thesis, we

are primarily interested in natural scene analysis, and in particular, the recognition of natural objects in images showing natural scenes. The recognition of natural objects in such images is not a trivial task and a number of other researchers have addressed this area. Natural objects have considerable variability that makes their identification difficult. In addition, because of this variability, some features such as shape, that are useful for analysing structured objects, are now of only limited use for scene analysis. Also natural objects are found in the real world that can not be structured as a laboratory environment. As a result of this, the effect of environmental conditions is a key factor in the quality of images that we can obtain. Finally, natural objects have a fractal nature, and a very large amount of visual information, which is difficult to capture in entirety using a digital camera.

Scene analysis research is one of the foundations of the development of autonomous systems that are able to think and act for themselves. The word “autonomous” brings to us thoughts of robots that can autonomously carry out the tasks that we do otherwise. Vision for them, is obviously a key sensor, and the analysis of natural scenes important for such systems to navigate and interact within an environment. However, these applications, though exciting as they are, tend to obscure equally important scene analysis applications that have nothing to do with mobile platforms such as robots. For example, scene analysis can be used for monitoring outdoor environments using a CCTV type camera. It can also be used for vehicle navigation, weather monitoring, detection of land mines, and a range of other civil applications. We are not restricted to the use of such technology to land only. Scene analysis has also been successfully used for underwater and space exploration. In addition, there is a range of military applications that benefit from scene analysis. Examples include surveillance, autonomous navigation, and weapons technology. In most of these applications, the recognition of natural objects is a goal in itself before some action can be triggered.

1.2 Building a chain

The analysis of natural objects is a complex task and involves a range of image processing steps before a final result is obtained. We can discuss these steps as follows. The first step in such analysis is the acquisition of natural images. For research purposes, these can be used from existing archives. These archives have been generated with different applications in mind. For real applications however, we might require the processing of a stream of video images rather than the analysis of single images from an archive. The same basic process of analysis applies to the video stream, however, the tools used can be quite different. In video analysis for applications that demand close to real-time requirements, tools that satisfy these requirements are to be preferred. In image acquisition, quite often we are not at liberty to choose the image resolution and camera settings as these are either already preset or only limited changes can be made. Image processing is a highly computationally intensive task and therefore there is a trade-off between image resolution and the speed of processing. In some cases we can take images of very high resolution but then eliminate areas that are not to be processed, thereby keeping a high resolution but still processing a smaller number of pixels. For example, once again consider the medical imaging problem. We can digitise an x-ray with a very high resolution, and devise methods to crop out information that is of no relevance in finding the tumour. In other applications, similar focus of attention technique can be used to process less number of pixels that would otherwise need processing. However, in scene analysis applications where we may

need to analyse the complete image, this is not mostly possible. Hence the image resolution must be carefully chosen and the speed of processing should be an important consideration throughout the development of a system.

Acquired images will not necessarily meet our criteria of good quality and will need some form of enhancement. Natural images are highly dependent on the quality of outdoor sunlight. The properties of an object surface such as texture is, in turn, dependent on such conditions. Generating a database of natural images, as we have done in this thesis, is by no means an easy task. At one end of our requirements, we need such a database to contain a diverse range of images that reflect real conditions as best as possible. At the same time on the end of the scale, these differences also need to be minimised, otherwise the whole database might become a collection of outliers. An attempt has been made in this study to keep an appropriate balance between these two requirements. As the second stage image enhancement is important for improving the quality of images such that image objects can be better identified. In most studies this would be an important preprocessing step. The images can be blurred or sharpened depending on how they look originally. However, enhancement is a very subjective process and needs to be optimised on a per image basis. In a study dealing with comparing texture analysis, such as ours, we are not sure how image enhancement affects texture measures and is avoided for most part on our data.

As the third step, images need to be segmented to identify their key components. An image consists of a number of natural objects defined by distinct regions. The image background in itself is a distinct region. These regions are defined by their boundaries that separate them from other regions. Image segmentation aims to identify these boundaries and identifies which pixel comes from which region. At this stage, for the image segmentation process it is not important to know which region belongs to which object. Each region is separated from others on the basis of some criteria of homogeneity within the region. The region definitions aim to maximise pixel differences across regions but at the same time minimise such differences within them. The process of image segmentation is a very difficult one. One of the primary reasons for this is the nature of objects imaged. These objects often do not have very well defined boundaries in images. As a result of poor lighting, the grey level differences between an object and its background are poorly contrasted making it difficult for a segmentation algorithm to generate an accurate boundary. At the same time, limited information on true boundaries makes it nearly impossible to evaluate these segmentation algorithms in isolation. In such difficult circumstances, it is to be expected that segmentation errors will be made. The result of such errors will be the imprecise definition of regions in images. Pixels will be wrongly assigned to a different region or object affecting its statistical distribution. There are two major problems with segmentation: under-segmentation and over-segmentation. If by default we presume that totally accurate segmentation for natural images is hard to achieve in practice, we need to minimise the under- or over-segmentation as much as possible. In the case of under-segmentation, full segmentation has not been achieved, i.e. there are two or more regions that appear as one. In the case of over-segmentation, a region that would be ideally present as one part is now split into two or more parts. These problems, though important, are not easy to resolve. As with enhancement, ideally segmentation should be optimised on a per image basis. However, with

limited measures for natural images on how well we are segmenting, this is hard to achieve and often the parameters are set for the complete batch process when analysing images.

As the fourth step, some ground truth data needs to be generated. In other words, we need to determine the identity of regions. Each region must come from one class or category. For example, in natural images, these classes can be trees, grass, clouds, sky, etc. One image can contain multiple regions from the same class. Also, not all classes are present in one image. Each region will be used in the next step to derive some characteristic features of that region. Each measurement from a region is called a sample or pattern. We need to establish the class identity of each pattern before performing any classification. Regions can be labelled using an interactive tool that allows the user to tag regions with class identity. The class identity can be an abbreviation such as T for trees, S for sky and so on, or simply sequential numbers such as 1, 2, ..., etc. The generation of ground truth data, though seemingly obvious, is not too trivial either. Fortunately, for the purposes of this study a purpose made tool exists that allows this task to be performed easily. At the end of the analysis, we are able to label each region in every image with a class tag.

As the fifth step, we need to define salient characteristics of the regions that are now labelled. For example, if we have one hundred regions of class 'trees', then we can derive a number of features from these regions that now define what a tree looks like. In future if a new tree region appears in a previously unseen image, by deriving the same measurements from this new region we could easily identify its class as tree based on feature matching. Characteristic features form the basis of most pattern recognition studies. They must be easily calculable and should be discriminatory across different classes. In addition, they should be easily interpreted in terms of their visual significance. A number of different studies have used different distinguishing features as discussed in the next chapter. The analysis of texture is of special importance in most studies as most natural objects have different surface properties and these can be used as primary means of distinguishing them. The use of different texture analysis algorithms yields different number and type of measures. These can be used on their own or combined together. The final metric of performance ability is the overall recognition accuracy defined as the proportion of samples that the system recognises correctly. All of the preceding steps except generating ground truth data are geared towards improving this ability.

As the final step, we need to test how well we can use the features so generated for developing a system that would classify natural objects. If the objective is to develop a system for commercial rather than research use, we would generate three separate sets of data. The first data set will be training data that will teach a classifier to recognise natural objects on the basis of samples given to it. A classifier can be thought of as the computer brain that remembers samples given to it. These samples are called the training data. The classifier has the task of recognising new unseen data that it has not seen before. This ability to identify new samples can be based on either generating from training data either mathematical boundary between data distributions as in the case of linear classifiers and neural networks, or on the basis of matching individual feature values as in nearest neighbour classifiers. How the classifiers are trained depends on application requirements. In some cases we may wish to bias the system towards better performance on certain categories or certain error measures. The performance of the

classifier data on data it already knows is not a good indicator of how well it will perform in the real world. Hence, a second data set called validation set can be used to test the true performance of the system under experimental conditions. Finally, if the system is deemed as satisfactory, it can be used for on-line applications where test data input feeds into the system. For laboratory based tests, often we have one complete set of data. Splitting it into a training and validation set does not lead to an exhaustive test of the system performance. Hence, in literature different cross-validation and bootstrapping techniques have been proposed. One of the most commonly used methods, also used here is of hold-one-out or leave-one-out. One sample can be taken for testing and all others used for training. This is done N times for N samples and the recognition results are averaged. This result is a much better reflection of the true performance of the system with the data available. Since it is never possible to have data for all possible conditions, one can only give a better representation of classifier performance in the real world by developing image database as similar to what the system is likely to encounter in the real world. The output of a classifier is an overall recognition rate and a confusion matrix that shows where the mistakes or misclassifications have been made. If different mistakes are weighted differently, the confusion matrix can be multiplied by a risk or cost matrix to give a final result showing how good or poor the system is.

1.3 Hypothesis

The above description details for us the various processes that form a chain. The input to this chain or pipeline are the images and the output is their content recognition. For each process, we have a variety of options. For example, for image enhancement, segmentation, texture analysis and classification we can choose one or more tools. The aim is to get the best output from this chain in terms of the best recognition performance. Now let us state the following hypothesis:

“The optimisation of this chain is not simply a matter of using the best known algorithms for each step. Their combined effect is one of the most important things to consider when developing natural object recognition schemes”.

As a part of this thesis, we take the above statement as a central theme. We aim to demonstrate through exhaustive experimental analysis that using different segmentation techniques on natural images followed by different texture extraction methods yield very different data. There is a considerable variability in our recognition performance depending on which data we use. The overall scheme is shown in Figure 1.1. Here we show that different recognition rates $R_1 \dots R_{20}$ will be obtained for different combinations of four segmentation methods followed by five texture analysis methods. It is common sense that different results will be obtained if we are to use different tools. As different image segmentation methods yield different region definitions, it is obvious that texture calculations from these will be invariably affected. However, the magnitude of variability in terms of different combinations of algorithms in succession has not been fully examined in scene analysis research. Due to the lack of such information, little optimisation is performed for making the chain output better. In this work we show that if we use the same feature extraction method but use different segmentation schemes to generate regions, significantly different results are obtained. The differences depend on the segmentation and texture algorithms chosen. Also there is a considerable difference between the worst and the best combination.

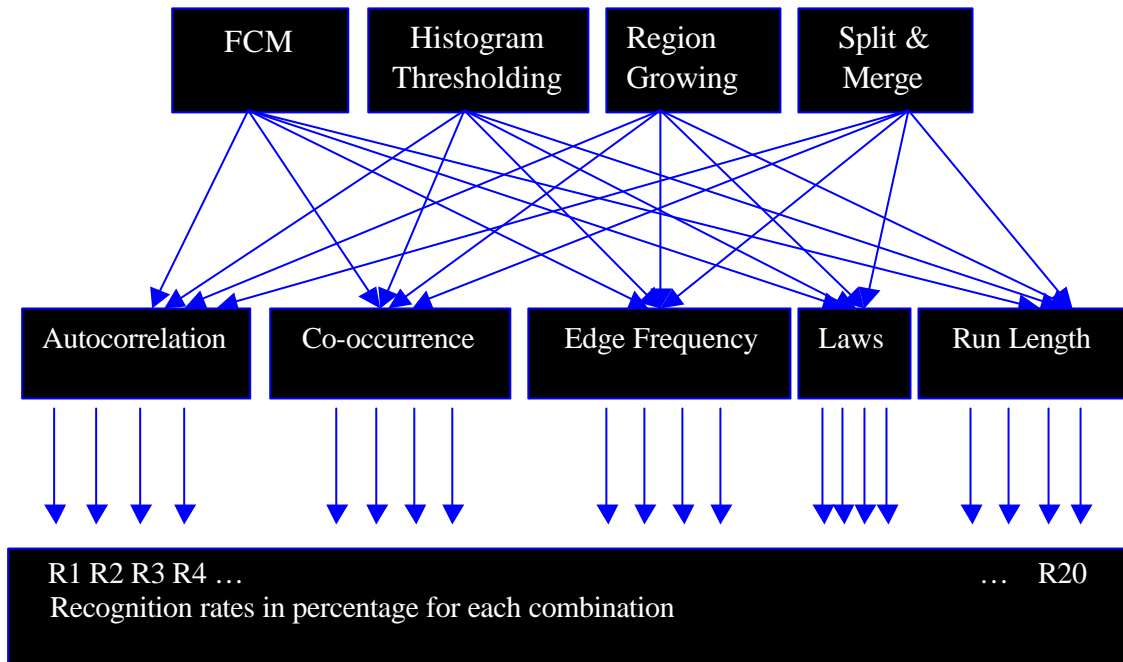


Figure 1.1 Comparison of recognition performances of different combinations of image segmentation and texture analysis methods.

In order to select good feature extraction methods, we have chosen some of the most popular methods used by several other researchers as shown in Figure 1.1. To verify their ability to recognise texture well, we demonstrate their ability on two synthetic benchmarks. The good results on these benchmarks show that these methods are capable of recognising different textures quite well and these can be used without hesitation for our scene analysis problem. Obviously it has not been possible to use all good methods of texture analysis or even segmentation. However, we feel that we are able to lend considerable support to our stated hypothesis based on our results. The main contribution of the work is this realisation of the importance of chain optimisation in image processing for scene analysis and the role played by different process combinations in this optimisation process. Only when we realise that we have a problem with traditional ways of approaching the development of object recognition schemes in natural scenes, that we can then begin to address it.

1.4 Thesis structure

The thesis is laid out as follows. In Chapter 2 we review the literature. We have reviewed research in the area of texture analysis, image segmentation and natural scene analysis. One of the key aims of literature review is to learn from others' experience and to make us better aware of what other research has taken place in similar areas. Also, the knowledge gained from this review allows us to select the most appropriate texture analysis and image segmentation algorithms for this study as well as texture benchmarks that have been used here. The review also puts our study in proper context and perspective. Only by investigating the results of other studies, we can measure our success and contribution.

In Chapter 3 we provide a detailed methodology of how we aim to build our image processing chain. In this thesis we have conducted experiments with synthetic benchmarks called MeasTex and VisTex. In this chapter we detail their properties and give example images. We also briefly

review other texture benchmarks that have not been used. One of the main outputs of this research has been the development of PANN Scene Analysis Benchmark that contains 448 natural images for scene analysis experiments. The characteristics of this benchmark data have been highlighted. For texture benchmark experiments, we do not need to segment images. However segmentation and ground truth data generation is necessary for the scene analysis benchmark. Flowcharts showing these are detailed in this chapter along with a detailed discussion of individual steps showing examples.

In Chapter 4 we describe the algorithms used for image enhancement, segmentation, texture analysis and classification. These algorithms are presented to show in more detail the processes that are applied to images before final results are obtained. These algorithms can be, with some additional reading, easily programmed. We also provide references from where some of the code has been used. The majority of the texture analysis and segmentation code has been written as a part of this project. Most other software code has been developed within our laboratory as a part of other projects.

In Chapter 5, we show the results obtained on the two texture benchmarks MeasTex and VisTex. The results have been presented on using the five texture extraction methods and for data from their combined set on texture recognition. For increasing the amount of data available, each image in MeasTex data has been subdivided into 16 parts and VisTex images into 4 parts. The performances of linear and nearest neighbour classifier are compared. The detailed results including confusion matrices for these results are shown in the appendices.

In Chapter 6 we detail the experimental layout for the next two chapters on analysing scene analysis benchmark. The benchmark contains eight different objects whose data distributions are highly overlapping. We devise a two step strategy for recognising such data. Colour information can be used to separate out vegetation data (trees, grass and leaves) from natural object data (sky, clouds, bricks, pebbles and road). Separate classifiers are trained and tested on these two different data sets. This scheme is computationally simpler and better suited for our analysis than a multistage classification strategy that is also explained in this chapter.

In Chapter 7 we detail the results of detailed experiments with vegetation data containing three classes namely trees, grass and leaves. A total of 20 data sets based on different combinations have been analysed. For each data set, linear classification results are compared with the nearest neighbour classification results. The results are compared on how well the two classifiers recognise individual classes as well as the overall classification ability. A detailed analysis of classifier mistakes and patterns of such mistakes across different data sets is also shown. The results are first discussed grouping five feature methods applied on regions generated from the same segmentation method. At the end of the chapter, results on the same texture method but now for four segmentation methods that generates different region descriptions are grouped together and the results are discussed. Finally, we summarise the results of the overall analysis and note the best segmentation method performance and texture method performance. These final results are used to give support to our earlier hypothesis by measuring the standard deviation across different performances and noting the vast difference between the worst and the best performances.

Chapter 8 is laid out exactly the same as chapter 7. Here we analyse the natural data using the two classifiers which has five classes namely sky, clouds, bricks, pebbles and road. The results have been produced using leave-one-out cross validation throughout the thesis.

In Chapter 9 we provide the conclusions for this thesis. As a part of this chapter we discuss the importance of our results to scene analysis research. We also suggest how our results can be further improved. Since the purpose of this study was to rigorously compare results on different feature sets, it has not been possible to improve data in any form before subjecting it to analysis. It is suggested that results can be further improved if we are to use outlier removal, better feature selection, colour and non-linear classifiers such as neural networks. We also demonstrate, showing results that principal components obtained from the various feature sets are easier to classify using the nearest neighbour classifier and yield some of the best results. The chapter concludes with a final summary.

Chapter 2

Literature Review

The field of scene analysis is a diverse and highly active research area. In this research work we address two important issues. We identify the relative advantages of using specific combinations of image segmentation and texture extraction algorithms for scene analysis; and we develop a robust methodology for scene analysis based on appropriate tools of image object recognition. For the first part, this work is related to a range of research studies published on image segmentation, texture analysis, performance comparison of image processing tools in these areas, and classification systems. For the second part, the literature available more broadly in the area of scene analysis is relevant where the performance of scene analysis systems can be compared. In this chapter we touch upon those studies that are of relevance to our work. These studies only form a small subset of literature that is available in the above mentioned areas. Every year, Azriel Rosenfeld from the University of Maryland publishes an extensive bibliography of published work for that year in the area of computer vision (see [184] for the recent survey). This chapter has been structured as follows. We review studies on image segmentation first, followed by research focussing on texture analysis, and classification in image analysis. Scene analysis study review forms the last part of this chapter.

2.1 Image segmentation studies

Image segmentation is one of the primary steps in image analysis for object identification. The main aim is to recognise homogeneous regions within an image as distinct and belonging to different objects. Segmentation stage does not worry about the identity of the objects. They can be labelled later. The segmentation process can be based on finding the maximum homogeneity in grey levels within the regions identified.

There are several issues related to image segmentation that require detailed review. One of the common problems encountered in image segmentation is choosing a suitable approach for isolating different objects from the background. The segmentation doesn't perform well if the grey levels of different objects are quite similar. Image enhancement techniques seek to improve the visual appearance of an image. They emphasize the salient features of the original image and simplify the task of image segmentation. The type of operator chosen has a direct impact on the quality of the resultant image. It is expected that an ideal operator will enhance the boundary differences between the objects and their background making the image segmentation task easier. Issues related to segmentation involve choosing good segmentation algorithms, measuring their performance, and understanding their impact on the scene analysis system.

2.2.1 Segmentation techniques

We review primarily those studies that are based on finding object regions in grey-level images. We also mention couple of studies that deal with colour segmentation to highlight how this has been used for outdoor scene analysis. Image segmentation has been approached from a wide variety of perspectives[165]. Our summary is presented for histogram thresholding, edge based

segmentation, tree/graph based approaches, region growing, clustering, probabilistic or Bayesian approaches, neural networks for segmentation, and other approaches.

Histogram Thresholding

Ohlander[151] proposed a thresholding technique that is very useful on segmenting outdoor colour images. This is based on constructing colour and hue histograms. The picture is thresholded at its most clearly separated peak. The process iterates for each segmented part of the image until no separate peaks are found in any of the histograms. The criterion to separate peaks was based on the ratio of peak maximum to peak minimum to be greater than or equal to two. Textured areas were separated from uniform regions by using a Sobel operator marking regions that contain large edge activity (more than 25 edge pixels in a 9x9 pixel window). These were called “busy” areas. The original area was not to segment inside these busy areas based on thresholding but sometimes it is necessary to do so as in the case of segmenting skylines.

Cheriet et al.[41] presented a general recursive approach for image segmentation by extending Otsu’s method[157]. This approach has been implemented in the area of document images, specifically for segmenting bank cheques. This approach segments the brightest homogeneous object from a given image at each recursion, leaving the darkest homogeneous object. This method is developed without any constraints on the number of objects in the digital image. The method is based on discriminant analysis. The thresholding operation is regarded as the partitioning of pixels of an image into two classes: object and background. For each iteration, the histogram of the image is drawn and the largest peak is separated from the rest of the image. The process is continued till there are no more peaks left in the histogram. Similar to Otsu’s method, the proposed method only performs well on the images with two classes. Using the method iteratively one can segment more than two objects. This technique also facilitates the process of information extraction from document images and preserves the topological properties of the extracted information that is used for further recognition. In this study, a set of human visual criterion was defined, where each criterion was given a weight. The system was trained on 220 bank cheques and 505 different cheques were used for testing the performance. Results of the performance analysis indicate that the percentage recognition rate varied between 93%-100% for different sets of test samples. The authors concluded that the method gives good results when the target object is the darkest object in a given image. However, when the target object is not darkest the method fails to segment properly.

In a number of applications, histogram thresholding is not possible simply because the histogram may be unimodal. In some cases the images may be of such quality that any preprocessing may not improve the contrast between objects sufficiently and hence one may not achieve two or more peaks in the histogram for selecting thresholds for segmentation. Unimodal distributions are typically obtained when the image consists of mostly of a large background area with small, but significant regions. This often happens in medical imaging applications. Similarly in aerial scenes with many different objects, the histogram may only have one peak because of the vast range of intensities for each object and an overlap between these. Bhanu and Faugeras[17] propose a gradient relaxation algorithm for solving the above problem and compare it to non-linear probabilistic relaxation algorithm proposed by Rosenfeld[183]. The process is based on assigning each pixel an initial probability of occurrence and then using

gradient relaxation based on compatibility function that taken into account the relationship between a pixel and its eight neighbours. The relaxation process is iterative where pixel values are changed so that the histogram is no longer unimodal and thresholds can be easily detected. Compared to Rosenfeld's method, the authors claim that this technique provides a better control by defining parameters that are under user control. The user can therefore control the degree of smoothing at each iteration as well as the initial assignment of probabilities. An extension of the gradient relaxation approach for image segmentation is presented in Bhanu and Parvin[18]. This new segmentation technique is based on recursive split and merge and has a number of advantages. The method avoids any heuristics and arbitrary measures for partitioning the image. Hence it does not require a robust merging step in order to remove boundary artefacts. In addition, the method does not require detailed peak location and selection procedure.

Li et al. [130] propose that the use of two dimensional histograms of an image is more useful for finding thresholds for segmentation rather than just using grey level information in one dimension. In 2D histograms, the information on point pixels as well as the local grey level average of their neighbourhood is used. The authors show that the application of Fisher linear discriminant to the histogram results in an optimal projection where the data clusters are better defined and hence easier to separate by choosing appropriate thresholds. The experimental results show that the proposed technique has the same computational demands as of one dimensional techniques but gives better segmentation results.

Edge based segmentation

Ahuja et al.[4] describe how pixel neighbourhood elements can be used for image segmentation. For each, its neighbours are first identified in a window of fixed size. A vector of these neighbours as individual grey values or vector of average grey levels in windows of size 1x1, 3x3 and 5x5 is determined. The paper uses both vector representations. The aim is to identify a weight matrix that multiplied with these vectors will yield a discriminant value that allows the classification of pixel in one of the two classes. Two different sets of experiments were performed on FLIR images. The first experiment is performed on 10 images. In the first experiment, two choices of feature vectors are chosen for every pixel. The images were of size 64x64 pixels containing a single tank each. The segmentation was done using the super-slice algorithm. The features were averaged over different sized windows (3x3 pixels and 5x5 pixels) and classification was performed using Fisher's linear decision rule. In the second experiment, a separate set of 25 images was chosen with light objects on dark background. Ten images were used for training and 12 points were chosen from the background and 12 from the interior of the object for each image. All of these samples were then used to train the classifiers. Testing was done on the remaining set of 15 images with some added noise. The authors stated that, for the noisy data, using the pixel grey level as the only feature, the results obtained were better than those obtained by pixel grey level properties. The authors proposed that the error rate for noisy images seems to depend primarily on number of features used. The authors concluded that the results suggest that the grey-levels of the pixel and its neighbours are a good set of features to use in pixel classification.

Prager[177] proposed a set of algorithms used to perform segmentation of natural scenes through boundary analysis. The goal of the algorithm is to locate the boundaries of an object

correctly in a scene. First, pre-processing of the images is done to clean up the raw data by smoothing and noise-removal. Second, the edge representation is generated. Differentiation is done to find the edge-strength at each point in the image. Suppression is then done to remove multiple edges formed by spatial differentiation of boundaries. Third, the edges are joined into line segments and features are computed. The features include length, contrast, frequency, mean, variance and location of each line segment. Fourth, post-processing is done to remove unwanted line segments and to build confidence for each of the remaining segments. The output of the system is a set of line segments with a list of attributes, such as length and confidence. The authors concluded that advantages of this approach are twofold. Firstly, the effects of each stage are well defined. Secondly, any stage can be omitted or replaced by a more sophisticated variant if desired.

Perkins[167] uses an edge based technique for image segmentation. It is acknowledged that edge based segmentation has not been very successful because of small gaps that allow merging of dissimilar regions. In order to avoid these problems, the paper proposes an expansion-contraction technique in which edge regions are expanded to close gaps and then contracted after the separate regions have been labelled. The size of expansion is controlled such that small regions are not engulfed by this process. The process involves the use of Sobel filter for producing edge strengths and directions at every point. The edges are thinned and the result is automatically thresholded leaving only ridges. The ridges separate regions of different intensity but there may be small gaps. Segmentation is performed by expanding active edge regions, labelling the segmented uniform intensity regions, and then contracting edge regions. The results are shown for landscape pictures and for pictures of electronic circuits.

Chan et al.[33] developed a new adaptive thresholding algorithm for image segmentation using variational theory. The method is a heuristic algorithm, which consists of seven steps. First, image smoothing is done using an average filter. Grey-level gradient magnitude is then derived from it. A thresholding and thinning algorithm is applied to the gradient magnitude to find the object boundary points. The image is then sampled at the boundary points as the local thresholds. The sampled local thresholds interpolate the threshold surface. Then the image is segmented by the threshold surface. Noise is then removed from the segmented image by a variational method. The method is performed iteratively to segment an image. The iteration process is stopped using an error or iteration time threshold. The authors presented the results on an image with 16 objects and simulated background and also on some handwriting images. The results were found to be more satisfactory when there are a large number of objects in the image as they yield a better thresholding surface. The method is demonstrated successfully on segmenting images for an OCR application.

Tree/graph based approaches

Cho and Meer[42] proposed a new approach for segmentation, which is derived from the consensus of a set of different segmentation outputs on one input image. Instead of statistics characterising the spatial structure of the local neighbourhood of a pixel, for every pair of adjacent pixels their collected statistics are used for determining local homogeneity. Several initial segmentations are derived from the same input image by changing the probabilistic component of the hierarchical Region Adjacency Graph (RAG) pyramid based technique. From

the ensemble of these initial segmentations, for every adjacent pixel pair a co-occurrence probability is derived, which captures global information (about the image) at the local level (pixel level). The final segmentation of the input image is obtained by processing the co-occurrence probability field with the same RAG pyramid technique. The pixel pairs with high co-occurrence probability are then grouped together based on the consensus about local homogeneity. This technique can also be used to extract the high confidence homogeneous regions from the co-occurrence probability field. Bayesian networks were then used to extract features from images. The features extracted were variance of the width of the region, ratio of average width to length and the average grey level. Then post-processing of over-segmented images is done based upon a priori information about the sought features. The RAG of the final segmentation provides the spatial relationship between regions and can be used for further interactive analysis of the image. This segmentation method is completely unsupervised. Experiments were performed on an aerial image, and images of boat, pentagon, and house.

Yeung et al.[229] proposed the technique of segmentation of video by clustering and graph analysis. The method is also extended to the Scene Transition Graph (STG) representation for the analysis of temporal structures extracted from a video. First, a video is taken and then similar shots are recognised to reduce the amount of information to be processed. Then dissimilarity between two shots is defined based upon the dissimilarity indices for all image pairs in the two shots. Then automatic segmentation of scenes and story units is done. The images are segmented based on the presentation of the featured video sequence along the timeline and the visual contents. The video shots are then clustered together based upon visual similarities of image contents and the temporal dynamics shown in the visual contents within individual shots (termed as time-constrained clustering). Then an STG is drawn and analysed. An STG is then used to represent compactly the structures of shots and the temporal flow of the story of videos. Some new attributes are defined from STG representation for the further analysis. It is assumed that two consecutive scenes in a video presentation do not share significant similarities in visual qualities. Cluster labels are given to shots in different scenes. The experiment was performed on a sequence of shots to extract story units. The video was successfully decomposed into a hierarchy of story units, each of which consisted of clusters of similar shots. The methodology also provided the mean of non-linear access to a featured program which in turn facilitated browsing of video contents.

Region growing

A range of image segmentation algorithms are based on region growing. We review some relevant studies that have used region-growing algorithms. Region growing algorithms take one or more pixels, called seeds, and grow the regions around them based upon a certain homogeneity criteria. If the adjoining pixels are similar to the seed, they are merged with them within a single region. The process continues until all the pixels in the image are assigned to one or more regions.

Chang and Li[34] proposed a region-growing framework for image segmentation. This process is guided by regional feature analysis and no parameter tuning or *a priori* knowledge about the image is required. The algorithm is known as Fast Adaptive Segmentation (FAS) algorithm. The image is first divided into many small primitive regions that are assumed to be homogeneous.

These primitive regions are then merged to form larger regions until no more merges are possible. Two regions are merged if they pass the homogeneity test and also if the value of the edge connecting them is weak. The focus of this study is on investigating how different merge criteria affect the quality of segmentation and the processing time. The experiments designed to evaluate the merge criteria are based on four important aspects of segmentation output: region mergeability, boundary accuracy, merge rejections, and number of iterations required. In these experiments, 300 images of size 50x50 pixels were used. Each image had two equal sized regions that share a simple straight 50-unit boundary. The data was randomly generated for the two regions from a pair of normal distributions having different means and variances. The best results using the fast merge method gives the correct classification rate of 86% (with less than 5 regions in the image). The authors concluded that the algorithm automatically computes segmentation thresholds based on local feature analysis. The algorithm is robust and produces high quality segmentation on a wide range of textured and grey scale images. This framework can also be easily adapted to different image applications by substituting the suitable features. The main limitation of this algorithm is however the limited applicability of the adaptive homogeneity test on very small regions and order dependency of its segmentation results.

For region growing, seeds can be automatically or manually selected. Their automated selection can be based on finding pixels that are of interest, e.g. the brightest pixel in an infra-red image can serve as a seed pixel. They can also be determined from the peaks found in an image histogram. On the other hand, seeds can also be selected manually for every object present in the image. Adams and Bischof[1] studied the effectiveness of seeded region growing approach for image segmentation of greyscale images where the seeds are manually selected. The method is employed to segment an image into different regions using a set of seeds. Each seeded region is a connected component comprising of one or more points and is represented by a set S . The set of immediate neighbours bordering the pixel is calculated. The neighbours are then examined and if they intersect any region from set S , then a measure δ (difference between a pixel and the intersected region) is computed. If the neighbours intersect more than one region, then the set is taken as that region for which difference measure δ is maximum. The new state of regions for the set then constitutes input to the next iteration. This process continues until all of the image pixels have been assimilated into regions. Hence, for each iteration the pixel that is most similar to a region that it borders is appended to that region. The SRG algorithm is inherently dependent on the order of processing image pixels. One implication of this algorithm is that raster order processing and anti-raster order processing do not lead to the same tessellation. The algorithm was applied on images with different types of objects in them. The authors concluded that the method has the advantage that it is fairly robust, quick, and parameter free except for its dependency on the order of pixel processing.

Mehnert and Jackway[141] improved the above seeded region-growing algorithm by making it independent of the pixel order of processing and making it more parallel. Their study presents a novel technique for Improved Seeded Region Growing (ISRG). ISRG algorithm retains the advantages of Seeded Region Growing (SRG) such as fast execution, robust segmentation and no parameters to tune. The algorithm is also pixel order independent. If more than one pixel in the neighbourhood have same minimum similarity measure value, then all of them are processed in parallel. No pixel can be labelled and no region can be updated until all other

pixels with the same priority have been examined. If a pixel cannot be labelled, because it is equally likely belong to two or more adjacent regions, then it is labelled as 'tied' and takes no part in the region growing process. After all of the pixels in the image have been labelled, the pixels labelled 'tied' are independently re-examined to see whether or not the ties can be resolved. To resolve the ties an additional assignment criterion is imposed, such as assigning a tied pixel to the largest neighbouring region or to the neighbouring region with the largest mean. This is a post-processing step. The algorithm in this study was tested on the image of man made objects such as a car, an aeroplane, and buildings. The authors concluded that ISRG algorithm produces consistent segmentation because it is not dependent on the order of pixel processing. Parallel processing ensures that the pixels with the same priority are processed in the same manner simultaneously.

Basu[12] developed general sets of semantics for region detection to describe a number of image models using them. The semantic set is established empirically based on simple and intuitive properties of a region. It can be extended to include new semantics without altering the conceptual and computational framework of the method. In case of region detection, each pixel of a digitised image is chosen as a primitive. An ideal region is obtained from a group of pixels of the given image by approximating the grey-level values of these pixels with a linear or quadratic approximation scheme. A set of attributes is calculated for each pixel of the group belonging to the given image as well as for the each pixel of the ideal region. In particular, four attributes: contrast, total variation, global average grey-level value, and representative grey-level value are used. Distance between the group of pixels and the ideal region is obtained as a numerical measure of dissimilarity between them. Pixel assignment to regions is decided according to the distance measure. The authors concluded that there is a noticeable improvement in these results when semantic information is added. There is no optimal set of attributes that can be used for all of the images. Any image has to be tested with a number of different attribute value combinations in order to obtain the best possible segmentation results. This scheme is found suitable for the class of problems in which fine structural detail of an image is not needed. The author demonstrated better results as a consequence of using semantic information than other known segmentation methods on natural objects.

Beaulieu and Goldberg[14] proposed a hierarchical stepwise optimisation algorithm for region merging which is based on stepwise optimisation and produces a hierarchical decomposition of the image. The algorithm starts with an initial image partition into a number of regions. At each iteration, two segments are merged provided they minimise a criterion of merging a segment to another. In this stepwise optimisation, the algorithm searches the whole image context before merging two segments and finds the optimal pair of segments. This means that the most similar segments are merged first. The algorithm gradually merges the segments and produces a sequence of partitions. The sequence of partitions reflects the hierarchical structure of the image. This is also termed as agglomerative clustering. The authors concluded that the results obtained from the final grouping of the algorithm are far better than the conventional non-hierarchical methods such as k-means clustering. The advantage of this algorithm over non-hierarchical methods is that there is no need to specify the seed points.

In some studies, edge or gradient information has been used in combination with region growing for image segmentation. Gambotto[72] proposed an algorithm that combines the region growing and edge detection methods for segmenting the images. The method is iterative and uses both of these approaches in parallel. The algorithm starts with an initial set of seeds located inside the true boundary of the region. The pixels that are adjacent to the region are iteratively merged with it if they satisfy a similarity criterion. A second criterion uses the average gradient over the region boundary to stop the growth. The last stage refines the segmentation. The analysis is based on cooperation between the region growing algorithm and the contour detection algorithm. Since, the growing process is performed by adding segments to a region, some pixels which belong to the next region and to the previous region, may be misclassified. A nearest neighbour rule is then used to locally reclassify these pixels. The algorithm was tested on a number of x-ray images. The authors concluded that the method uses both the statistical model of the region and the average gradient computed over its boundary. The system behaves similar to the snake class of algorithms but it can also be used to segment closed regions having unknown and complex shapes.

Hojjatoleslami and Kittler[100] proposed a new region growing approach for image segmentation which uses gradient information to specify the boundary of a region. The method has the capability of finding the boundary of a relatively bright/dark region in a textured background. The method relies on a measure of contrast of the region that represents the variation of the region grey-level as a function of its evolving boundary during segmentation. This helps to identify the best external boundary of the region. The application of a reverse test using a gradient measure then yields the highest gradient boundary for the region being grown. The unique feature of the approach is that in each step at most one candidate pixel will exhibit the required properties to join the region. The growing process is directional so that the pixels join the grown region according to a ranking list and the discontinuity measurements are tested pixel by pixel. The authors have performed a number of experiments on synthetic and real images. The authors concluded that the results are more reliable and consistent than conventional thresholding methods. The algorithm is also insensitive to a reasonable amount of noise. The main advantage of the algorithm is that no *a priori* knowledge is needed about the regions.

Lu and Xu[132] proposed a region growing technique for texture segmentation, in which a two-dimensional autoregressive model is used for texture representation. In this technique an artificial neural-network is adopted to implement the parameter estimation process for the autoregressive model and to compute the local texture properties of regions during the segmentation process. The segmentation procedure consists of two stages, namely the initial stage and the refining stage. At the initial stage, the image is partitioned into disjoint blocks or regions in the form of windows of a fixed size. The block is considered as a part of the internal area if it has the same texture class as its four neighbouring blocks in the horizontal and vertical direction. Otherwise, it is considered as undetermined. At the refining stage, all initial blocks are extended in parallel. Each undetermined block is divided into sub-blocks with equal size. Each sub-block is then compared with its neighbouring blocks by using the local information obtained by the neural network. The process terminates when each pixel in the image is

assigned to one of the initial regions. The algorithm was tested on 38 different textures from the Brodatz album. The algorithm provided accurate segmentation with less computational time.

He and Chen[96] propose a resonance algorithm for image segmentation that is not much different from seeded region growing algorithm. It is stressed that the resonance based segmentation algorithm is much more robust to illumination changes than conventional algorithms that work directly on grey scale images. The basic idea of resonance is based on the distribution of energy from a source to other elements in the system. This can be illustrated by considering each pixel in the image as a mass connected to a platform by a spring. The image contains several such mass-spring pairs that can be considered as immersed in water. When an external force acts on one such pair, there is a certain amount of displacement. If the external force matches the natural frequency of the mass, then the mass will resonate and the vibrations will spread to other mass-spring pairs that will oscillate as a result. The effect will only, however, last within a circle of certain radius and become weaker as we move away from the source. It is proposed that a seed based segmentation algorithm based on the above theory should be used only on feature images rather than original grey level images. The experiments are performed on Brodatz texture images of size 512x512 pixels. The grey level average is extracted within windows of size 8x8 and a feature image is generated. Since averaging is illumination dependent, it is expected that the true ability of the algorithm can be experimentally determined using such a feature. The results show better performance of the resonance algorithm compared to fuzzy c-means and histogram analysis segmentation methods.

Singh and AlMansoori[194] compared region growing and gradient based techniques for detecting regions of interest in digital mammograms. These regions of interest form the basis of applying shape and texture techniques for detecting cancerous masses. The study also proposes a two stage method where gradient based techniques are applied followed by region growing method which yields less number of regions for analysis. First, histogram equalization and fuzzy enhancement techniques improve the quality of image. Their utility is then compared using three quantitative measures. After the enhancement step, the images are then subjected to region growing or gradient operations (masking) for segmentation purpose. The segmented image is then analysed for estimating the regions of interest and the results are compared against previously known diagnosis of the radiologist. The authors concluded that the region growing segmentation gives lesser number of regions for the analysis as compared to gradient based techniques without compromising on quality.

Clustering

Image segmentation can be performed effectively by clustering image pixels. Cluster analysis allows the partitioning of data into meaningful subgroups and it can be applied for image segmentation or classification purposes. Clustering analysis either requires the user to provide the seeds for the regions to be segmented or uses non-parametric methods for finding the salient regions without the need for seed points. Clustering is commonly used in a range of applications such as image segmentation and unsupervised learning [108]. A number of issues related to clustering are worth studying including how many clusters are the best and how to determine the validity of clusters. In a number of segmentation techniques, such as fuzzy c-means clustering, the number of clusters present in the image has to be specified in advance. Several

techniques that do not require such initialisation have been proposed in literature (see [164] for a discussion on the limitations of traditional clustering techniques and description of non-parametric clustering that does not need prior initialisation).

Kurita[123] developed an efficient agglomerative clustering algorithm for region growing. This algorithm is a typical example of hierarchical clustering. The algorithm starts with an initial partition of a given image into N segments and sequentially reduces the number of segments by merging the best pairs of segments among all possible pairs in terms of a given criterion. The merging process is repeated until the required number of segments is obtained. The algorithm uses sorted linked lists to maintain neighbouring relations of segments and also a heap structure to store dissimilarities of all possible pairs of segments. A set of experiments was performed on a monochrome image and some range images of size 128x128 pixels. The number of regions turned out to be around 100 and the computational time was about 6 seconds. In case of range images each planar region is correctly classified into one segment but the curved regions were classified into several patches. The authors concluded that the algorithm makes the stepwise optimisation approach for region growing.

Frigui and Krishnapuram[64] have addressed three major issues associated with conventional partitional clustering. The issues addressed are sensitivity to initialisation, difficulty in determining the number of clusters and sensitivity to noise and outliers. Their proposed method Robust Competitive Agglomeration (RCA) starts with a large number of clusters to reduce the sensitivity to initialisation and determines the actual number of clusters by a process of competitive agglomeration. This method combines the advantages of hierarchical and partitional clustering techniques. To overcome the sensitivity to outliers, concepts from robust statistics are incorporated. Overlapping clusters are handled with the use of fuzzy membership. To handle doubtful regions, and to reduce the sensitivity to initialisation, the algorithm uses finite rejection. The agglomerative property makes it relatively insensitive to initialisation and local minima effects. The algorithm was tested on noisy data sets with synthetic ellipsoidal clusters and linear clusters, and a number of real range images. The results proved that RCA can provide robust estimates of the prototype parameters even when the clusters vary significantly in size and shape, and the data is noise contaminated.

Pauwels et al.[164] investigated non-parametric clustering for image segmentation. They propose a robust and versatile method that is able to handle unbalanced and highly irregular clusters using intermediate level processing. First, a density image is constructed by convolving the image data with a Gaussian density kernel. The nearest neighbours for each pixel are determined in a neighbourhood. The neighbourhood considered is one percent of the total number of pixels in the image. After convolution, candidate clusters are identified using gradient ascent and each point is linked to the point of highest density among its neighbours (including possibly itself). These process results in an overestimation of the number of clusters, carving up the data set into a collection of relatively small clumps centred around local maxima. A hierarchical family of derived clusters is constructed by using the data density to systematically merge neighbouring clumps. The order of merging is established by comparing the density values at neighbouring maxima with respect to density at the saddle-point (which is defined as the point of maximal density among the boundary points). The measures of cluster

validity for a given cluster is then quantified by two mathematically defined indices: isolation and connectivity. Using this clustering algorithm, it becomes possible to extract image regions that are salient and semantically meaningful. The authors concluded that the algorithms produced a smooth version of the inverse equalisation algorithm without destroying any information in the image. The method can be used on greyscale images such as natural scenes, faces, handwritings etc. The method is analysed on a number of synthetic and real images. The regions segmented were found homogeneous and the objects were easily identified from the image background. One of the advantages of the algorithm is that it automatically determines the number of objects from the image (or the number of optimal clusters) and therefore no *a priori* knowledge about the number of objects is needed.

Ohm and Ma[154] proposed a feature based cluster segmentation method for image sequences. The algorithm analyses specific features from the image sequence and checks their reliability and evidence locally for images in order to build segments that are probably part of one object. The segmentation procedure is basically a clustering procedure, which takes into account different features such as colour and motion. The approach is similar to vector quantisation. Different weights are applied to different features according to their reliability. The pixel-feature vector is then compared to a set of cluster-feature vectors and hence classified to a feature class to generate clusters. The set of cluster feature vectors is updated for each image in the sequence, which is also used for the segmentation of the next image. The labels associated with different clusters remain identical for different images of a same scene. After a scene change, the whole process is started with a completely new set of feature vectors, which are computed, from the first frame after the change and hence the whole sequence is segmented. The algorithm was performed on a sequence of flower garden images. In all of the frames the segmentation and tracking was done automatically. The authors concluded that the algorithm is a hybrid combination of a block-recursive and a pixel-recursive technique and produces a dense vector field. The algorithm also has the capability to set flexible weights for different feature components automatically.

Ng[149] describes an extension to the conventional k -means algorithms by modifying the splitting rule in order to control the number of cluster members. By adding suitable constraints into the mathematical program formulation, the author developed an approach that allows the use of k -means paradigm to efficiently cluster data sets with a fixed number of elements in each cluster. The main objective of this algorithm is the minimisation of an objective function usually taken as a function of deviations calculated for all patterns from their respective cluster centres. This algorithm minimises the objective function through a scheme that starts with an arbitrary initial cluster membership in an iterative manner to obtain better clustering results. Distributed pruning technique is used to reduce the number of clusters in the parallel manner. The results show that as the amount of pruning increases, as the data between partitions becomes more skewed i.e., the similarity between a sample and its membership in two or more clusters increases. The algorithm was used on a Print Circuit Board (PCB) insertion problem.

The comparison of various clustering techniques and studying their behaviour is important. It is important that an optimal clustering technique is able to choose the correct number of clusters. Dubes and Jain[56] detail comparisons on three classes of clustering techniques. A total of eight

programs are compared on minimised squared error based clustering, hierarchical clustering and graph theoretic clustering. Important issues related to the use of clustering programs such as user options, computational cost, input and outputs, and comparing programs that yield different outputs have been considered. They propose that different clustering algorithms can be compared on the basis of four factors: manner in which the next clustering is chosen or updated; criterion for creating new clusters; criterion for deleting and merging clusters; and the initialisation procedure. Approaches based on ranking clustering programs on the basis of their performance on a standard data set with performance criterion such as minimising squared error are criticised. One way of categorising such program is to cluster their performances. A set of nine criterion are defined for comparing clustering programs based on the manner in which clusters are formed, structure of data, and the sensitivity of the clustering technique to changes in data that do not essentially alter its nature. The different clustering programs are compared on the basis of these criteria using handwritten character data and a similarity matrix for these methods is shown.

The validity of clusters is important to study. Yarman-Vural and Ataman[227] critique several areas of clustering methodology including the definition of clusters, determination of the number of clusters, heuristic partitioning clustering algorithms and the effect of noise on determining accurate clusters. Cluster validity criteria including maximum likelihood information criteria and sum of squared errors is discussed. The first criteria is found to be better when the number of clusters changes. This paper also proposes a method of improving conventional clustering algorithms so that they display better performances with noise contaminated data. The ISODATA algorithm is modified to show its improved performance on up to 20% noise contamination in an artificial Gaussian mixture data set. Similar good performances are achieved on image analysis applications. Dubes[57] investigates the utility of Davies and Bouldin index for cluster validity and compares it with the proposed improved Hubert statistic. Results on a Monte Carlo study are reported by varying parameters such as dimensionality, number of patterns, sampling window, and number and spread of clusters. The new index is found to perform much superior as it shows more consistent performance when the above parameters are varied.

Zahid et al.[231] proposed a new heuristic method for fuzzy cluster-validity. Its principle is based on the evaluation of fuzzy separation and fuzzy compactness of clusters. The main attempt of the study is to analyse the natural structure present in the data set. This evaluation, which measures the quality of fuzzy clustering, assigns a real number to the outputs of the used fuzzy clustering algorithm. Two kinds of outputs are considered. For each output, a combination of fuzzy separation and fuzzy compactness criterion is evaluated. Maximising the resulting criteria results in good clustering. This maximum value allows in identifying the right number of clusters. The first function calculates a fuzzy separation and fuzzy compactness ratio by considering geometrical properties and membership degrees of the data. The second function evaluates this ratio by using only the properties of membership degree. The aim of classification phase is to generate a well-defined fuzzy c-partition that is as close as possible to the natural structure of the data. Four numerical examples were used by the authors to illustrate the use of the proposed algorithm as a validity function. The obtained results indicate that the proposed

method provides better answers than other cluster-validity functions especially with overlapping fuzzy clusters.

Probabilistic and Bayesian approaches

Haddon and Boyce[77] use co-occurrence based approach to image segmentation making use of region and boundary information in parallel for improved performance on a sequence of images. The authors examined image segmentation by unifying region and boundary information using co-occurrence matrices. The co-occurrence matrices were used to generate the feature space. The analysis was performed in the context of an ensemble of images. Based on the location of the intensities of each pixel and its neighbours in the co-occurrence matrix, initial segmentation is done. Each pixel is then associated with a tuple, which specifies whether it belongs to a given region or if it is a boundary pixel. This tentative segmentation was then refined by relaxation labelling that ensures local consistency of pixel labelling during segmentation by minimising the entropy of local neighbourhoods (see [84,103] for details on relaxation labelling for 2D and 3D images). If a pixel does not belong to the boundary, then it is assigned to one of the regions. This classification is entirely uni-dimensional in the co-occurrence direction and contains no explicit local consistency. The consistency for regions and boundary was obtained assuming that boundaries are not wider than one pixel. The algorithm was applied to synthetic images and infrared images of natural scenes. The authors find that segmentation worked reasonably well for most of the areas in the images while the majority of real edges and boundaries were correctly located. The algorithm was applied on a series of infrared greyscale images taken by a low flying aircraft. The authors conclude that the technique is robust and performs well even in poor imaging conditions. The algorithm is less effective if the clusters in the co-occurrence space have substantial overlap due to the imposition of local consistency. Since the techniques use global information in a local context, it was possible to adapt it to varying image characteristics i.e. variation in colour and texture. The difference between the co-occurrence matrices generated by a sequence of images is negligible if the image content and conditions do not change significantly.

A further extension of the above methodology is provided by Haddon and colleagues in their later study, Haddon and Boyce[78]. They suggest two intrinsically similar techniques for image segmentation and edge detection based on co-occurrence matrices. The first technique defines transforms that enhance the difference between typical and atypical image features. The second technique uses the location of region distributions in an edge co-occurrence matrix and defines the location of corresponding boundary distributions. The techniques are analysed by labelling the matrices. The labelled matrix can be used to segment the regions of an image and to simultaneously detect prominent regions. The image segmentation and edge detection is done at the same time. The operator used to generate the edge co-occurrence matrix effectively generates a lookup position within the labelled matrix. The operator is convolved with the neighbourhood of a pixel and this pixel is then replaced by the label of the co-occurrence matrix at the lookup position. This process is repeated using the same operator for a variety of operator orientations. Correlation techniques and conjugate iteration schemes are used to determine the location and the extent of the distributions in the co-occurrence matrix. The algorithm is applied to FLIR imagery and multispectral data. The majority of components in the algorithm are

performed in parallel. The results are significantly improved as compared to the previous algorithm where segmentation and edge detection were performed simultaneously.

Haddon and Boyce[79] used co-occurrence method to segment a sequence of FLIR images into its major regional components. These regions are subjected to texture classification. The co-occurrence matrices are temporally smoothed to ensure consistency of segmentation in the sequence. The edge co-occurrence matrices are decomposed using discrete Hermite functions because their underlying structure is Gaussian due to the Gaussian noise present in the images. If this structure is removed, what is left is the texture structure of the image. Neural networks (MLP) are then used to classify the images using the coefficients of the Hermite functions. Co-occurrence matrix is a multidimensional histogram, in which each element (i, k) , is the frequency with which two events i and j co-occur with specific relationship to each other. Edge co-occurrence matrices emphasize the intensity difference between two pixels. The authors of this paper used the canny edge operator to form the matrices. A sequence of 300 FLIR images from a low flying aircraft approaching a bridge are obtained and classified. These images are first segmented using co-occurrence matrices into their major regions and using temporally smoothed averaged matrices. Then a co-occurrence matrix is formed for every major region. The matrices are decomposed using orthogonal Hermite functions. The higher order functions describe the texture of the region and form the input vector of 121 features. Every third data sample was taken to form the validation set. Training and validation sets are disjoint. MLP networks were trained until validation error started to increase (over-train). Additionally, principal component analysis was used to extract the most descriptive of those 121 features. Very good results have been obtained with both a single layer network (architecture 25x10x5) with 96.9% training and 98.3% validation accuracy and a two layer network (architecture 25x5x8x5) with 94.8% training and 88.7% validation recognition accuracy. The major regions classified were grass, trees, sky, river reflecting trees and river reflecting sky. The major problems are the reflecting classes. PCA results were not as good as with the use of the full 121 features. The techniques described in this paper seem to be robust with noise. Overall, 93.4% of image area was correctly classified.

Haddon et al.[80] developed an algorithm for automatic segmentation and classification of images using a co-occurrence based approach using Hermite functions. The basic approach consists of four steps: segmentation, texture analysis, initial classification and labelling and spatio-temporal classification. In the segmentation phase, the key regions are separated from each other and from the remainder of the scene using co-occurrence matrices and edge detection simultaneously. The texture classification of segmented regions is performed using discrete Hermite functions. The co-occurrence matrices are decomposed for the calculation of features of segmented regions. The result of texture analysis is a low order feature vector that describes the texture of a segmented region. The features that contain information on discriminating between classes are selected and used as inputs to a multi-layer neural network classifier. Local consistency of interpretation is achieved in regions both spatially within an image and temporally within a sequence of images. A relaxation labelling approach is used to associate pixels belonging to particular coefficients. The compatibility coefficients take into account the current image and its predecessor in a sequence of images and results of relaxation labelling are applied to the previous image for classification. The authors evaluated the results on a sequence

of FLIR (Forward Looking InfraRed) images taken from a low flying aircraft and also performed snow profile analysis. The authors found that the algorithm is generally applicable to a wide range of imaging applications.

Haddon and Boyce[81] presented a method of ensuring consistency of interpretation in terms of the classification of segmented images, both spatially within an image and temporally through a sequence of FLIR images taken from a low-flying aircraft. It is important to maintain consistency with time in video analysis. Inconsistencies in segmentation between frames of a sequence are unacceptable in autonomous vehicle navigation applications. The image is first segmented using a co-occurrence based technique. After the image is segmented, the regions are classified based on texture. Co-occurrence matrices are also used for describing the texture of each region. The co-occurrence matrices are then decomposed using discrete Hermite functions. Features selected by using the linear discriminant analysis and principal components analysis were fed into an MLP neural network for classification. In enforcing spatio-temporal consistency, the output values of the neural network are taken into consideration. Two or more roughly equal outputs indicate that both outputs are considered equally likely. Consistency is enforced only when there is insufficient evidence of a region class. Consistency is enforced by using relaxation labelling, however, instead of using it in its classical way (looking at the image pixel by pixel), the authors treat a region as a single entity with both spatial and temporal neighbours. Adjacent regions, both spatially and temporally, are considered as neighbours. These neighbours influence the probability that a region belongs to a particular class if the result of the neural network is insufficient. These techniques were applied to a 12 second 300 frame sequence of infrared images. Five classes, trees, grass, sky, river reflecting trees and river reflecting sky are considered. The authors find very good results after the initial classification (first frame) regions. Regions that have equal membership in all classes were left as white. After consistency enforcement, all regions are classified, and only a few misclassification errors are found.

Comer and Delp[45] proposed a new algorithm for the segmentation of textured images using a multiresolution Bayesian approach. This algorithm uses a Multiresolution Gaussian Auto-Regressive (MGAR) model for the pyramid representation of the observed image. This algorithm effectively uses larger neighbourhoods to perform the segmentation than the single resolution algorithms. The observed data, which is a multiresolution representation of the observed image, is obtained using Gaussian pyramid decomposition. The nodes in a binary tree then index the data and an MGAR model is used. Value of a random variable is then predicted as a linear combination of the values of the random variables at previous nodes. A Gaussian pyramid representation of the image is generated. A multiscale Markov Random Field model is then used for the class label pyramid. The optimisation criterion is then used for the image segmentation. This criterion minimises the expected value of the number of misclassified nodes in the multiresolution lattice. Expected Maximisation (EM) algorithm is finally used for parameter estimation. This method also assigns a cost to an incorrect segmentation based on the number of incorrectly classified pixels in the segmentation. The algorithm was applied to two different images where three pyramid levels were considered, i.e. the original image was examined at three different resolutions. The algorithm performed well on both the images

segmenting them into regions of homogeneous textures. The algorithm was found successful in segmenting the image better than the EM and Multiresolution Posterior Marginal techniques.

Neural networks segmentation

Campbell et al.[31] proposed a automatic segmentation and classification method for outdoor images using neural networks. First, the images are segmented using Self-Organising Feature Maps (SOFM) based on texture and colour information of the objects. SOFMs used consisted of 64x64 nodes for best segmentation. A set of 28 features is then extracted from each region. The features include: average colour, position, size, rotation, texture (Gabor filters) and, shape (using principal components). Classification is then performed using a Multi Layer Perceptron with 28 input nodes and 11 output nodes. The training is performed on 7000 regions and testing is done on a independent set of 3000 samples. Over 80% regions were classified correctly using Learning Vector Quantisation and 91.9% regions were classified correctly using the Multi Layer Perceptron.

Papamarkos et al.[160] have developed the procedure of image segmentation using self organising maps. The use of these neural network paradigms is considered equivalent to multithresholding where the output of the network defines a number of homogeneous clusters. One of the interesting note from this paper is the technique used for finding the optimal number of thresholds or in other words the number of segmented regions. Considering that grey level distributions within the region are Gaussian, it is suggested that the linear combination of probability density functions for individual regions should match the overall density of the global histogram of the original image. For different number of segmented regions, we can find which result minimises the error.

Other approaches

Medioni and Yasumoto[140] proposed using fractal dimension method for image segmentation. The authors computed the fractal dimension of textures in frequency domain by approximating the log power spectrum by a straight line. The difference of the logarithm of the expected values of the two dipole-statistics yields a single texture feature. This procedure approximates the fractal dimension, which relates to the expected statistics of the grey-level difference versus the statistics of distance vector. The main drawback of this method is that segmentation based on single feature extracted from images is not always possible.

Pentland[166] addressed the problem of fractal based description of natural scenes. The authors addressed the following problems: (i) Representing natural shapes such as mountains, trees, and clouds, and (ii) Computing such description from image data. It has been observed that many natural objects such as leaves, snowflakes, etc. have a fractal nature. This makes it impossible to measure them accurately in an image. For example, measuring the length of a coastline, no matter what the size of the measuring tool is selected, all the curves smaller than the size of the measuring tool will be missed. Standard notions of length and area do not produce consistent measurement for many natural shapes: the basic metric properties of these shapes vary as a function of the fractal dimension. Fractal dimension, therefore, is a necessary part of any consistent description of such shapes. This means that fractals have to be used in the measurement of such shapes. The characterisation of image texture by means of a fractal surface

model makes it impossible to describe image in a manner that is stable over transformation of scale and linear transforms of intensity. The authors demonstrate how natural images can be segmented using this method. It is suggested that in this manner the segmentation is more stable in scaling the images down rather than using thresholding techniques on image intensity. Additionally, comparison with correlation techniques and co-occurrence techniques has been made using a mosaic of 8 natural textures constructed by Laws[127]. On these textures, the authors demonstrate that the fractal methodology yields better results than known segmentation algorithms.

Xu et al.[225] present a segmentation algorithm that is based on partitioning the image into arbitrarily shaped connected regions to minimise the sum of grey level variations over all partitioned regions under the constraints that each partitioned region has a specified number of pixels and that two adjacent regions have significant differences in average of grey levels. A minimum spanning tree has been used to construct these partitions and as such the segmentation problem reduces to tree partitioning problem. The effect of various noise contamination on the segmentation results is evaluated on the basis of how the noise affects the tree representation and tree partitioning.

In some studies, topological maps have been used as a structural method for image segmentation. Brequelaire and Brun[23] proposed a novel technique for image segmentation using topological maps and inter-pixel representation. The authors describe a data structure that allows the system to store and process regions of any size or form. It is called a model of discrete maps which helps in efficient computation of parameters required by the segmentation algorithm. There are two aspects in a segmented image representation: the geometrical aspect that describes the shape of regions, and the topological aspect that describes neighbourhoods and inclusions of regions. A topological map is defined as a partition of an oriented surface in a finite set of points. A discrete map is based on both geometric and topological levels of representation co-operating together. The discrete map data structure is then used to design segmentation algorithms. In the study the authors have described a recursive split and merge algorithm for segmentation using discrete data structure. In the first step, segmentation is done and in the second step refinement is done recursively to improve the quality of segmentation. The results are provided for a test image of *Lena* and images for various iterations are shown.

Ojala and Pietikäinen[156] presented an unsupervised texture segmentation method using feature distributions. The proposed algorithm uses distributions of local binary patterns and pattern contrasts for measuring the similarity of adjacent image regions during the segmentation process. Texture information is measured with a method based on local binary patterns and contrast (LBP/C). The segmentation method consists of three phases: hierarchical splitting, agglomerative merging and pixel-wise classification. First, hierarchical splitting is used to divide the image into regions of roughly uniform texture. Then, agglomerative merging is performed to merge similar adjacent regions until a stopping criterion is met. A pixel-wise classification is then performed to improve the localisation. The authors use a non-parametric likelihood test as a pseudo-metric for comparing feature distributions. The authors tested the method on four texture mosaics and two greyscale images of natural scenes. The method performed very well in these experiments. The authors concluded that the method is not

sensitive to the selection of parameter values and does not require any *a priori* knowledge about the number of textures or regions in the image. The method can also be easily generalised to utilise other texture features, multiscale information or colour features.

Yoshimura and Oe[230] proposed a segmentation algorithm for texture images using genetic algorithms that automatically determines the optimum number of segmentation areas. The authors divide the original image into many small rectangular regions and extract texture features from the data using two-dimensional autoregressive model, and other features such as fractal dimension, mean, and variance. The authors use three types of evolutionary segmentation methods. The first method uses Genetic algorithms (GA), the second method uses GA's and Self Organising Maps (SOM), the third method uses GA's and SOM considering the optimum number of segmentation areas in an image. Various experiments were performed on a set of simulated images with 3 or 4 different real textures. The authors found that the third method performed the best and the algorithms were visually accurate as compared to other conventional techniques. The first method produced good results but it is necessary to specify the number of regions beforehand. The second method selected the number of regions automatically, and the third method finds optimum number of regions with homogeneous texture. The authors concluded that the methods are effective for the segmentation of images that contain similar texture fields.

Perner[169] presents a methodology for a case based reasoning image segmentation system. The image segmentation is performed by looking up a case base for similar cases and using the segmentation parameters associated with the matched case. Hence, images are first matched for similarity on the basis of features such as image moments. Similarity is determined on the basis of measure proposed by Tversky[210] for non image information and a weighted similarity measure for image data. If ideal segmentation parameters are known for reference images, by matching new images to these reference images, the same segmentation parameters can be used. The system is shown to perform well on brain CT segmentation.

2.1.2 Segmentation evaluation

Kitchen and Rosenfeld[118] discuss the various issues related to under- and over-segmentation of images. The problems are blamed on the fact that context information is not used in segmentation algorithms. For example, a difference in brightness that is not significant in some contexts (such as in fluctuation in a textured background) may well be significant in other backgrounds (such as part of a low contrast border of an object with its surrounding). Under-segmentation is considered as a much more serious problem as it is easier to recover true segments through a merging process after over-segmentation rather than trying to split a heterogeneous region. In some situations, segmentation evaluation is based on how well the segmented image corresponds to the ground truth segmentation in terms of pixel by pixel differences. In most scene analysis applications, these differences need to be minimised and hence the segmentation criteria are rather harsh in terms of penalising algorithms severely if they do not segment accurately. However, in some applications of segmentation, such as in medical imaging, it may be enough that the segmented region overlaps with the true region (e.g. if the segmented region of interest overlaps with the true region, e.g. a tumour, then the segmentation can be classed as successful). Hence segmentation can be considered acceptable

even if it only partially detects the true borders or region. For measures on how to evaluate segmentation in such cases, refer to [113].

The evaluation of image segmentation programs is an important field of study. The evaluation can be categorised as supervised or unsupervised based on whether *a priori* information is available or not. A number of different approaches have been adopted. Some qualitative guidelines have been developed in Haralick and Shapiro[89]. Quantitative techniques of evaluation have been used by several authors. Weszka and Rosenfeld[224] used a busyness measure and classification error as performance criteria. Lavine and Nazif[126] defined a set of parameters for unsupervised segmentation including region uniformity, region contrast, line contrast and line connectivity. Sahoo et al.[186] used the uniformity criterion of Lavine and Nazif and a shape measure computed from the gradient values and the selected threshold value. In the case of supervised segmentation, errors are measured using a reference. The differences between the reference and segmentation output are measured to determine how well the segmentation algorithm performs. The simplest measure for supervised segmentation is probability of error as shown in Yang et al.[226]. This criteria can be used for finding optimal threshold values. This error can be decomposed into an under-merging or over-merging error[126]. This error measure may not however give full information on the quality of segmentation. Yasnoff et al.[228] have in addition proposed the pixel distance error for measuring segmentation performance in which the positions of misclassified pixels are taken into account. Two error measures, the percentage area misclassified and pixel distance error are used.

Another approach to measuring the quality of segmentation is based on the differences in the values of features computed from an ideally segmented image and an actual segmented image. These can be any object features, e.g. shape features such as eccentricity, or their texture, etc. This approach has been detailed by Zhang[232] and Zhang and Gerbrands[233]. Yang et al.[226] use shape features including area, circularity, orientation and elongation of segmented objects for the comparison of three segmentation methods. The under-merging, over-merging and ultimate measures of accuracy criteria are evaluated for comparing the algorithms considered. Wang et al.[218] compares several approaches for texture image segmentation. The techniques used are *k*-means, fuzzy *k*-means, fuzzy Adaptive Resonance Theory (ART2) and fuzzy Kohonen Self Organising Feature Mapping (SOFM). In their tests five features including energy, entropy, correlation, inertia, and homogeneity are used. After segmenting the images, features are extracted using Spatial Grey Level Dependence Method (SGDLM). The method is developed by Wang et al. [217]. This method is used because it is 3 times faster than SGLDM method developed by Haralick. The co-occurrence features are calculated in four directions of 0, 45, 90 and 135 degrees with respect to the horizontal axis. These features are selected because of: (i) strong within class variance; (ii) strong between class separation; and (iii) low sensitivity to small sample size. Experiments were performed on 2 images with different textures of size 150x360 pixels and 100x100 pixels. The experiments showed some limitations of *k*-means algorithm such as its slow processing ability and instability. Clustering errors were reduced by SOFM. Another advantage of SOFM algorithm is that the size of the neighbourhood was automatically adjusted due to its self-organising nature. The numerical results showed that

the fuzzy approaches are better than conventional k -means algorithms and SOFM provides the best approach for texture image segmentation.

Hoover et al. [101,102] proposed a methodology for evaluating range image segmentation techniques. The authors have also created a framework for the comparison of range image segmentation techniques. Experiments were performed on the data acquired through laser range finder or a triangulation technique. Images are segmented using any popular method that is to be evaluated. The authors have developed a tool for a human operator to create ground truth segmentation. The operator assigns three different labels to the regions. The labels are shadow pixels, noise pixels and cross edge pixels. Then machine segmentation is compared to the ground truth. Every region in the ground truth and machine segmented images is mapped as one of the five possibilities: correct detection, over-segmentation, under-segmentation, missed and noise. The first three categories apply to mappings of regions between the ground truth and machine segmented images. A missed classification applies to a region in the ground truth image and noise classification applies to a region in the machine segmented image.

2.1.3 Comparative studies and reviews

Haralick and Shapiro[89] state that: “As there is no theory of clustering, there is no theory of image segmentation”, (p. 100). They categorise segmentation algorithms as: measurement space guided spatial clustering, single linkage region growing schemes, spatial clustering clustering schemes, single linkage region growing schemes, hybrid linkage region growing schemes, centroid linkage region growing schemes, and split and merge schemes. The paper details the basics of these schemes with examples.

Similarly, Fu and Mui[65] state that: “Almost all image segmentation techniques proposed so far are *ad hoc* in nature. There are no general algorithms that will work for all images”, (p. 4). Their survey categorises image segmentation studies as follows: feature thresholding or clustering, edge detection techniques, and region extraction techniques. The authors recommend that one of the fruitful areas of further investigation lie in combining spatial and semantic information with edge detection and thresholding or clustering techniques to perform image segmentation. A number of review studies on image segmentation are available including [65,88,89,158,186,196].

2.2 Texture analysis studies

The analysis of image texture is extremely important. Its study primarily as surface property of objects requires a detailed understanding of object properties and imaging optics (see Jähne and Haußecker[107] on a treatment of some important topics in this context such as radiation, illumination, radiometry, solid state imaging, imaging optics and calibration of imaging systems). It also requires us to understand how humans discriminate between different textures and how best to model our algorithms to do a task as good. Undoubtedly, texture analysis has a wide range of applications. These include: (a) classifying images based on texture; (b) segmenting an input image into regions of homogeneous texture; (c) determining surface-shape on the basis of texture gradient; (c) synthesising natural looking textures for graphics applications; and (d) image retrieval from a database based on texture similarity. Before one analyses an image for texture, one must ask the question: Is there any texture in the image? Karu

et al.[116] have addressed this issue. Their work deals with the determining whether a given image has any texture in it for analysis. Most standard definitions of texture treat it as a measure of coarseness in an image. Images that have no texture can be classified as noise images, and images containing distinguishable objects that are too large to be termed texture. Hence, for an image to contain texture, it should be between these two extremes. Statistical texture analysis methods based on grey level properties are less useful in identifying texture than structural methods that are based on textural primitives and their placement rules. It can be reasonably assumed that each image has some texture at some scale. A simple measure of texture coarseness is based on first computing the local extrema of image function along rows and columns. The density of these extrema can then be used as a measure of coarseness of texture. The coarseness at a pixel location can be determined by doing this computation within a small neighbourhood.

Ahuja and Rosenfeld[5] provide an overview of the structural and statistical approaches to texture analysis. Statistical models have been classed as time-series models or random field models. Structural models are based on primitives and their placement rules. In the following section, we describe in brief a range of texture techniques that have been proposed by various researchers. Their comparison appears in the next section. Finally, we discuss studies that are related to finding texture similarity and salience.

2.2.1 Texture techniques

A number of studies have developed their own techniques for texture analysis. We categorise them here as geometric and topological approaches, second or higher statistics based approaches, texture with masks and logical operators, texture with stochastic models or random walk, texture based on gradient information, texture based on spectral filters, and other methods.

Geometric and topological approaches

Kundu and Chaudhuri[122] propose the use of fuzzy geometric features for texture classification. In their approach, first a set of 2D local membership value extrema is detected for the image. These are used as seed regions and grown till they do not touch other seed regions. The resulting regions are called regions of influence. A number of features are then calculated from these including fuzzy area, perimeter, compactness, height, and width, that form the basis of texture classification. On a total of 16 images taken from Brodatz album of size 128x128 pixels, a recognition rate of 90% correct classification was obtained.

Chen et al.[36] and Chen[37] propose a very interesting use of binary stacks for characterising texture. The SGF method is based on generating L different binary images from the original image that has a total of L grey scales, each time using slightly higher threshold for segmentation. A collection of these resulting images is called the binary stack corresponding to the original image. For each binary image, all 1-valued pixels are grouped into connected regions. The same is done for all 0-valued pixels. For each of the connected regions, a measure of irregularity or circularity, defined as slightly different, are computed. These measures are then weighted on the basis of the total size of connected components. For each binary image, the following four characteristics are computed: the number of 1-connected regions, the number of 0-connected regions, and the two weighted irregularity measures for these connected regions.

For each of these four characteristics, four features can be derived as follows: max value, average value, sample mean and sample standard deviation, yielding a total of 16 features. These features are used for texture classification. The studies compares the classification ability of these features against co-occurrence matrix approach, Fourier features and Statistical Feature Matrix (SFM) features on the complete set of Brodatz images (all 112 Brodatz images). The results are generated using the leave-one-out nearest neighbour classifier. On the complete data, the recognition performances were found as follows: SGF (85.6%), SFM (72.8%), Co-occurrence matrices (64.6%) and Fourier features (32.7%).

The idea of using binary stacks has been recently used by Garcia et al.[73]. The features are computed on binary images using 1D Boolean model. The splitting of images into binary stack is different here compared to the study by Chen et al.[36]. Rather than using 256 threshold planes for slicing, the authors consider various ways in which this information can be reduced. Bit slicing method is used in this study that generates different planes depending on the different bits. Hence a total of 8 planes are generated per image corresponding to each pixel bit. The boolean model is defined by the origin and distribution of line segments that can be defined as continuous pixel strings of the same value. The locations are defined as the left most end and are defined by marking probabilities. The distribution of line segments themselves is defined by a discrete distribution function. These probabilities are parameters of the boolean model. The procedure uses four Hilbert scanings to obtain the pixel strings. In order to compare texture, the distance between two boolean models with different marking distributions and discrete distribution functions is defined. The experimental results are generated on 30 texture database and different bit slicing methods are compared on the basis of the results obtained. Binary planes are considered for classification on their own and in pairs. Quantisation with equalisation scheme with a sum of similarities based classifier gives the best result of 97.2% correct on 360 test images of size 128x128 pixels.

Pietikänen et al.[175] investigate the efficiency of distribution based classification and their proposed feature set in rotation-invariant texture classification. This study also investigates the effect of using small windows of size 32x32 pixels compared to 64x64 windows on texture analysis. Texture measures based on center-symmetric autocorrelation, rotational invariant binary patterns and grey-level differences are used in the experiments. A number of features based on local pixel neighbourhood are derived that measure covariance (SCOV) and symmetric autocorrelation (SAC) defined as the ratio of symmetric covariance and local neighbourhood standard deviation. If the grey levels are replaced by their ranks in the original image, the ranked symmetric autocorrelation measure (SRAC) can be obtained. In addition, three local variance measures are calculated. Also, local binary patterns are obtained by thresholding the pixel neighbourhood on the basis of centre pixel value. The output is multiplied by a binomial function and the resultant matrix is summed into a single value that acts as a feature. This is called Local Binary Pattern (LBP). Finally, using the grey level differences a probability distribution is defined from which the absolute grey level differences in all four directions are accumulated as a feature. The authors use a k-nearest neighbour classifier on 15 Brodatz images. The performance of the features is evaluated as single features and in pairs. The results show that better performances are achieved with larger window size. Some interesting conclusions are drawn on how the various features perform on rotated images.

Second or higher order statistics

Haralick et al.[86] proposed the novel technique for texture image classification. This study is concerned with the task of developing a set of features for classifying or categorising pictorial data. Texture is chosen as the most suitable feature to represent images. Texture contains important information about the structural arrangement of units and their relationship to the surrounding environment. This discriminatory information can be used to classify images. The authors present a general procedure for extracting textural properties of blocks of image data. These features are calculated in the spatial domain and it is assumed that the texture information in an image I is contained in the overall or “average” spatial relationship that the grey-tones in the image have with one another. A set of grey-tone spatial-dependence probability-distribution matrices is computed and a set of 14 textural features, which can be extracted from each of these matrices, is suggested. The matrices are constructed by assuming that every pixel, except the peripheral ones, have eight nearest neighbours (horizontally, vertically and diagonally at 45 degrees). It is also assumed that the texture-context information is adequately specified by the matrix of relative frequencies. Some of these textural feature measures, obtained from the matrices relate to specific textural characteristics of the image such as homogeneity, contrast and the presence of organised structure within the image. Other measures characterise the complexity of the grey-tone transitions, which occur in the image. The usefulness of textural features for categorising images has been tested on three sets of images. In the first instance, photomicrographs of sandstones that are important in the petroleum industry were categorised. The data set was consisted of 243 images divided into five classes. The set was divided into training and test data. A set of 8 variables comprising the mean and variance of the textural features obtained from the matrices was used for classification. The classification result yielded 89% accuracy. In the second instance, a set of aerial photographs was classified using the min-max decision rule. The data set consisted of 170 images divided into 8 categories. Four grey-tone spatial dependencies and 11 textural features were used in the classification that resulted in 82.3% correct classification. In the third instance, a set of satellite images was classified. The 624 samples in the data set were divided into training and test sets of equal sizes. The mean and variance of the four textural features and eight spectral features was used for classification. The result was 83.5% correct classification as opposed to the 74-77% correct classification obtained by using spectral features. Since this seminal study, the features suggested by Haralick et al. have been used in most of texture studies. The theoretical and visual significance of these measures is given by Baraldi and Parmigianni[11].

Weszka and Rosenfeld[222] proposed an application of co-occurrence matrices based texture analysis to materials inspection. The study proposes texture methods that are used to assess material surface quality in order to distinguish between poor and acceptable quality samples. For these purposes, 12 digital photographs of the material surfaces are used. Within an image, a small window of pixels is used for calculating texture features. Texture measures used include Fourier power spectra and co-occurrence matrices. For co-occurrence matrices, four measures are calculated in four principal directions. These include angular second moment, entropy, contrast and correlation. In addition, for each feature, the mean, standard deviation, maximum, minimum, and range of these three statistics were also computed and used as features. The aim of the study was to find which set of features provides the best classification that tallies with the

human expert. The authors found that the best results were obtained for the three co-occurrence based features, namely entropy, standard deviation of entropy, and maximum of correlation. It was also concluded that the judged quality of surface texture did not correspond with the obvious visual property of the material.

Davis et al.[49] and Davis et al.[50] describe generalised co-occurrence matrices for texture discrimination. These do not describe texture directly but rather describe the spatial arrangement of local image features such as edges and lines. The description of Generalised Co-occurrence Matrices (GCM) is based on three attributes: image feature prototype, spatial predicate and prototype attribute. The prototype is regarded as the structural definition of the image features of interest. For example, the prototype edge-pixel can be defined as an ordered triple with three attributes (location, orientation and contrast). Spatial predicate is a mapping from image feature pairs into {true, false} category. The authors compare three prototypes, namely pixel intensity, edge pixel, and extended edge. For each of the three categories, their spatial predicates are defined. The following features are extracted from GCM: contrast, uniformity, entropy, and correlation. In their first study, classification experiments are performed on 30 texture samples. They find that compared to the co-occurrence matrix approach, their method performs much better (an average of 60 percent versus 43 percent for single features, and 68 percent versus 43 percent for pairs of features). In their latter study, three separate experiments are performed based on two classification schemes of nearest neighbour classifier, linear discriminant analysis, and the computation of inter-class distances using Bhattacharya method. 128 images of natural texture containing 64x64 pixels each are used for experiments. For the nearest neighbour analysis, feature pairings of size 2 are used and for the linear classifier all features are used. The best nearest neighbour result of 61% correct is obtained using the contrast and entropy pair of features on edge-pixel prototype and for the linear classifier the best result of 77% correct is obtained for pixel intensity prototype.

More recently, an interesting method of improving the quality of co-occurrence matrix features has been proposed by Walker et al.[215]. They classify the features proposed by Haralick and colleagues as being weighted on either the matrix element's value or its spatial location. For example, energy and entropy measures are weighted on the basis of value, and inverse difference moment, shade, inertia, promonance, correlation and variance are weighted on the basis of spatial location. The authors propose that it is best to suppress those elements of the matrix that yield little to the discrimination ability. Hence, on the basis of Bhattacharya distance calculation, one can find which elements are the most discriminatory. A discrimination matrix containing these weights can be multiplied with the original matrix to yield a better representation of values that are discriminatory. From these value one can either compute traditional measures or features as weighted sum of elements. Based on new calculations, the authors find that on six out of eight measures much better performances are observed on a cell nuclei classification task with some improvements as much as 70% greater accuracy.

Kovalev and Petrou[119] proposed a novel method for object recognition and matching based on quantitative estimation of relations between elementary image structures. This technique is based on automatic search of features that characterise a certain object class using a training set consisting of both positive and negative examples. Multidimensional co-occurrence matrices of

order 10 on each axis are used for the description and representation of these image structures. The features obtained are rotation and reflection invariant. Instead of comparing the elements of the matrices directly, the authors proposed the extraction of some features out of the matrix by considering all possible pairs of non-zero matrix elements and taking their ratio. Thus, only features that separate the positive and negative examples into two linearly separable classes are considered. All reliable features depending upon the reliability coefficient are then used for the classification. The co-occurrence matrix of a given unknown image is constructed and the corresponding features are computed. A positive vote is counted every time a feature of the test image has a value within the acceptable range and a negative vote is counted for every feature that takes a value outside the acceptable range. The total number of votes indicates the confidence with which the object was classified. The experiments were first performed on 18 CT scans of the brain out of which 8 were negative samples and 10 were positive samples. The results proved that all 8 normal images were correctly recognised while 9 out of 10 positive samples were correctly classified. Leave-one-out testing method was used for the validation of the quality of results. The authors concluded that not all-basic structures of a certain type are equally important for object recognition. The authors proposed the use of only those structures that distinguish one object from another is important.

Al-Janobi[6] presented a new texture analysis method called Cross Diagonal Texture Matrix (CDTM). This method incorporates the properties of grey level co-occurrence matrices and texture spectrum methods that are both explained. The proposed method is based on characterising the texture information of an image by separating the eight neighbouring pixels around a central pixel in a neighbourhood of 3x3 pixels. The eight elements in the texture unit are divided into two groups: the Diagonal Texture Unit (DTU) and Cross Texture Unit (CTU). The members of each unit have a value of either 0, 1 or 2 depending on whether the pixel in that position is less, equal or greater than the central pixel. These two units are combined into a CDTM by taking CTU as x-axis and DTU as y axis of the matrix. From these units the Haralick's features are extracted for texture information. This method has the advantage that the grey level of the image has no effect on the size of the matrix. In addition, the computational complexity is reduced because of the reduced size of the matrix. Nine images from the Brodatz album have been used for the evaluation of this method. Because of limited number of images, each image is divided into 25 sub-images. Classification is performed using Bayes classifier and the following features are compared: 13 co-occurrence features from the original images, 13 texture features from CDTM, and 8 texture spectrum features. The results from CDTM are found to be the best (7.3% error) compared to 18.4% error with co-occurrence matrices and 31.9% with texture spectrum.

Statistics of order greater than two have also been applied for texture analysis. Murino et al.[145] propose a novel texture classifications scheme that is based on using Higher Order Statistics (HOS) for defining discriminatory features. One of the advantages of using these parameters is that they are insensitive to additive Gaussian noise, especially third order statistical parameters. The authors define HOS functions and their properties for texture classification. Cumulants, defined as higher order combination of moments, are used as features. The authors analyse the performance of these features on leave-one-out classification of images superimposed by Gaussian and exponential noise with different signal to noise ratio.

They find that near perfect classification is possible for zero noise and robust behaviour is shown towards additive noise.

In 1981, Julesz[112] wrote a seminal paper on textons. His experiments with identical second-order statistics revealed that pre-attentive texture discrimination system can not globally process third- and higher-order statistics. The discrimination is a result of a few local conspicuous features called textons whose first order characteristics have perceptual significance. These are based on iso-second-order textures. The textons are invariant to positional and scaling transformations. These symmetries of textons can not be broken by pre-attentive and peripheral vision. The analysis of textons is now a well-developed area in texture analysis and several studies have used them for classifying and modelling texture (see Pratt[178] for a description of texture fields defined by Julesz).

Gagalowicz[70] investigates the models for texture fields for modelling human vision. The model is based on second-order statistics as proposed originally by Julesz in 1962. The authors design a new tool to synthesise stochastic texture fields and propose a new conjecture for the visual discrimination of texture fields. The notion of second order statistics, proposed by Julesz, is replaced by the notion of second order spatial averages and the concept of locality.

Texture with masks and logical operators

Unser[211] developed local linear transforms for texture measurement. He proposed simple and small convolution masks in combination with the computation of the local variance using a moving window in the image. Instead of using different filters, he proposed using four 2x2 Hadamard masks. The first of these masks has equal elements and measures the magnitude. The other three masks have two elements equal to 1 and other two elements equal to -1 which are used to approximate the derivatives in horizontal, vertical and diagonal directions.

Manian et al.[136] present a new algorithm for texture classification based on logical operators. Operators built from logical building blocks are convolved with texture images. The logical operators are based on order-2 elementary matrices whose building blocks are symbols 0, 1, -1 , and matrices of order 1x1. These low order matrices can be operated on by using the following operators: row-wise join, and column-wise join. The study selects a total of six operators on the basis of their best performance. The six operator masks are first convolved with the texture regions and the response is used to compute a standard deviation matrix using a sliding window. Features are next computed by zonal-filtering using zonal-masks that are applied to the standard deviation matrix. The following features are obtained: horizontal slit feature, vertical slit feature, ring feature, circular feature, and sector feature. These features are normalised and a feature selection scheme based on distances between feature means and measure of standard deviation is used to find the best features for classification. Bayes classifier, Euclidean classifier and nearest neighbour classifier have been used. A total of 39 textures from the Brodatz album are used in the classification study. On majority of the images, the Euclidean and nearest neighbour classifiers perform the best showing between 90 and 100% accuracy. On mosaic of six Brodatz textures, the average results on comparisons of this technique are as follows: Logical operators (93%), co-occurrence matrix method (70%), Fourier power spectrum method

(59%), tree-structured wavelet transform method (61%), Law's texture features (61%) and Gabor features (63%).

Texture with stochastic models or random walk

Faugeras and Pratt[61] describe decorrelation methods of texture feature extraction. An array of independent identically distributed random variables is passed through a linear or nonlinear spatial operator to produce a stochastic texture array. By controlling the generating probability density and the spatial operator, texture fields of specific statistical properties can be created. Texture features generated on texture fields using co-occurrence matrices are criticised because of large dimensionality of the feature space and accuracy limitation in characterising low contrast texture. It is also concluded that the probability density of the whitened field and spatial autocorrelation function are incomplete descriptors of the stochastic process generating the texture. The authors develop a methodology of extracting autocorrelation of the texture field plus the first four moments of the first-order amplitude histogram of its decorrelated field. Laplacian and Sobel operators are used in place of whitening operator to generate the decorrelated field. The Brodatz texture features are evaluated on images of size 64x64 pixels on the basis of Bhattacharya distance as a figure of merit.

Weschler and Kidode[220] and Weschler and Citron[221] detail the use of random walk methods for texture classification. A probabilistic planar random walk model is defined. It is characterised by a particle moving in unit steps in one of the four directions parallel to the x- and y-axis if one assumes four neighbour connectivity. The random walk is fully specified if we define the probabilities of leaving a given pixel for any of its four neighbours. If the random walk is performed over some window A, then the absorbing barrier of this can be defined as those pixels in A which have at least one neighbour outside of A. The pixels belonging to the absorbing barrier are called mesh points M. This barrier is composed of four disjoint subsets in directions above, right, below and left. Gradient difference magnitude between absorption distributions of these subsets forms the basis of texture features. In the first study, 14 texture images from the Brodatz album have been used for classification using leave-one-out procedure with nearest neighbour classifier. The results are as good as 86% correct. In the second study, the authors test these features on 32 Brodatz images and find a classification accuracy of 89.7% correct.

Ahuja and Rosenfeld[5] detail mosaic models for texture that use random pattern generation processes in a plane to provide image structure. Two classes of mosaic models are discussed with examples including cell structure and coverage models. The cell structure model is generated by tessellating a planar region into cells and independently assigning one of m colours to each cell according to a fixed set of probabilities. The tessellation process determines the amount of randomness and a set of different mosaics can be created that are categorised as checkerboard model, hexagonal model, triangular model, Poisson line model, occupancy model and Delaunay model. Coverage models are obtained by a random arrangement of a set of geometric figures (bombs) in a plane. For each mosaic, its geometrical properties and its texture features can be computed. Given a digital image, it can be mapped to see how well it fits in with a particular mosaic generated by a model. A number of measures for finding this match are

proposed. By fitting mosaic models to real textures, real textures can be discriminated and modelled at the same time.

Modestino et al.[144] illustrate the various problems associated with using different traditional methods of texture analysis. They illustrate images that will have different texture but similar power spectra, run length values or autocorrelation features. They present an alternative method of texture extraction in the form of 2-D random fields. This approach makes use of the co-occurrence matrix, but instead of extracting features from them similar to Haralick's method, they perform maximum likelihood hypothesis testing of the co-occurrence matrix. The proposed approach is tested on a few of the Brodatz images.

Texture based on gradient information

Hong et al.[99] propose an edge based procedure for extracting primitives from texture. The technique is used to group region boundaries by joining facing pairs of edge points. The algorithm locates edges by applying an edge detection operator followed by thresholding to eliminate weak edges, and nonmaximum suppression to eliminate redundant responses to a single boundary. This achieves a cleaned edge map of potential boundary points that enclose regions. The regions so extracted are primitives from which the following characteristics can be measured: area, perimeter, dispersedness, elongatedness, eccentricity, direction of major axis, and the average grey level. The method is evaluated on a pilot study in which a few texture samples taken from Brodatz album and terrain images are classified on the basis of the above features. All features perform well on classification except for eccentricity and orientation.

Chou[44] shows a simple techniques for classifying image pixels as belonging to one of the three categories: shaped feature point, smooth feature point, and textured feature point. The decomposition is based on a set of simple fuzzy rules operating on the edge strength information. This decomposition can be very useful if we wish to decompose the image into three separate images and operate on them individually.

Texture based on spectral filters

Jernigan and D'Astous[111] proposed an entropy measure of the normalised power spectrum within regions in the spatial frequency representation of an image as a texture feature. For highly structured spatial distributions, we will get a low value of this measure, and more random distributions will give a high value. By treating the image spectrum within a region as a probability distribution, an entropy measure is obtained that reflects the distribution of spectral components. For discrimination between two textures, an entropy vector is obtained from different parts of the texture. The distance between entropy vectors of two regions shows the dissimilarity between them. The paper also introduces a frequency normalisation scheme that achieves a degree of size-invariance for frequency domain entropy measure.

Bovik et al.[22] model texture as irradiance patterns that are distinguished on the basis of high concentration of localised spatial frequencies. Textured images are encoded into multiple narrow spatial and orientation channels. The output of each channel is a complex modulated subimage whose instantaneous amplitude and phase envelopes describe the spatial support of

the frequencies and/or orientations to which the channel is tuned. 2D Gabor filters discussed in detail as the chosen channel filters that can be used for texture discrimination and segmentation.

He and Wang[93] proposed a new set of texture measures derived from the texture spectrum of an image. The authors stated that a texture image is considered as a set of small units, known as texture units, which characterise the local texture information for a pixel and its neighbourhood. Texture spectrum is termed as the frequency distribution of all the texture units. Based on texture spectrum, various features are extracted from a texture image. The features extracted include: black-white symmetry, geometric symmetry, degree of direction, orientational features, and central symmetry. These features extract textural information of an image with a complete set of texture characteristics in all eight directions instead of using only the row texture spectrum as used in other studies. The authors performed two sets of experiments to prove the validity of the method. In the first experiment, four oriental features, one central-symmetry feature, and one directionality feature were used to discriminate between six different textures from the Brodatz album. The textures were discriminated successfully by using these measures. In the second experiment, the performance of texture spectrum method was compared with the co-occurrence matrices on SAR images. Only three features, namely black-white symmetry, geometric symmetry, and degree of direction, were used as compared to five of Haralick's texture measures. The proposed method performed significantly better than Haralick's measures.

He and Wang[94] extended the previous study by proposing unsupervised texture classification using texture spectrum. In this case again the same features are extracted from a texture image. A clustering algorithm is then used for unsupervised classification. The user provides the number of clusters and a certain threshold, which is then used iteratively to form best clusters. The main aim of this method is to use texture spectrum alone as the texture measure of the image. The method was evaluated on the same set of six images from the Brodatz album. The recognition rate of 98% was achieved in discriminating the six textures. The authors show that the algorithm requires only few initial parameters. So it can be easily used in practice for processing a large amount of data without supervision.

He and Wang[95] proposed an application of the texture spectrum method for edge detection. Texture spectrum method is a statistical approach for texture analysis, where the local texture for a given pixel and its neighbourhood is characterised by the corresponding texture unit, and an image can be characterised by its texture spectrum. Texture spectrum is the occurrence frequency of all the texture units in an image. In this study texture spectrum features are combined with traditional edge detection operators i.e. when applying a traditional edge detection operator over an image, the grey-level of each element of the operator is replaced by the texture spectrum calculated over a neighbourhood. The authors used images from the Brodatz album and calculated texture spectrum for 30x30 pixel moving windows. Edges were then detected using Roberts edge detection operator. The results were then made visible by stretching the grey-levels of the image between 0 and 255. Better results were obtained using this method than using edge detection operators on their own.

Chen and Chen[39] detail the traditional Gabor filter originally proposed by Gabor[69] and propose a modified version that works at different image resolutions. The traditional Gabor filter is applied to an image for extracting texture by convolving them. The procedure involves taking the Fourier transform of the image and the Gabor function, taking the convolution, and then taking an inverse transform. By this process, different frequency subbands are highlighted depending on the Gabor filter parameters and the resultant image can be analysed for detecting features that describe texture properties corresponding to different frequencies. In the case of multi-resolution Gabor filters, low-pass filter is applied as the finest resolution (level 0), the next high-pass filter at level 1 by subsampling the image, and so on. At each step, the Gabor filter is convolved in four separate directions, and in the resultant image, the authors compute mean and variance as texture features. The system is 58% faster in terms of computational time compared to calculating the same features on standard Gabor filter implementation. In their experiments, nine test images of size 512x512 pixels are further subdivided into a total of nine parts giving a total of 81 subimages. The single nearest neighbour classification with leave-one-out cross validation shows that both the multiresolution and traditional Gabor filters achieve the same accuracy of 98.8%. The main advantage of the multiresolution method is its faster computational speed.

Haley and Manjunath[83] describe a method of rotation invariant texture discrimination based on Gabor filters. The paper develops a 2D Gabor wavelet polar representation and a multiresolution family of these wavelets is used to compute microfeatures. These microfeatures characterise the spatially located amplitude, frequency and the directional behaviour of texture. In addition, macrofeatures are derived from the estimated selected parameters of the micromodel. A rotation invariant set of macrofeatures is used for classification. The results are presented for recognising textures at various rotations. Classification accuracy of 96.8% is achieved on a total of 624 sample images. The complete Brodatz album has also been analysed and a classification accuracy of 80.4% on 872 sample images is reported.

Other methods

Mitchell et al.[143] proposed the use of the number of grey level extrema per unit as a texture feature. Directional backlash smoothing is applied to eliminate minor local extrema, i.e. only succeeding maxima and minima which exceed a certain difference threshold are counted. These extrema are counted for two threshold values, and the texture feature is computed by dividing the difference of the numbers of extrema by the difference of the threshold values. This procedure is followed for a certain direction using a moving window. A space-filling curve that combines horizontal and vertical scanning directions is used at the end to find texture of an image. The authors adapted a method to avoid transitions due to the concatenation of the line or column segments from the moving window. Various experiments were performed on images and the results proved to be well suited to compute the texture of images with both isotropic and non-isotropic texture.

Often, texture methods have been devised based on our understanding of how human vision works. Tamura et al.[204] proposed textural features corresponding to the visual perception. These are based on six basic textural properties, i.e. coarseness, contrast, directionality, line-likeness, regularity, and roughness. The paper describes the psychological experiments on

basic textural properties and discusses the results to assess the corresponding computational measures derived from the psychological specifications. A total of 48 human subjects are presented with a total of 16 Brodatz images in pairs and for each pair the subjects are asked to make decisions on six specifications, i.e. to choose the coarser, the higher in contrast, the more directional, the more line-like, the more regular, and the rougher pattern of each pair. For each specification, the pairs are rank ordered. A correlation matrix is derived across these six specifications. There appears to be a strong correlation between coarseness, contrast and roughness. Also the correlation between line likeness and directionality is high. The authors used basic definitions of the six textural properties, either proposed by other research, or defined by the authors themselves, and modify the texture feature calculations such that the results show similar correlation as achieved in the psychological experiment. Best results are achieved for coarseness, contrast and directionality. On others, more work is needed.

Similarly, Kruizinga and Petkov[120] propose a non-linear model for texture analysis based on the neurophysiological studies on how the visual cortex of monkeys works. The model has been inspired by the recent discovery of an orientation-selective neuron in the visual cortex of monkeys called the grating cell. These cells respond vigorously to a grating of bars of appropriate orientation, position and periodicity. The model consists of two stages. In the first stage, the responses of so-called grating subunits are computed as the input to the responses of centre-on and centre-off simple cells with symmetrical receptive fields. The unit is activated by a set of three bars with appropriate periodicity, orientation, and position. In the second stage, the responses of the grating subunits of a given preferred orientation and periodicity within a certain area are added together to compute the response of the grating cell. The texture feature operators are based on grating cell responses that are obtained on the application of grating operators with eight preferred orientations and three preferred periodicities. This yields a 24 dimensional texture vector. The results of using these for classification of images have been compared in this study with Gabor features and co-occurrence matrix features using Fisher linear discriminant analysis. The authors find that on average the relative distance between the feature vector clusters obtained with the grating cell operator was twice as large as the relative distance between the clusters obtained with Gabor-energy operator, and about three times as large as the distance between the clusters resulting from the co-occurrence matrix operator.

Tomita et al.[207] introduce a structural analysis system for describing natural textures. The system has three objectives. First, to learn textures, second to classify an unknown texture, and third to reconstruct texture. For this purpose, the images are preprocessed by first computing the edge value and edge direction that are then used for nonmaximal suppression. In the resultant images, texture elements are defined as regions of homogeneous properties. For each region, the features brightness, area, size, elongatedness, and curvature are extracted. It is assumed that if there are some clusters in the distribution of the properties of elements, then they represent different kinds of elements which can be defined by recursive thresholding of property histograms. On the x-axis of such a histogram, we have the property and the y-axis is the frequency of that property. For a total of N classes, the system computes the mean and standard deviation of property p of elements in a given class, the density of elements in that class, and adjacency probability between the given class and others for defining texture. Brodatz images

are used for classification. A total of 16 textures are divided into 4x4 subimages of 64x64 pixels each. The test results show accuracy of recognition lying between 58% correct to 100% correct

Picard et al.[171] apply two powerful recognition algorithms, principal components analysis and autoregressive models, to the entire Brodatz database. The data is derived from Brodatz album that is formed by cropping nine 128x128 pixel subimages from the centre of 111 different original 8 bit 512x512 images resulting in a total of 999 different images. It is acknowledged that excellent performances obtained on small subsets will not translate to the whole database. The principal components technique has been made shift invariant for the purposes of this study by discarding phase information. The features include 99 projection coefficients onto a set of eigenfunctions. The eigenfunctions are from a pooled covariance matrix of a randomly chosen set of 100 images. The autoregressive model is characterised by five parameters at each level of resolution. At any one level, the model is shift invariant. A simple classifier based on Mahalanobis distance is used. The results show that the 15 autoregressive model features obtain a best performance of over 90% accuracy whereas the 20 principal components achieve a best recognition of less than 80%.

Ng et al.[148] demonstrate how composite features can give better results than individual features. Composite feature vectors are defined by concatenating two feature vectors. A composite Euclidean distance between features from two class distributions using composite features is defined. Also an extended nearest neighbour rule can be used instead for allocating test samples to the winning class. The decision using the extended rule is based on highest collective confidence across all features. The authors generate composite feature vectors from the following features: Fourier and Walsh transform features, grey level functions from first order and second order histograms, and statistical geometric features from binary image stacks such as the number of connected regions and irregularities. The feature combination is tested on the entire Brodatz album of 112 images with 16 samples taken from each image. The authors introduce a new measure of recognition success called total number of perfect classes, where a perfect class is defined as the one whose all samples are correctly classified. It is shown that the combination of features with weak correlation results in the best performance. The best recognition rate of 90.2% is obtained with 71 perfect classes.

Kaplan[115] proposed an extended fractal analysis approach to texture classification and segmentation. The author states that features from methods such as Gabor transforms provide a compact description of the harmonics in texture using local linear transforms. These techniques succeed in classifying a variety of textures but fail to distinguish a variety of natural textures that do not show any periodic nature. As natural textures do not contain any detectable quasi-periodic structure, other alternatives are needed. This paper proposes a method based on Brownian motion model characterised by Hurst parameter. This study compares the classification results based on Gabor features and fractal features. For analysis, Vistex and Brodatz images have been used. The study concludes that multi-scale Hurst parameters allow better texture discrimination than traditional methods and the performance using 10 such features was as high as that obtained using 48 Gabor features (on average Hurst result of 85.6% compared to 89.4% for Gabor features for one of the experiments and in another 86.2% and 87.6% respectively).

2.2.2 Comparative studies and reviews

A range of texture techniques based on the spatial or spectral domain has been used in literature. Early psychological studies suggested that human vision utilises statistical moments of distributions of grey level values for texture recognition. This led to the development of methods exploiting second order image statistics such as co-occurrence matrices. Neurophysiological studies have, at the same time, supported the view that human vision involves Fourier like decomposition of the visual stimuli into spatial frequency components representation of texture in the form of either the energy of the output of a bank of filters tuned to different spatial frequency bands, or to power spectrum itself. The following studies compare texture methods that are based on its analysis in the spatial or spectral domain.

Some basis of comparison is however needed. Texture feature evaluation methods describe how the texture between two images can be compared. Faugeras and Pratt[61] define the most popular methods. The three key methods include synthesis, classification, and figure of merit. In the case of synthesis method, an artificial texture field is created on the basis of texture feature parameters that are obtained from the original field and some error functional is then performed on the original and reconstructed fields. The basic philosophy is that reconstruction error should be small for good texture measures. In classification methods, it involves the prediction of the classification error of independently categorised texture fields. In the third method of synthesis, some functional distance measures between texture classes is developed in terms of feature parameters such that a large distance implies low classification error and vice-versa.

Compared to one and another, each evaluation method has its own merits. The synthesis method gives high evaluation to information preserving textures but such measures are likely to be complicated (e.g. based on two-dimensional histograms). At the same time, non-information preserving measures may be adequate for image segmentation and classification. Theoretically, the use of classification error is an appealing measure as it can be utilised for parameter optimisation. Unfortunately, however, it is very difficult to establish the relationship between classification error and feature value. Also, the disadvantage is that a large amount of data needs to be analysed and the results depend on the integrity of data. Another difficulty lies with the classification error not being a function only of the features but also of the choice of the classifier. The figure of merit approach has the advantage that it does not depend on any classification scheme, and in addition, error is bounded by the figure of merit for some classifiers.

Weszka et al.[223] compared texture measures for terrain classification. The features used for comparison include those categorised into four sets as Fourier power spectrum, second order grey level statistics, grey level difference statistics, and grey level run length statistics. The authors compute a total of 64 features for 9 classes using 54 aerial images. The authors have used a simple Fisher linear discriminant technique. The authors find that when a single feature is used, the classification results are better than chance but not very good. Much better results are obtained when two features are used in combination for each of the four sets. A total of 120 feature pairings are thus tested. They find that the best pair performs a recognition accuracy of 75% correct. For feature pairs they find that those based on difference statistics seems to have done significantly better than Fourier features, which perform significantly better than second

order statistics and run length. The best feature pairings are identified in this paper. Next, the authors equalise some of the previously used features with respect to size and orientation. The features are calculated for four sizes and four directions. Once more feature pairs perform better than single features on classifying texture. Compared to unequalised features, the results on second order grey level statistics feature set are vastly improved giving the best recognition of 80% correct. In addition to these analyses on synthetic images, the authors also discuss the main study related to the classification of terrain samples taken from satellite images. A total of 60 windows of size 64x64 pixels are used for calculating features. Seven sets of 16 features each, equalised in size and orientation sensitivity, are used. As before the same scheme of classification using single and paired features is used and a number of important conclusions are drawn. The recognition accuracy lies between 75% correct for single features and 93% correct for the best pair. One of the key conclusions of this work is that the statistical features capture better the essence of texture than Fourier features. Also, grey level statistics gave better results for larger sizes and distances than statistics based on single grey levels. This is because at larger distances, single grey levels are relatively uncorrelated so that the data becomes noisy, whereas the averages remain correlated, since they arise from adjacent neighbourhoods.

In 1977, Parikh published a comparative study of cloud classification [162]. The study was designed to compare the utility of infrared vs. visible features, texture vs. spectral features, linear vs. quadratic classification, and the difference between hierarchical vs. single stage decision logic. The study was based on satellite images where clouds were specified as of type *low* (86 samples), *mix* (87 samples), *cirrus* (46 samples) or *cumulonimbus* (24 samples) classes. Two studies were conducted. First study contained the classification of all classes and the second study only the classification of the first three classes. The study used a total of 45 spectral features including mean, standard deviation, and features based on cumulative frequencies. The textural features computed are based on those proposed by Weszka et al.[223]. These features measure factors such as the amount of total variation within the cloud sample and the overall homogeneity of the sample data. A joint probability distribution of the difference in grey levels separated in direction θ and distance r can be generated. On the basis of this, four texture measures are computed: mean, contrast, angular second moment and entropy. In addition, features that are independent of direction are also included. Mean, standard deviation, minimum, maximum and range are measured in all four directions for $r = 1, 2, 4, 8$. For a maximum likelihood based classifier using Fisher distances, 91% classification accuracy is achieved on the four class problem and 98% success on the three class problem on the training set. The author concludes that the spectral features are by far the most important and the textural features add little to the recognition performance. Also, it is concluded that the classifiers tested are similar in performance. However, this is contrary to research published in subsequent years and this result might well be because of a small data set.

Connors and Harlow[46] compare four different texture algorithms including Spatial Grey Level Dependence Method (SGLDM) or co-occurrence matrices, Grey Level Run Length Method (GLRLM), Grey Level Difference Method (GLDM) and Power Spectral Method (PSM). The evaluation method used here differs from other studies as it does not compare recognition performances across different methods. Instead, what is examined is the amount of texture context information contained by these algorithms. First analysis is performed on Markov

generated textures. The second analysis extends this to textures generated by translation stationary random fields of order two which generates a broader class of textures. The following texture features are used. For SGLDM method, second order probability density function is considered in the 0, 45, 90, 135 degrees direction. The knowledge of distribution in directions 180, 225, 270 and 315 degrees does not add any further information to the analysis of co-occurrence matrices. Five texture features are selected including energy, entropy, correlation, local homogeneity, and inertia. For the GLRLM method, run length features include short run emphasis, long run emphasis, grey level distribution, run length distribution, and run percentages. For GLDM method, features include contrast, angular second moment, entropy, mean, and inverse difference moment. For PSM method the features used include annular-ring sampling geometry, wedge sampling geometry, and parallel-slit sampling geometry. The evaluation procedure involves finding all texture pairs that can be discriminated using one algorithm. Say this set of pairs is A. Similarly we can find all texture pairs that are discriminated (at one time one texture discriminated from another) by the second algorithm and let us call it B. If A is a subset of B, then the algorithm that can discriminate all B is more powerful than algorithm that can separate all A. If they are totally disjoint then the two algorithms can not be compared. The study draws some important conclusions. The SGLDM method cannot innately discriminate between all visually distinct Markov texture pairs for certain distances. The five measures computed from the co-occurrence matrix do not contain all of the important texture-context information available in the matrix. For GRLM method, it cannot discriminate between a Markov texture from its 180 degree rotated version. The same applies to GLDM and PSM methods. GRLM is also very sensitive to noise. Comparing the merits of the four algorithms, the authors note that SGLDM is the most powerful of all methods considered. A Venn diagram shows the overlap between different methods in terms of the texture pairs that they can discriminate is shown.

VanGool et al.[213] present a survey of texture analysis methods. They discuss various statistical and structural approaches to texture extraction. In the case of statistical approaches, the methods detailed include co-occurrence matrices, grey level difference method, grey level run length, Fourier power spectrum, autocorrelation, filter masks, random walk procedures, and methods based on texture models. For structural approaches they discuss techniques based on placement rules and primitive extraction, and syntactic approach. An overview of reported results by other studies is also provided. The authors recommend that texture features should correspond more closely with the human perception of texture which characterises features such as coarseness, directionality, etc. Also, the dimensionality of features extracted should be limited for reasons of computational ease of processing. At one end we have very computationally demanding procedures such as co-occurrence matrices and at the other end fractal based models that yield low dimensional features with much less computation. The authors also detail texture as a basis of segmentation reviewing studies on edge and region based segmentation.

Buf et al.[27] presented a comparative study of texture features, with particular emphasis to unsupervised image segmentation. A benchmark test is introduced in which a set of 20 simple images is used for feature extraction and segmentation. The authors have used three texture decomposition methods, two methods based on second order statistics, and two ad hoc methods

for texture extraction. The texture methods used were co-occurrence method, fractal dimension, Laws, Hadamard masks, spectral decomposition, and grey-level extrema method. The segmentation method adopted in the study is based on quadtree smoothing, developed by Spann and Wilson[200]. A feature image is expanded using a quadtree structure and a local clustering is performed. Boundaries found by pixel classification at that level are refined for each step going down one level in the tree, until the original image size is reached. The experiments were performed on the set of 20 texture images. The accuracy of the segmentation result, expressed in the mean boundary error, was used as the evaluation criterion. From the seven feature extraction methods tested, Haralick's method, Laws masks and Hadamard masks gave best overall results. Results obtained also proved that the direct feature statistics such as Battacharyya distance are not appropriate evaluation criteria if texture features are used for image segmentation. The authors also concluded that the combination of features from different methods does not always guarantee an improvement of the segmentation result.

Ohanian and Dubes[150] studied the performance of four types of texture feature extraction methods: Markov Random Field Parameters, Gabor multi-channel features, fractal-based features and co-occurrence features. They used four classes of images in experiments: fractal images, Gaussian Markov Random Field (GMRF) images, leather images, and painted images. Each image class contained four types of images: the synthetic fractals, and GMRF images (generated by using different parameter values) and the natural images represented different types of leather and painted surfaces. The images used for experimentation were of size 32x32 pixels with 4 or 8 grey-levels. With these images four 4class problems and one 16-class problem were established. Whitney's forward selection method was used for feature selection and a k nearest neighbour ($k=9$) decision rule was used for classification. The co-occurrence features outperformed other features followed by the fractal features. The authors stated that the co-occurrence features should always be considered with small window sizes. Using features in more directions improves the classification results. The main drawback of co-occurrence features is the huge number of potential features and guiding theory to use only specific features for a particular problem. The overall recognition rates obtained were: co-occurrence matrices (95%), fractal features (91%), Markov Random Field, and Gabor features had correct recognition of only 65%. The authors concluded that co-occurrence and fractal features should be considered for small image sizes and Gabor features for images with large sizes.

Reed and Buf[180] presented a review of the texture segmentation and feature extraction techniques. The authors presented techniques for all three categories of feature extraction methods namely feature-based, model-based and structural. In case of feature based segmentation methods the studies include Laws masks, co-occurrence matrices, Hadamard masks, Chen and Pavlidis[40] method, grey-level dependency matrix, and transform domain features. In case of model based methods, the studies include fractal models and stochastic models. Structural methods include region-based methods, boundary based methods, and hybrid methods. Spatial frequency methods were also included in the study. These methods include Gabor power spectrum and global power spectrum. The main aim of this study was to briefly examine recent texture segmentation techniques, with a primary focus on those, which have a potential for unsupervised applications. The authors concluded that all the techniques have

distinct application areas. A rigorous quantitative comparison of various methods is conducted which yields promising results for both stochastic and structural textures.

Compared to filtering features, co-occurrence based features were found better as reported by Strand and Taxt[201]. Augustejin[8] compares a range of features for ground cover identification in satellite images using a neural network classifier. The texture measures considered include co-occurrence matrices, grey-level differences, texture-tone analysis, features derived from the Fourier transform, and Gabor filters. In addition to the Haralick's features used in this study, grey level difference features proposed by Weszka et al. have been used. For texture tone analysis, three sets of measures were used. In the first set, they use four central moments of pixels and the deviation of grey level values. In the second set, they use six measures corresponding to the grey level differences of a pixel from its neighbours. In the third set, they use grey level averages in the segment above and below a certain grey level reference. The authors further use measures from the Fourier power spectrum. Three sets of features are derived. The first set is based on the radial distribution of values in the spectrum and it is sensitive to the texture coarseness in the image segment. Coarse textures will have higher values in the power spectrum concentrated near the origin while fine textures will spread out. The averages of power spectrum values around the origin provide a measure of texture. The second set of features are based on four statistical features of the power spectrum namely maximum magnitude, average magnitude, energy of magnitude, and variance of magnitude. Finally, the amplitudes of a selected set of frequencies are selected in the feature set. The authors also use Gabor filter features for texture. A total of 100 segments of 8x8 pixels are extracted from each TM band for each class. Raw pixel based classification is used to provide a baseline. The authors find that no universally best feature set is found. The best feature set depends on the data to be classified. In ranked order of recognition performances, they find that Fourier features are the best with roughly 90% recognition result. Co-occurrence and Gabor features come second best with performances around 86% correct recognition, followed by performances of the texture tone and grey level analysis with around 85% correct recognition.

Ojala et al.[155] compared different texture measures with classification based on feature distributions. Texture measures used in this study include grey-level difference method, Laws texture method, center-symmetric covariance measures, and local binary patterns. The primary goal was to find pairs of features that provide complementary information about the texture. In case of grey level difference method, histograms of neighbouring pixels computed in horizontal and vertical direction were chosen as features along with third feature representing absolute difference in horizontal and vertical direction, and the fourth feature accumulating difference in all four principal directions. In case of Laws method, four 3x3 masks were used. For the center-symmetric covariance matrices three measures were used. Measures used include center-symmetric autocorrelation measure with linear and rank-order versions with center-symmetric covariance measure. These three measures were introduced by Harwood et al.[92]. For local binary patterns the method introduced by Wang and He[216] was used. The method is based upon texture spectrum. Two sets of experiments were performed. In the first set of experiments, nine classes of texture taken from Brodatz album were used. The texture images were corrected for mean and standard deviation in order to minimise discrimination by overall grey-level variation. Windows of size 32x32 pixels and 16x16 pixels were considered and all the features

were used individually for classification. Then Kullback's entropy measure was computed for classifying 9000 random samples (Kullback's principle measures likelihoods that samples are from alternative texture classes, based on exact probabilities of feature values of pre-classified texture prototypes). The best performance was obtained for the local binary pattern feature followed by histogram features. Covariance measures also performed better than Laws. In the second set of experiments, the method applied by Ohanian and Dubes[150] for texture analysis was used on the same set of images. The images were again corrected for mean and standard deviation and the same process was repeated for 200 samples for a 4-class problem and 16-class problem, again using single features. Difference histogram method performed best for the both problems followed by covariance features while Laws features performed the worst. The authors concluded that distributions of feature values should be used instead of single values. The use of complementary measures also improves the results. Quite poor performance of Laws approach indicates that the discriminating power of these measures is mostly contained in the variances of the feature distributions, and they also need larger data sets to perform better.

Pichler et al.[173] compare wavelet transforms with adaptive Gabor filtering feature extraction and report superior results using Gabor technique. However, the computational requirements are much larger for these than needed for wavelet transform, and in certain applications accuracy may be compromised for a faster algorithm.

Smith and Burns[198] proposed a framework for comparing texture classification algorithms and also for measuring their accuracy. The framework consists of several suits of texture classification problems and a method for computing a score for each algorithm. The framework provides a way of standardising the algorithm results. It can be used to benchmark the performance of any combination of features and classifiers. The authors also provided a set of images as Meastex texture benchmark. The benchmark contains images divided into five different classes namely asphalt, concrete, grass, miscellaneous and rock. Each image in the database is 512x512 pixels. To demonstrate the usefulness of the framework's quantitative measure a multivariate Gaussian classifier with Gabor energy features was implemented. The effect of feature set dimension was investigated by varying the orientation resolution of features. The framework gave consistent results over a range of test-suits, even where the absolute variations in accuracy are not large. The average classification scores and confidence-based classification scores were computed. The experiments proved that the confidence-based scores had higher percentages of correct scores. The authors concluded that the synthetic textures in the database offer an advantage to control the difficulty of texture problems. The framework can be easily used for quantitative comparison of texture classification algorithms. The framework is modular and additional test suits can be incorporated without modification of the current structure and the test suits can be available on the internet for the benefit of other researchers.

Randen and Husøy[179] provide a detailed comparative study of various filtering approaches to texture extraction. The basic assumption for most filtering approaches is that the energy distribution in the frequency domain defines a texture. Hence, if the frequency spectrum is decomposed into sufficient number of subbands, the spectral energy signatures for these are very different for different textures. The approaches compared in this study include Law's masks, ring/wedge filters, dyadic Gabor filter banks, wavelet transforms, wavelet packets and

frames, Quadrature Mirror Filters (QMF), Discrete Cosine Transform (DCT), eigen filters, optimised Gabor filters, linear predictors, and optimised finite impulse response filters. A summary of other studies comparing texture algorithms is also provided. The basic procedure for filtering is as follows. First, the input image is subjected to filtering that allows certain frequencies to pass through and blocks the remaining frequencies. A non-linear function is applied to the filtered image which rectifies the filter response followed by smoothing. Some of the commonly used non-linearities are magnitude, squaring, logarithm, and the rectified sigmoid. Mostly, smoothing functions include rectangular and Gaussian functions. The resultant feature image is used for classification along with other such images derived with different choices of filters and post-processing functions. The experiments were conducted on composite texture images. The authors find that no clear hierarchy of classification performances is observed. Different methods perform better on different images. The traditional Law's method and ring/wedge filters are never the winners or stand out as being very good. The poor performances are also observed for Gabor filter bank and DCT. DCT however has the least computational complexity of all methods. The QMF and wavelet frame approaches are among the best for most images. Co-occurrence matrix method is also compared and found to be the worst in all experiments.

Four filtering methods of texture discrimination have been compared by Chen and Chen[38]. These methods include Fourier transform, spatial filter, Gabor filter and wavelet transform. A total of six texture images are divided into one hundred 128x128 pixel subimages from each texture giving a data set of 600 images. Leave-one-out classification technique using nearest neighbours has been used. They find that wavelet and Gabor features perform equally well but the wavelet method is computationally less intensive. These two performances are better than the two other feature methods.

Fioravanti et al.[62] compare the spectral and rank order approaches to texture analysis. They define Wigner distribution for texture analysis. This distribution provides a cojoint spatial/spatial frequency representation of the texture pattern. The rank ordered approach acts on first order image statistics and morphological aspects. The results are compared on four Brodatz images that have cracks superimposed. The aim is to determine which approach is the best in finding these cracks. The Wigner distribution approach is found to be superior on this task than the rank ordered texture analysis approach.

Tuceyran and Jain[209] present different definitions of texture discussing the various applications where the study of texture is important. A survey of relevant studies in the areas of inspection of materials, medical image analysis, document processing, and remote sensing is provided. Next, a taxonomy of various texture models is presented including statistical methods, geometrical methods, and model based methods of signal processing. In the case of statistical methods, autocorrelation features and co-occurrence features are described in detail. The authors conclude that co-occurrence features are better suited for texture analysis rather than image segmentation. Also, they conclude that autocorrelation features can be used to assess the amount of regularity as well as the fineness or coarseness of the texture present. Finally, the texture problems are categorised into four broad categories: texture segmentation, texture

classification, texture synthesis, and shape from texture. The authors demonstrate the use of low level vision algorithms for texture analysis on SAR images.

A general review on texture analysis can be found in [51,87,90,209]. The problem of analysing texture in images and segmenting them has been tackled in Pietikänen[174].

2.2.3 Texture similarity and salience

Some studies are primarily interested in studying texture for determining texture features that is not ultimately needed for classification. For example, we might be simply interested in knowing whether an image has significant amount of texture, and if so, its orientation, location, etc. This can be used for focussing attention at specific parts of the image. Content based image retrieval is also mostly based on matching object textures where nearest neighbour techniques can be used [60]. Here we review some of the studies in these areas.

Sadjadi[185] presented a comparison of four separability measures for registration of two dimensional digital images. Image registration is the problem of matching an image denoted as a reference with a usually much broader and differently obtained picture. The main objective of the experiment was to decide which of these image views would perform better when used as a reference in a scene matching problem. The four separability measures used were Bayes probability of error, Chernoff bound, Bhattacharyya bound and Fisher's criteria. All of the four separability measures were used for the general two category Gaussian density classification problem to find the best view. Four sets of image data of 128x128 pixel size and 64 grey levels were used in the experiment. The pictures were the down-looking and target-looking views of a building and their corresponding synthetic pictures. Target-looking is the term used when the camera is at an oblique angle while down-looking is the term when the camera is directly over the target scene. The reference image in each set was selected such that the target area was totally included in it. A matching area of 9 pixels was chosen for the estimation of the correct match statistics. Correlation functions for area, edge and variations of mean were computed and then separability measures were derived and a comparison of the resulting error probabilities was made. The experimental results show that for the real images, the target looking view performs better, for the synthetic images the down-looking view performs better. The comparisons of the various probabilities of error show that Fisher's criteria produces the highest probability of error and Bayes the smallest with Chernoff and Bhattacharyya measures lie in between the two.

The orientation of texture can be used in itself as a texture classification attribute, or used for grading or detecting oriented texture. Chaudhuri et al.[35] define a technique based on Hough transform for calculating texture orientation. In this simple method, an edge image is formed first using Laplacian of Gaussian approach. An orientation histogram of dominant local orientation is constructed from the edge image. Next, the algorithm detects the peaks and valleys of the histogram. The measure of texture orientation corresponds to the height and width of peaks. The authors show results on 16 texture images from Brodatz album of size 128x128 pixels.

Picard and Kabir[172] proposed a method for finding similar patterns in large image databases. The method is used for automatic searching through large sets of images to find a pattern. A mean-squared error or weighted mean squared error is used to measure closeness of an image with the provided database. The authors have used eigen-filters and principal components of the texture covariance from the image as features. This method for feature extraction is also termed as Karhunen-Loeve transform or Principal Components Analysis. Principal components are computed directly from the covariance estimate of the spatial data. The performance of the shift-invariant principal components algorithm was characterised for each pattern and for various subsets of features. The method was applied to all 111 images from the Brodatz album. A set of nine 128x128 images was obtained from each image. The experiment was performed on 999 samples. The results were found to vary for difficult and easy images. The authors have also presented an ordering of all of the Brodatz images from difficult to easy based upon their recognition. The data can thus be used as a benchmark for comparing alternative algorithms.

Azencott et al.[9] define a texture similarity measure based on symmetrized Kullback distance. This measure forms the basis of a simple minimum distance classifier for texture discrimination. Texture is considered as a random Gaussian field that is measured using windowed Fourier filters. The symmetrized Kullback distance is the sum of distances in both directions when comparing two probability distributions. For developing a classifier based on this, a supervised approach is followed. From the training set, feature centroids are found that act as reference points for test cases. For any new image, the classifier measures the distances between the features and each centroid, and allocates the image to the class with minimal distance. This scheme is compared to the normalised quadratic distance scheme. The study uses 16 images from the Brodatz album and divides them into 49 subimages of size 32x32 pixels with an overlap of 16 pixels between adjacent windows. The proposed scheme performs better than the quadratic distance measure by 5% better accuracy showing 95.1% correct recognition.

Santini and Jain[187] discuss similarity measures between textures. They use fuzzy logic to compute distances between features. This allows the modelling of the interference between the features on which similarity is based. They use a total of 100 images from the Vistex texture database. Textures are characterised using Gabor filters. Euclidean distance, and their proposed Fuzzy Feature Contrast similarity measure are compared. They find that Euclidean distance suffers from the problem of different feature scales. Their comparison of raw and normalised data using this distance showed that the results are not very different, although the normalised data gives slightly better performance. The performances are compared on a set of 10 images retrieved by the two distance measures and a human subject based on texture similarity.

In scene analysis, quite often we are confronted with images that contain too many objects or regions that may not be important. If we can rank different image regions on the basis of the quality of texture, then we can focus our attention only on those perceptually important regions that have high texture saliency. Also, this technique can be employed in visual tracking, surveillance and indexing image databases. Syeda-Mahmood[203] presents a measure of texture saliency ranking. This method is based on inferring such information from bright and dark regions within a texture. The following attributes are computed to find the texture saliency measure within a texture field: the number of holes, maximum number of holes within a white

region, area occupied by holes within the white region, shape of black regions and distribution of holes within white regions. These attributes are weighted by coefficients that have been derived from psychophysical experiments. Results are shown on Brodatz and natural images. Similarly in cluttered scenes, it may be useful to focus attention on specific parts of the image. This is especially important when we want to locate specific objects quickly or when we want to process only a specific part of the image. Itti et al.[105] and Itti and Koch[106] define how saliency maps can be formed by combining a number of feature maps computed on an image on the basis of different features. Neural networks or other classifiers can be trained on these for supervised location of areas of interest.

2.3 Classification

The performance of a scene analysis system relies heavily on the quality of classifier used and issues surrounding data preparation. It has been found in numerous studies that using different data and the same classifier we get different results, and vice-versa. A number of classifiers have been used in scene analysis applications. These include nearest neighbour and Bayesian classifiers [53,68] and neural networks [21,234]). Mantas[137] presents a brief survey of methodologies in image analysis and pattern recognition. According to Mantas, the main objective of image analysis is to describe the object image into its main features. In case of matching, stereo-matching and time-varying imagery are briefly addressed. In case of image segmentation techniques like thresholding, edge detection and advanced methods including region adjacency graphs are briefly described. A brief survey is also presented on different methodologies for pattern recognition including statistical pattern recognition, syntactic pattern recognition and hybrid methods in pattern recognition. Jain et al.[109] provide a broader review of statistical pattern recognition. A range of problem domains, applications and their corresponding data representation are highlighted. The four major approaches to recognising patterns are explained including template matching, statistical approach, structural approach and neural networks. The authors include discussion on statistical pattern recognition, and neural networks excluding review of work on fuzzy techniques. The paper discusses the curse of dimensionality and the peaking phenomenon. A range of important issues is discussed including dimensionality reduction, feature extraction, and feature selection. Classifier design and combination are discussed for a range of methods. The properties of various classifiers are highlighted and their error estimation is discussed. The paper also discusses unsupervised classification concluding with a treatment on the topic of frontiers in pattern recognition showing the issues that are most likely to dominate pattern recognition research for some years.

In some applications, image classification is based on pixel-by-pixel classification. For example, Eklundh et al.[59] proposed a relaxation method for multi-spectral pixel classification. The authors compared three approaches for reducing errors in multi-spectral pixel classification. The approaches compared were pre-processing, post-processing and relaxation. In this approach, the pixels are initially classified probabilistically i.e. for each element a probability is estimated that it belongs to a certain class. Then the probabilities are all adjusted in parallel on the basis of the probability assignments at adjacent pixels. The process is iterated, and finally each pixel is classified as belonging to a class that has the maximum probability. In the experiments, the authors have *a priori* information about the correct pixel classification, and therefore the experiments are iterated until the classification error is minimised. All of the three

approaches are applied to a coloured house picture and the results are evaluated using the ground truth obtained by manual segmentation. The experiments proved that the pre-processing and post-processing techniques give a similar amount of reduction in error rate whereas the relaxation approach results in a far more substantial improvement. The authors concluded that the relaxation approach eliminates about four times as many errors as the other two stages.

In most studies, feature based classification is used. Here, the understanding and processing of data for classification is of utmost importance. In particular, research is always trying to improve classifier performances. First, there is an issue related to having too much data sometimes. In some applications, we can have too many samples. The problem of data reduction has not been fully addressed in detail in the literature. Usually, data reduction is viewed as a problem of reducing data dimensionality. However, quite often some data sets simply contain too many points and as such reducing the number of samples without affecting the characteristics of the data in terms of its distribution, is an attractive proposition. Fukunaga and Mantock[67] have investigated this problem. For their analysis, they divide the data into two partitions called store and test. A data distribution criteria based on nearest neighbour density is minimised as single samples are transferred from one set to another until no more transfers can take place. The aim is to reduce the original data by half. The experiments show the successful implementation of this technique on Gaussian simulated data.

Second, feature extraction is a key step in data analysis for improving performances. By using redundant, and too many features, the quality of classification is degraded. A number of probability distribution difference metrics have been described by Webb[219] including Bhattacharya distance, Patrick Fisher distance, divergence and Chernoff distance. Based on these, each feature can be ranked according to their importance in discriminating between any two classes. In addition to these methods, search strategies can also be used for effective subset selection. Some of these procedures include branch and bound procedure, best individual N method, Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), and Floating Search methods. Lerner et al.[129] show a comparative study of neural network based feature extraction paradigms. The results are tested on five databases. The methods tested include Multi-Layer Perceptron (MLP), Auto Associative Neural Network (AANN) and Neural network implementation of Sammon mapping. The results are compared for both exploratory data projection and classification. The results are compared with the non-linear principal component extractor. The authors draw the following conclusions. First, projection using the non-linear MLP feature extraction distorts the data structure and the underlying inter-pattern distances. However, this yields the best classification results. Second, linear models preserve the data structure but they are not a good alternative to the non-linear models in terms of their classification ability.

The fact that appropriate feature selection is important for obtaining good recognition performances is demonstrated by Aha and Blankert[3]. This study is based on the extraction of 204 features from regions of size 16x16 pixels. In order to exhaustively test the best feature subsets, a total of 2^{204} feature subsets must be evaluated which is computationally infeasible. Previous work by Blankert[20] on this data set has applied a simple feature selection approach that improved the results from 75.3% on the original features to 78.9% on a subset of 15

features. Aha and Blankert explore two search strategy combinations with two control strategies. They used the forward sequential selection and backward sequential selection as search strategies. For control strategies they use filter and wrapper. These strategies have been shown in more detail in their paper. The best strategy yields recognition rates as high as 88% correct on the 10 class problem.

Third, the design of experiment in terms of how inputs and outputs are defined can be crucial. Multistage classification using a tree structure proposed by Parikh[162] can be valuable for improving performance in terms of recognition accuracy. Different decision tree structures that describe the order in which classification takes place have been defined. Starting at the root of the tree, one child is specified as a distinct category of classification with other children grouped into one category appearing at the same level. In this study, a flat structure is compared with a hierarchical structure. The results of 89.7% correct classification for the flat structure is improved to 91.4% correct for the best hierarchical model. Also the study suggests that classification accuracy can be improved by designing discriminant functions to assume unequal *a priori* probabilities. One of the advantages of using multistage classification is that we have less number of classes at a given time to classify and as such classification error can be lower provided that the groups classified are fairly homogeneous.

Other recent studies have also used multistage classification. For example, Vailaya et al.[212] describe the application of classifying images on the basis of features including colour histogram, colour coherence vector, DCT coefficient, edge detection histogram, and edge direction coherence. A total of 171 outdoor images have been categorised by 8 human subjects into the following clusters: forests and farmlands, natural scenery and mountains, beach and water scenes, pathways, sunset/sunrise scenes, city scenes, bridges and city scenes with water, monuments, scenes of Washington DC, a mixed class of city and natural scenes, and face images. The authors recommend that rather than attempting a multiple class classification, multiple two-class classifications are better. Based on this, a hierarchical classifications scheme is used. The authors examine the relative advantages of these features based on the plot of inter-class and intra-class distributions. They find that edge based features are the most distinguishing factors. The recognition result of 93.9% is obtained with a weighted nearest neighbour classifier based on the evaluation of 2716 images. The objective was to classify images as city versus landscape. This approach is further extended to classify sunset/sunrise images vs. forest and mountain images which results in the recognition rate of 94.5%. The forest and mountain images have been further classified as the two separate classes with 91.7% accuracy. The main application of such analysis lies in the area of content based image retrieval especially for browsing images.

The above approach is justified by Fukunaga and Flick[66] when discussing the classification errors for very large number of classes in any pattern recognition problem. When the number of classes is large, the probability of error becomes very large. This is because the class is surrounded by a large number of neighbouring classes. If one has to make a choice between a given true class and a neighbour class, the pairwise error may be low. But pairwise errors tend to add and the error in selecting a true class in the midst of many neighbouring classes may be several times larger than the pairwise error. The authors propose that a group rather than single

class classification scheme is a better choice. In this case, classes are grouped according to characteristics of interest. The group error is just a fraction of single class error. Fukunaga and Flick provide standard curves to predict single class and group error.

Multistage classification in outdoor scene analysis has been used by Foresti[63]. First the scene is decomposed into few main classes that represent generic static objects (e.g. road, vegetation, house, sky, etc.) and into one class corresponding to mobile objects (e.g. vehicles and pedestrians). Then each region belonging to the mobile object class is further classified as the type of vehicle or pedestrian. Neural trees have been used for the classification problem. The nodes of such trees are perceptrons without hidden layers. The paper defines their architecture and methods of training such trees. The results are generated for both optical and infrared images. On a total of 250 optical and 50 infrared images, the authors report a recognition rate of 82% and 74% respectively on distinguishing mobile objects from static objects and a recognition rate of more than 70% correct on further classifying mobile objects.

Fourth, classifier combination can yield better results than using single classifiers. Roli et al. [182] review comparative studies of statistical classifiers and neural networks in the context of remote sensing studies. They propose that for image classification, classifier ensembles are likely to provide better results. They detail how classifier outputs can be combined using voting principles, Bayesian average, and belief functions. It is expected that combined classifier can keep the distinct advantages of classifier that are combined together. The image classification tests are carried out on a per pixel basis for five agricultural classes of data. The classifiers used include Gaussian Classifier (GC) Nearest Neighbour (NN) classifier with k ranging between 1 and 91, MLP neural nets, Radial Basis Function (RBF) neural nets, and Probabilistic Neural Network (PNN). The average performance of these classifiers individually is reported as GC (79.4%), NN (88.4%), MLP (81.6%), RBF (78.9%) and PNN (88.6%). Using the majority rule on a combination of NN, MLP and PNN, an accuracy of 90.4% is obtained which improves to 91.3% for combination by belief functions. One of the other main conclusions is that the nearest neighbour classifier performs the best.

Quite often, good data preparation is more important than the choice of classifier itself. However, it is important to understand the nature of classifiers before making a selection for object recognition task. In several studies it has been found that non-parameteric classifiers have similar performances on the same data. For example, Greenspan[76] studied the classification performances of three classifiers on texture features extracted in the frequency and orientation space. The classifiers compared include rule-based network, nearest neighbour classifier, and neural networks. For feature extraction, Gabor function and its Fourier transform on the original image are computed. Then, a Gabor wavelet decomposition scheme is employed to extract orientation and frequency responses from local areas of the input images. Three scales are used with four orientations per scale. A pyramidal representation of the image is convolved with fixed spatial support oriented Gabor filter. A measure of power or energy associated with each filtered map forms the basis of feature vectors for response on 8x8 pixel windows. The rule-based network is developed in two stages. In the first stage, features are passed through a k -means unsupervised clustering scheme to bundle them into discrete partitions that are later labelled. Using these partitions for confidence in that rule. The other two classifiers used are

standard. The results compared for different window sizes show in each case a difference of less than 3% in performance is obtained across the three classifiers. The best recognition result obtained is 100% with neural networks 32x32 pixel window on a total of 30 texture images.

Aggarwal and Shah[2] proposed a method for object recognition and performance bounds. The main purpose of the study is to look at the fundamental problems of object recognition and discuss various ways of formulation in practical object recognition systems. The authors have used three approaches: Bayesian approach, neural network approach and rule based approach for pattern recognition. The authors have presented a coherent comparison of the methods and discussed the ability of each in measuring the performance of the object recognition process by incorporating a degree of uncertainty. The authors proposed that Bayesian statistics provide a firm theoretical footing to improve the performance of a pattern recognition system and incorporate error estimates for the overall process. Neural networks are data-driven modifying patterns on inter-node connectivity and modelling a function of the training data is the learning approach. Neural networks do not require geometric models but instead they require that the set of samples used for training should come from the same distribution. The rule based systems can handle uncertain decisions by attaching a measure of belief to each of their output decisions. However, rule based systems are also limited in their ability to interpret typical knowledge bases and also to evaluate error in their decision for characterising performance. The results proved that the Bayesian paradigm in its formulation is comparable to neural networks and rule based methods if the relevant features have a Gaussian distribution. The authors concluded that an ideal object recognition system should use a combination of all the three methodologies.

Finally, it is important to know how well a system performs in terms of its ability to give consistent performances with high recognition accuracy. To know how to do an optimal design for a vision algorithm, one needs to understand for a synthesis problem how to propagate a perturbation process from the input to the output, considering images as input and the application of an algorithm or the classification end as the output. Haralick[91] comments on the propagation of errors from input to output in vision algorithms and devises a method of modelling the propagation of input covariance matrix to output covariance matrix. Liu et al.[131] discuss the importance of statistically validating computer vision software by studying random perturbations. This is based on the fact that there is inherent uncertainty associated with any computer vision algorithm. These uncertainties are best represented in terms of statistical distribution means and covariances. Often the image processing runs in several lines of code. One of the ways to check whether the software implementation and theoretical calculations are correct is to provide the algorithm with controlled input data, with known statistical characteristics, which is possible as the data is artificially generated, and check if the output is actually distributed as what is predicted by theoretical calculations. The authors state that for a distribution with q parameters, a total of $3^q - 2^q$ hypothesis tests can be performed. The authors propose different hypothesis tests when data comes from multivariate normal distribution. Kolmogorov –Smirnov test is also described here that measures if two distributions are alike.

Kanungo et al.[114] define the methodology for the quantitative performance evaluation of detection algorithms. A typical detection task relates to the identification of a target in an image. The task is to compute a number called *evidence strength* that measures the evidence whether

the target is present. The problem can be complicated by the low signal to noise ratio. There are several applications including detection of edges, objects, etc. within an image. Each evidence strength has a certain frequency and a histogram can be plotted as one way of illustrating the performance of a detection algorithm. One can define the various probabilities of false alarms and correct recognition from which an overall probability of error is computed. A plot of this against the signal to noise ratio is another useful representation of algorithm performance with change in noise. The effect of changing our acceptability criterion, that determines whether the evidence of strength is larger than it or not, we can replot the above curves. The authors also present a method of combining several operating curves into one to capture more information than otherwise possible.

2.4 Scene analysis studies

Outdoor scene analysis is a complex problem. A number of different approaches have been used for recognising different objects in such scenes. In early experiments on scene analysis, simple problems were tackled. For example Brice and Fennema[24] defined the procedure to interpret simple objects in images such as wedges, cubes, wall and floor. They define a simple procedure for grouping regions and understanding them to see their shape properties for object recognition. Also semantic information, such as the fact that there is a specific spatial relationship between floor and wall, is used for simplifies the process of scene interpretation. Model based approaches, such as the one proposed by Brooks[26], have met with some success provided that objects can be defined with geometric primitives. These approaches have problems with recognising natural objects such as trees for example where such primitives are hard to define. Another approach is based on the use of a knowledge based scheme where hand-coded rules are used for object recognition. These rules describe the characteristic properties of objects of different types. Some examples of such work include SCHEMA vision system by Draper et al.[55], region based scene analysis by Ohta[152], and VISOR connectionist system for scene analysis [128]. Although these approaches have shown reasonable results, a significant amount of computational effort is required even for simple scenes. More recently, far more complex problems are being solved using texture based image analysis. The theses of Becalick[15] and MacKeown[133] provide an excellent review treatment of other studies in this area and summarise the progress made. In particular, Becalick reviews important studies based on knowledge based approaches in scene analysis, Autonomous Ground Vehicle (AGV) research, and image database and multimedia search.

A number of studies based an automated segmentation and texture analysis has been recently applied for natural object recognition in outdoor scenes. In most studies leave-one-out classifications schemes have been used for statistical classifiers and ten fold cross-validation for neural networks, however, more advanced bootstrap procedures can also be used [43]. Based on *k*-means segmentation on a 11 class problem, Campbell et al.[29] demonstrate recognition accuracy of 81.4% correct. The work by Mackeown et al. [134,135] formed a part of British Aerospace/ Bristol University project aimed at recognising natural objects for developing autonomous land vehicles. Neural networks have been used for learning object characteristics. The studies use an image set of 40 urban and 40 rural scenes of 24 colour bit depth for a 12 class discrimination problem. A total of 29 features are used where 16 are contextual features that are derived at the segmentation stage. The 12 classes are selected after grouping originally

29 classes. On all 29 classes, the classification accuracy for 80:20 train/test set split is 51.1% and on only 12 objects it is 64.1%.

Campbell et al.[30] presented a technique for the automatic classification of outdoor images using a neural network. The technique is to segment the images, extract features for each region and then train the neural network to act as a Bayesian classifier. For segmenting the image into its major regions, k -means clustering of the grey-level histogram has been used. It is important that the algorithm produces closed regions since feature extraction is to be performed. It has been found that the use of four neighbours ($k=4$) in the k -means clustering, produced the optimum balance between under-segmentation and over-segmentation. For feature extraction, 28 features were extracted from each region including average color, position, size, and rotation. Additionally, texture, shape and context features of the region were also obtained. For classification, the optimum results were obtained using a network of size $28 \times 24 \times 11$ (i.e. 28 inputs, 24 hidden nodes and 11 outputs). The number of hidden units was found using conjugate gradient descent optimisation. The network was trained to recognise 11 possible output labels including sky, vegetation, road, wall, buildings, and so on. The data used for testing the network was extracted from the Bristol Image Database. This database consists of a large number of high quality colour images of outdoor scenes. The network was tested on unseen data from 3751 regions. Over 80% of regions were classified correctly, corresponding to 91.1% of image area. Most of the errors occurred when the classifier had to separate between road and pavement, fence, wall and buildings, and so on. Another cause for misclassifications was attributed to errors during segmentation. This paper compares the classification techniques and found that Learning Vector Quantisation (LVQ) was superior to MLP neural network with the best result of 76.7% correct classification.

The study by Campbell et al.[32] describes the classification of objects in the Bristol Image database. This database consists of 350 images of urban and rural scenes. The classification is performed on 11 natural object classes including vegetation, road marking, road, pavement, building, fence/wall, road sign, signs/poles, shadow and mobile objects. The problem is approached with pixel classification and region classification as two separate modes of object recognition. In both cases, neural networks are used as classifiers. In case of pixel based classification, only four objects namely sky, vegetation, road-like and others were classified. Each pixel is characterised by its colour information. It is also characterised by Gabor texture features based only on frequency and not orientation. The pixel intensity is also a feature. The authors find that using intensity alone, a recognition rate of 73.4% is obtained that improves to 85.0% when colour information is added and to 87.1% when texture information is added. In the case of region based classification, the images are first automatically segmented using k -means segmentation. Region based features including contextual features and shape features, in addition to all of the previously used features for pixel classification, are now computed for the segmented region. Contextual features are defined as a measure of confidence that the region is surrounded by pixels of a given class. Shape measures are obtained by finding principal components of region boundary information. The results obtained on testing over 3000 regions shows an overall recognition accuracy of 82.9% correct on regions and 91% of the image area being accurately classified.

Betke and Makris[16] describe the problem of natural object recognition from the perspective of using the theory of statistical estimation. An object is considered complex if it is composed of elaborately interconnected parts. The complexity of an object can be defined as the ratio of the outer volume to the coherence volume. This complexity information is defined as positional complexity, rotational complexity and contractional complexity. This information is based on the object's Fisher information. An information conserving method is then developed for the recognition of objects in complex scenes. The complexity is then measured for a number of traffic signs and the classification is applied to traffic sign recognition on 408 scene images. The recognition rates for nine sign models show large number of correct matches and correct negatives with small number of false positives.

A number of recent studies by Singh and his colleagues have addressed the problem of scene analysis with Forward Looking InfraRed (FLIR) images for military applications. In most of these studies, the images have been acquired as a video stream and segmented using probabilistic labelling and relaxation. Co-occurrence matrices have been decomposed as a function of Hermite coefficients that are used as texture features. In Singh et al.[188] two nearest neighbour classifier models have been proposed that are also used in this thesis. The application of these two models, one based on conflict resolution and the other on average distance across neighbours, is shown for the scene analysis problem for classifying five classes of data including river, trees, grass, and the reflection of trees and sky in river, are shown in Singh et al.[195] where an accuracy of nearly 70% is obtained using leave one out method of classification with nearest neighbour models. Much better results are reported on classifying more than ten classes using the same set-up in Singh et al.[190]. Neural networks have also been applied to this data for analysis in Singh et al.[191]. One of the key problems in dynamic scene analysis is that often new objects are found in images on which classifiers have not been trained and there is no *a priori* information available on them. A system that is capable of automatically detecting samples of completely unknown objects on which classifier has no ground truth data and issues related to its labelling have been discussed by Singh et al.[192], and Singh and Markou[193].

Battle et al.[13] provide a review of various scene analysis studies categorised as bottom-up schemes, top-down schemes and hybrid approaches. In the case of bottom-up schemes, a scene is partitioned into regions by using a segmentation technique. These regions are then characterised by a fixed set of attributes and the scene can be described in terms of relationships between objects. The labelling process uses an inference engine to match a region to object model. In the top-down approach, we start with the hypothesis that the image contains a particular type of object and then try to prove the hypothesis. The hybrid scheme is a mixture of the two. The authors review a number of studies that fall in one of the three areas. We have borrowed their table for presenting this information in Table 2.1 where the first four rows are examples of top down approaches, the next five rows examples of hybrid schemes and the other rows are examples of bottom up approaches.

In scene analysis applications, quite often we deal with video sequences rather than still images. In some of these applications, we may require image segmentation and object recognition for most applications (in other cases we might be simply interested in tracking natural objects, for

example as in tracking the road when driving [159]). In order to perform classification of video shots, it is important only to process those shots that contain regions or objects that have not been already classified. For this purpose, it is important to identify the shot transitions in video that define the introduction of new frames that are significantly different than old frames. The identification of these transitions is called video segmentation. Dailianas[47] defines various methods of identifying video short transitions and ways of ranking their performance on a video stream. The differences between successive frames can be computed on the basis of colour histogram changes, changes in moment invariants, changes in edges, model-based approach or a simple differencing technique. These methods are compared on four test videos using the results obtained by human observers as a baseline. The performance metrics include the correct identification of short transitions and false alarms.

Natural object recognition is only the first step of a complete scene understanding system. The interpretation of these natural scenes requires semantic understanding of object relationships. In one way, these relationships allow us to better understand images. At the same time, if we have already stored some contextual information from our experience with how outdoor scenes should look like, then this information can be used for guiding low level image processing operations such as image segmentation. For interpreting scenes, the role of context is paramount, and the understanding of spatial relationships between objects important.

Toussaint[208] discusses the role of context in pattern recognition especially for image classification. The role of context at perceptual, cognitive and mathematical levels is discussed and illustrated with examples. The paper surveys the techniques for using contextual information in pattern recognition. The emphasis is on text recognition as an application that can benefit from the use of context. The techniques that use context in such an application include dictionary look up methods, Markov and probability distribution approximation methods, and hybrid methods.

Scene analysis mostly consists of identifying objects reliably within an image. Each of the objects however has some spatial relationship with others and on the basis of these one can guide the scene analysis process. Only those object labelling are valid that can be derived from the arrangement of real objects in space. The projected properties and relationships constrain the possible labelling of regions with object identifications. Thus scene analysis can be viewed as a constraint satisfaction problem. Kitchen and Rosenfeld[118] define how constraint satisfaction can be used for scene analysis. A constraint network is first set up and list of possible labels is attached to the nodes. Labellings can be filtered depending on whether they violate binary constraints. Finally, labels are eliminated that violate the requirement that the arc and node labelling are consistent, as well as the labels that violate essential constraints.

Michalski et al.[142] describe the semantic interpretation of outdoor images. A number of studies on machine learning in computer vision are reviewed before tackling the problem of scene understanding. It is proposed that the combination of symbolic learning with neural networks is to yield the best solutions for classification. This is explored under the framework called MIST that stands for Multi-Level Image Sampling and Transformation. This framework has two stages of operation, the learning mode and the interpretation mode. The learning mode

uses symbolic learning and neural network classifiers. In the interpretation mode, the system applies descriptions from the image knowledge base to semantically interpret the new image. In their experiments, region features from natural images are extracted on the basis of attributes including hue, saturation, intensity, horizontal and vertical gradients, high frequency spots, and Laplacian operators. On the basis of training areas of size 10x10, 20x20 and 40x40 pixels, classification features are extracted. Accuracy between 95% and 100% is obtained for different classifiers using a random 60:40 split for training and test sets. On the basis of learned one-level descriptions from these small regions, the entire image can be semantically labelled.

In futuristic scene analysis systems, it is expected that we will be able to classify natural and synthetic objects much more accurately. In addition, hopefully better frameworks can be developed that capture object relationships and allow us to interpret scenes better. Ideally, contents of image analysis systems should be able to assess their own performance, provide feedback to other components and adjust process parameters to optimise performance for the prevailing image conditions [82].

In this chapter we have reviewed some studies to put own work into context. It is an obvious conclusion that for solving the same problem, a range of methods is available that give different results depending on application. In the next chapter we define our research methodology.

System identification	Scene	Colour space	List and number of objects to recognise	Object characteristics	Segmentation technique	Object and scene knowledge representation	Labelling engine
Campani et al.[28] & Parodi & Piccioli[163]	U	Colour	Road boundaries, road signs, cross-walks, vehicles, buildings, trees (6)	Spatial disposition in the scene, colour and segments	A specific gross segmentation based on edges, colour, vanishing point detection	Encode the specific features of the objects inside the algorithm	A specific procedure that finds specific features
Hild and Shirai[97]	N	Hue, brightness	Tree trunks, branches, grass, leaves, sky (5)	Shape, orientation, position, hue and texel direction	Likelihood pixel classification	Feature vectors	Select the best candidate by shape processing
Efenberger & Graefe[58], Regenberger and Graefe[181]	R	Grey levels	Road, tree trunk, tree, rock, barrel, car (6)	Edges and grey levels	Edges and grey levels	Subsampled images and prominent edge elements	2D correlation functions and special purposive methods
Ide et al.[104]	U	Grey levels	Poles (1)	Diameter, height and layout	Vertical straight line detection	Encode the specific features of the object to recognise inside the algorithm	Recognition based on specific characteristics
VISIONS Draper et al. (1989) & Hanson and Riseman[85]	R, H	Combination of RGB and YIQ values	Road scenes: sky, foliage, shoulder, trunk, sign-post, wire, warning sign, phonepole, road, roof, building, roadline, grass, unknown (14) House scenes: Sky, tree, grass, bush, shutter, wire, housewall, roof, roadline, road, film border (11)	Colour, texture, shape, size and spatial relations among objects	Combined histogramming and region merging methods	A schema and two graphs (part of invocation) for scenes, and schemas for the objects	Schemas based on solving specific confidence functions in order to verify hypothesis
CONDOR Strat[202]	N	R,G,B	Geometric horizon, complete sky, complete ground, skyline, sky, ground, raised object, foliage, bush, tree trunk, tree crown, tree, trall, grass (14)	Colour, texture, geometric shapes and spatial relations among objects	A set of specific context sets (rules) which include texture operators, edge operators, histogramming techniques, ...	Semantic networks and context sets (rules as pairs of conditions, actions)	(a) Candidate comparison by likelihood methods; (b) grouping mutually consistent hypothesis; (c) Select the best description
Asada and Shirai[7], Hirata et al.[98], Taniguchi et al.[205]	R	(T,q,S), brightness, hue and saturation	Road, roadlines, crosswalk, sky, trees, buildings, poles, cars, truck, not interpreted (10)	Colour, heights, range information and spatial relations among objects	Colour (they design a specific split and merge algorithm)	Each object is represented as a frame. A scene is represented as a network of frames structures	Rules in specific methods

Ohta et al.[153]	U	Combination of RGB	Sky, tree, building, road, unknown, car, car shadow, building window (8)	Colour, texture, position and shape	Region splitting using multi-histograms	Semantic network	Instantiate production rules (which the condition part is the fuzzy predicate)
Gamba et al.[71], Mecocci et al.[139]	`U	Grey levels	Natural objects, lateral vertical surfaces, frontal vertical surfaces, horizontal surfaces, vanishing point, unrecognised (6)	Segments and region localisation with respect to vanishing points	A specific region growing algorithm, edge analysis and vanishing point detection	Encode the specific features of the objects inside the algorithm	Criteria (rules)
Bajcsy and Joshi[10]	N	Colour	Ground, sky, horizon, skillines, tree (4)	Colour, sizes, and spatial relations among objects	Colour separation	Rules (named facts)	Partial match operations on rules and facts
Lavine[124], Lavine and Shaheen[125], Nazif and Lavine [147]	H	R,G,B	Bushes, car, fence, grass, road, roof, shadow, window (8)	Colour and spatial relations among objects	Region, edges and area	Rules	Constraint relations, rules in order to verify the hypothesis
Douglass[54]	H	H,S,I	Trees, house, grass, sky, car, street, window, brick-wall, concrete-wall, roof, ground (11)	Colour, texture, boundary shape, size, curvature and orientation	Edges, colour and texture (the algorithm combines edge detection and region growing)	Associative net (semantic net), where nodes are objects and links are logical and spatial relationships between objects	Probabilistic method
Kim and Yang [117]	R, U	R, g, b, r-b, intensity and saturation	Sky, foliage, road, grass, wall, roadline, window, footway, tree (9)	Spatial disposition in the scene, colour, texture and geometric features	Region growing	Feature vectors and graphs	Labelling the nodes of a region adjacency graph by using a simulated annealing method
Kumar and Desai [121]	R	Grey level	Sky, tree, sidewalk, road (4)	Spatial disposition in the scene, grey level, texture and geometric features	k-means clustering	Feature vectors and graphs	Labelling the nodes of a region adjacency graph by using a simulated annealing method
Bhanu et al.[19], and Peng and Bhanu[168]	R	R,G,B	-	-	A genetic algorithm selects parameters automatically for region splitting algorithm	-	-
Campbell et al.[32]	R	Combination of RGB	Sky, vegetation, road markings, road, pavement, building, fence/wall, road sign, signs/poles, shadow, mobile objects (11)	28 features including colour, texture (isotropic Gabor), shape and contextual information	k-means clustering	Feature vectors	Neural networks

Table 2.1 Review of outdoor scene analysis techniques by Batlle et al.[13].
(Scene: U (urban), N (natural), H (house), R (road))

Chapter 3

Methodology

The purpose of this chapter is to provide a detailed overview of the methodology employed for this study. This chapter describes the data used, its main characteristics, and an overview of the technical details of this research. Two sets of experiments were performed. The first experiment is for the evaluation of five different texture extraction algorithms on two texture benchmarks MeasTex and VisTex. The second experiment evaluates the performance of image segmentation and texture extraction algorithms on PANN database. This chapter details the methodology for both the processes with the relevant examples and parameters used.

The chapter first describes the international texture benchmarks used and the PANN database. In the next section the detailed methodology followed for both the experiments is illustrated with the flowcharts and detailed description of all the steps involved.

3.1 Data details

This section gives detailed information on the texture benchmarks and an overview of outdoor data used. The experiments are first performed on standard texture benchmarks for texture extraction algorithm evaluation, and then on real outdoor data to evaluate the differences in performance when different image segmentation algorithms are used in combination with different texture extraction methods.

3.1.1 Texture benchmarks

Two texture benchmarks called MeasTex and VisTex are used in this study for the performance evaluation of texture feature extraction methods without any segmentation. Various studies have already used the same benchmarks for testing and comparing texture algorithms. Both of these databases are publicly available for research purposes. Guy Smith's MeasTex database [138] is an image database and quantitative measurement framework for image texture analysis algorithms. VisTex database [214] is another popular collection of texture images by MIT Media lab. This database provides a large set of high quality textures for various computer vision applications. The databases are explained in detail below.

MeasTex database

MeasTex is a publicly available international database. MeasTex is about the MEASurement of TEXture classification algorithms. This texture database contains images of natural objects with homogeneous texture. The database contains several test suites of texture classification problems. It is proposed as a standard framework for the analysis of robust methods of texture feature extraction. This framework has been tested on several algorithms and Unix platforms. Other studies have implemented various texture algorithms on this database and the results are available for comparison and evaluation [198].

There are a total of 67 images of synthetic homogeneous textures. The images are divided into 5 different categories: Asphalt, Concrete, Grass, Miscellaneous and Rock.

File formats and image size

All images in the MeasTex database are stored as 512x512 pixels raw pgm (P5) images. For our analysis, each image is divided into 16 parts to get enough samples for classification. Table 3.1 shows the number of images available for analysis. We have not used any samples from the miscellaneous category. The table details the information about the total number of images available in the MeasTex database, the images used in this study, and also the total number of samples used.

Objects	Total images in the database	Images used	Samples used
Asphalt	4	4	64
Concrete	12	12	192
Grass	18	18	288
Miscellaneous	8	0	0
Rock	25	25	400

Table 3.1 MeasTex database composition and samples used.

Various examples of images from the MeasTex database are presented in Figure 3.1. The first set of images includes various samples of asphalt. The samples are quite similar to each other. The second set of images includes various samples of concrete. The third set shows different samples of grass. The fourth set shows the miscellaneous images. It is clear from the images that the miscellaneous set contains images that are quite different from each other and also the texture contained in different images is also variable in nature. So these images are not used for the analysis. The fifth set contains different images of rocks.

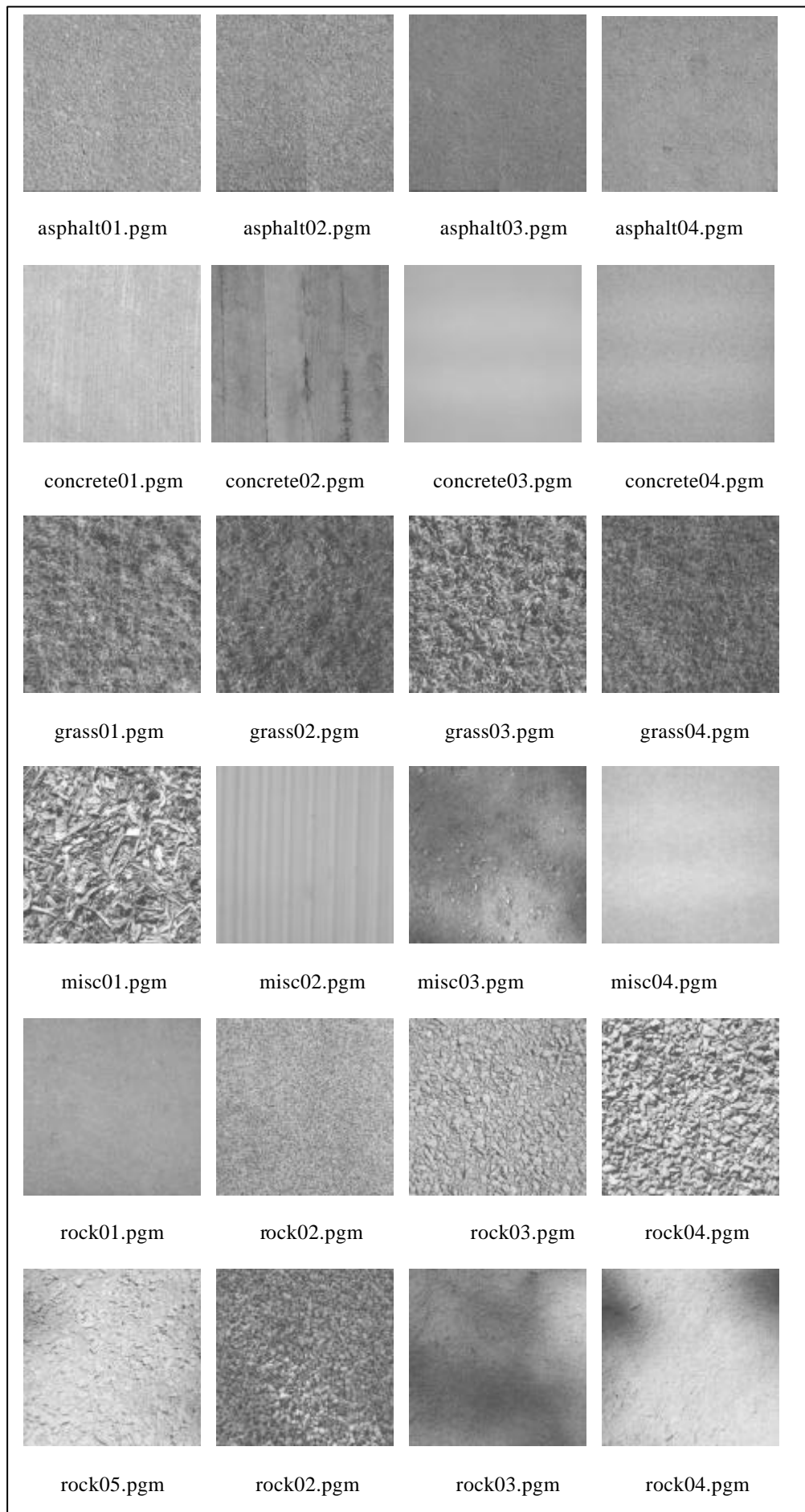


Figure 3.1 Sample images from MeasTex database.

VisTex database

Vision Texture (VisTex) database is a publicly available international database by MIT Media lab. The database contains images of 18 different natural objects. The goal of VisTex is to provide texture images that are representative of real world conditions. This database is a standard framework for measuring the accuracy of texture classification algorithms. The database consists of real world images in different lighting conditions for analysis of various texture feature classification algorithms.

There are a total of 67 images of different scenes with different lighting conditions. The images are divided into 18 different categories namely, Bark, Brick, Buildings, Clouds, Fabric, Flowers, Food, Grass, Leaves, Metal, Miscellaneous, Painting, Sand, Stone, Terrain, Tile, Water, Whereswaldo and Wood.

File format and image size

All images in the VisTex database are stored as raw ppm(P6) files. There are two standard sizes for images in VisTex: 512x512 pixels and 128x128 pixels. For our analysis, each 512x512 pixel image is divided into 4 parts to get enough samples for classification. Some objects in the database do not have enough samples so we have not used those image categories with less than five samples. In addition, some images are of extremely poor quality and we have excluded them from analysis. Also, we have not used those image categories that do not have uniform texture and would ideally need segmentation prior to feature extraction. In our study only seven objects are chosen for classification as discussed below.

Various example images that are used for analysis from the VisTex database are presented in Figure 3.2. The first set of images shows bark samples. The second set of images shows fabric samples. These samples are highly variable in their composition. The third set of images shows food samples. Texture primitives are again highly variable in this case for different samples. The fourth set of images shows various samples of metal. The fifth set shows sand samples. The sixth set of images is shown for tiles. Different samples of tiles appear different from each other. The seventh set shows different images of water.

Figure 3.3 gives some example images from the VisTex database which are not used for our analysis. There are several reasons for not using these images. The first reason is that the database contains too few samples of some categories. These categories will not yield a lot more samples even after subdivision. In order to avoid unbalancing the classification database with these categories, we have not used them. Also we have not used some of the images that have been taken under extremely poor lighting conditions. As such, good quality feature extraction from these can not be expected. Finally, we have not used those images that need segmentation. In images that contain buildings, flowers, paintings, etc., the texture is not uniform and greatly varied across different samples. Such images have been excluded from our analysis.

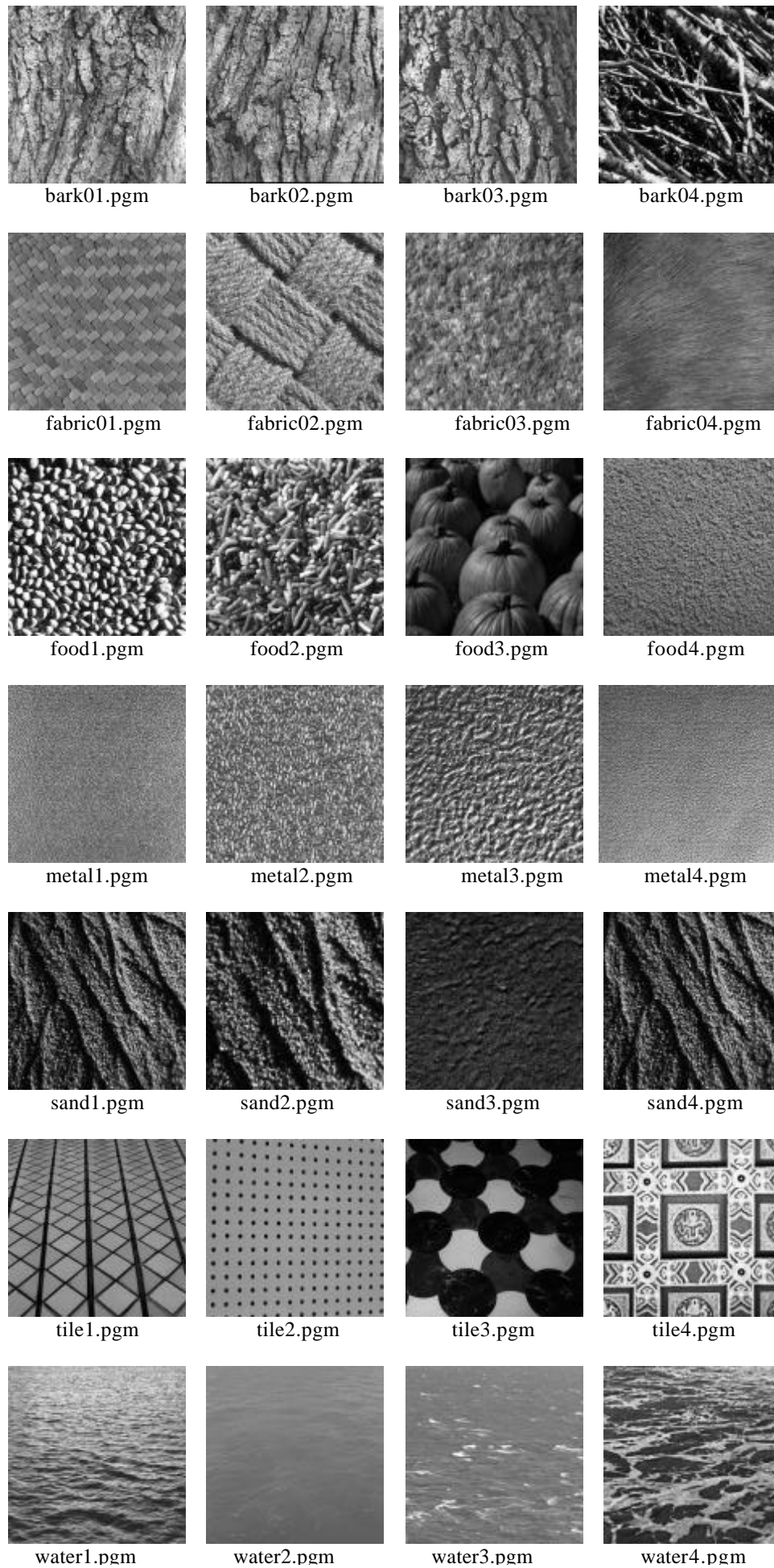


Figure 3.2 Sample images used from the VisTex database.

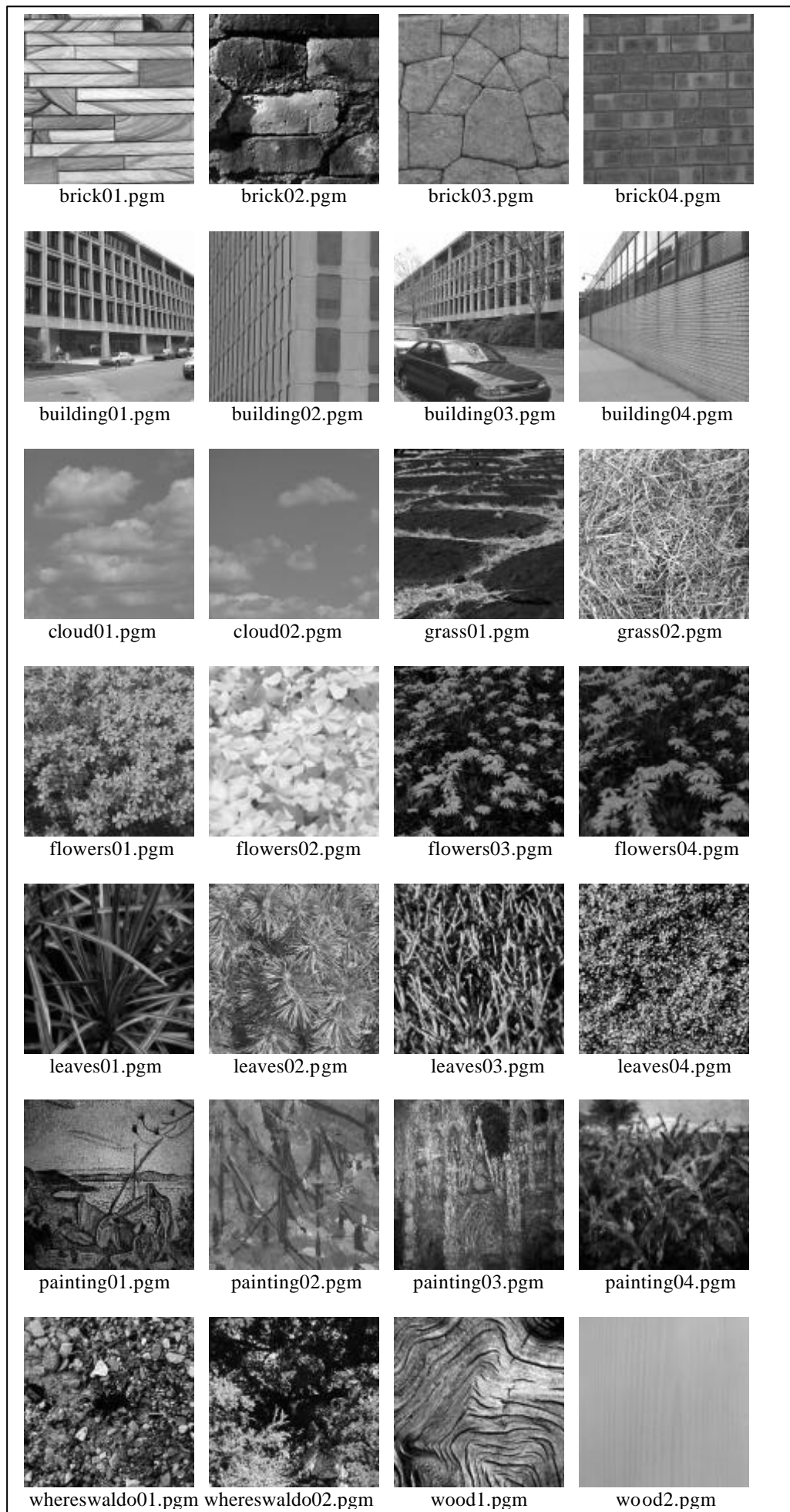


Figure 3.3 Sample images not used from the VisTex database.

Objects	Total images in the database	Images used	Samples used
Bark	13	9	36
Brick	9	0(poor quality)	0
Building	11	0(segmentation needed)	0
Clouds	2	0(< 5 samples)	0
Fabric	20	20	80
Flower	8	0(segmentation needed)	0
Food	12	12	48
Grass	3	0(< 5 samples)	0
Leaves	17	0(poor quality)	0
Metal	6	6	24
Miscellaneous	4	0(< 5 samples)	0
Painting	14	0(segmentation needed)	0
Sand	7	7	28
Stone	6	0(poor quality)	0
Terrain	11	0(poor quality)	0
Tile	11	8	32
Water	8	8	32
Wherswaldo	3	0(< 5 samples)	0
Wood	3	0(< 5 samples)	0

Table 3.2 VisTex database composition and samples used.

In Table 3.2 above we show the total number of samples available for our analysis from each category.

Other databases

In this study we have used two of the popular texture benchmarks. This is not to say that any other benchmarks are not available. For the sake of comprehensiveness, it is appropriate here to point them out and briefly summarise their characteristics.

Brodatz data

In 1966 Brodatz[25] published a photographic album of textures. A recent version of this has been printed again. Some example images are shown in Figure 3.4. A number of web sites have digitised textures from this book and thus digital texture data is available for analysis. The album contains 112 grey scale pictures. There are several advantages of using this texture data. First, this data is de facto standard for texture analysis. As we have seen in our literature survey, several studies use Brodatz images for texture analysis. Hence, there is a common understanding on how difficult these images are for classification. However, one of the problems using this data is that several different digitised collections exist and a number of authors have been very choosy in which images they have used for analysis. This makes comparison difficult. Another disadvantage is that the album was never designed in the first place for image analysis and as such does not contain all textures that are relevant for vision

research. Some links to these digitised collections can be obtained from <http://wcc.ruca.ua.ac.be/~visielab/wta/brodatz.html>.

Elena classification data

ELENA is ESPIRIT III Basic research action project (no. 6891)

Under this repository, databases have been identified as real or synthetic. The real database consists of images of 11 different textures (grass, lawn, pressed calf leather, handmade paper, raffia looped to a high pile, cotton, canvas, etc.). The feature sets on this data are also available as 40 attributes built by the estimation of fourth order moments in four orientations of 0, 45, 90 and 135 degrees. The data is available from: <ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases>

Columbia-Utrecht reflectance and texture database

Researchers at Columbia University and Utrecht University have put together a database that contains reflectance measurements for over 60 samples each observed with 200 different combinations of viewing and illumination conditions [48]. As a part of this, 60 different real-world surfaces have been used. The categories include specular surfaces (aluminum foil, artificial grass), diffuse surfaces (plaster concrete), isotropic surfaces (cork, leather, styrofoam), anisotropic surfaces (straw, corduroy, corn husk), surfaces with large height variations (sandpaper, quarry tile, brick), pastel surfaces (paper, cotton), coloured surfaces (velvet, rug), natural surfaces (moss, lettuce, fur), and man made surfaces (sponge, terrycloth). The appearance of texture is considered as a function of the viewing and illumination conditions. The data is available through <http://www.curet.cs.columbia.edu/curet/data>. This is a relatively new data set compared to other texture benchmarks and not enough studies on its analysis are available. Also the database has been developed more for the graphic community for texture rendering studies. The database acts as a starting point for exploring 3D texture rendering algorithms. As such, it remains to be seen if it will prove useful for texture discrimination studies.

3.1.2 PANN database

As a part of this project, outdoor data was collected from the University of Exeter campus using a Panasonic digital video camera (Model NV-DS77B) having 720x576 pixels resolution. The data was collected in the form of 448 coloured digital stills. Images of different natural scenes containing grass, trees, sky, clouds, pebbles, road and different samples of bricks were collected. The camera was stabilized using a tripod. All the images were taken during daytime to give a realistic view of the real environment. At the same time, every effort was made to minimise the shadow effects. A selection of images from this database is shown in Figure 3.5.

File format and image size

All of the images collected are stored in bitmap (.bmp) format with 16-bit colour depth and a resolution of 760x576 pixels. The images have been reduced to a size of 512x512 pixel resolution using the 'convert' command in Linux for further analysis. At the same time, the images have been converted in .pgm format with 256 grey levels.

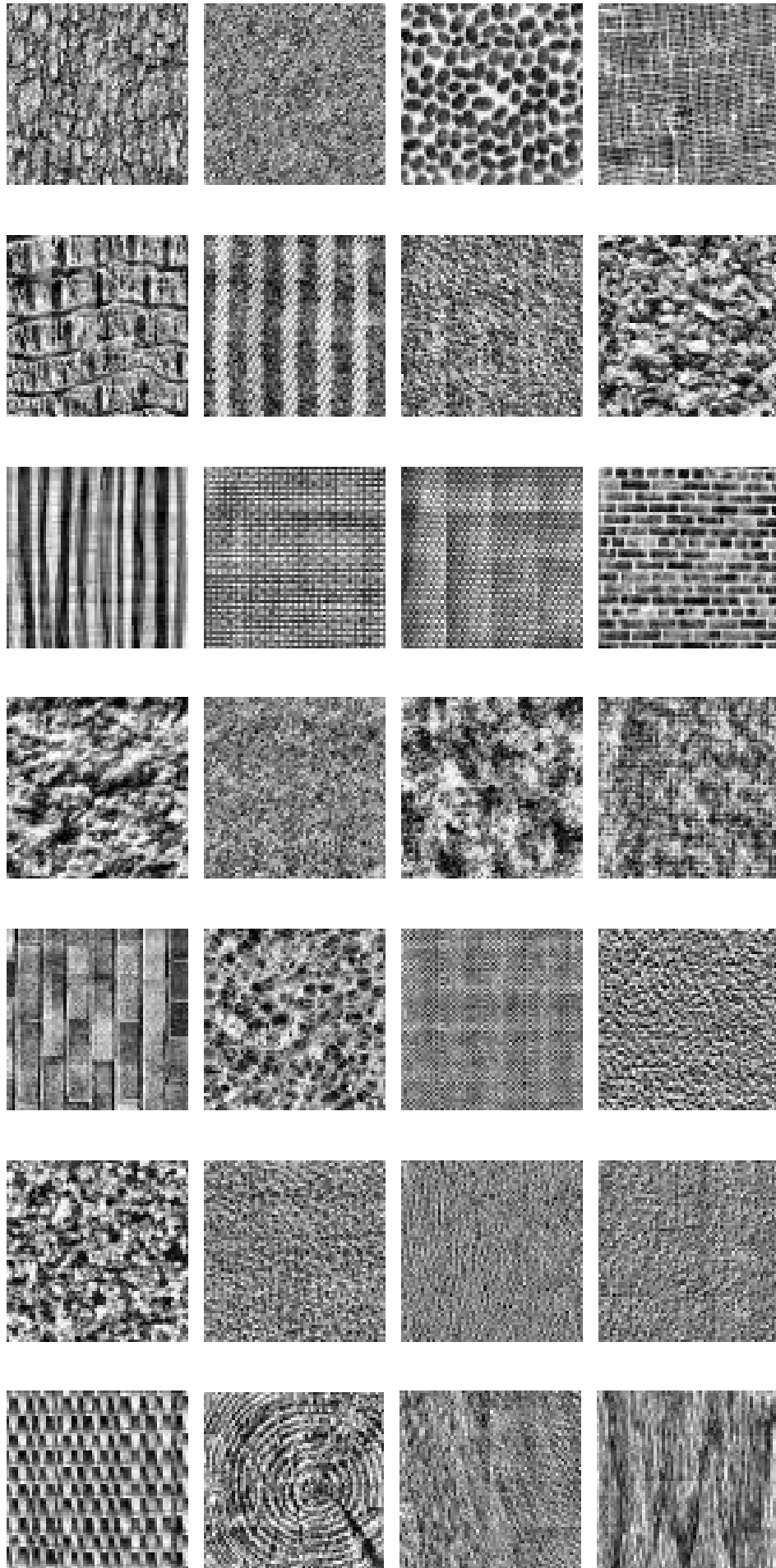


Figure 3.4 Sample images from the Brodatz album.

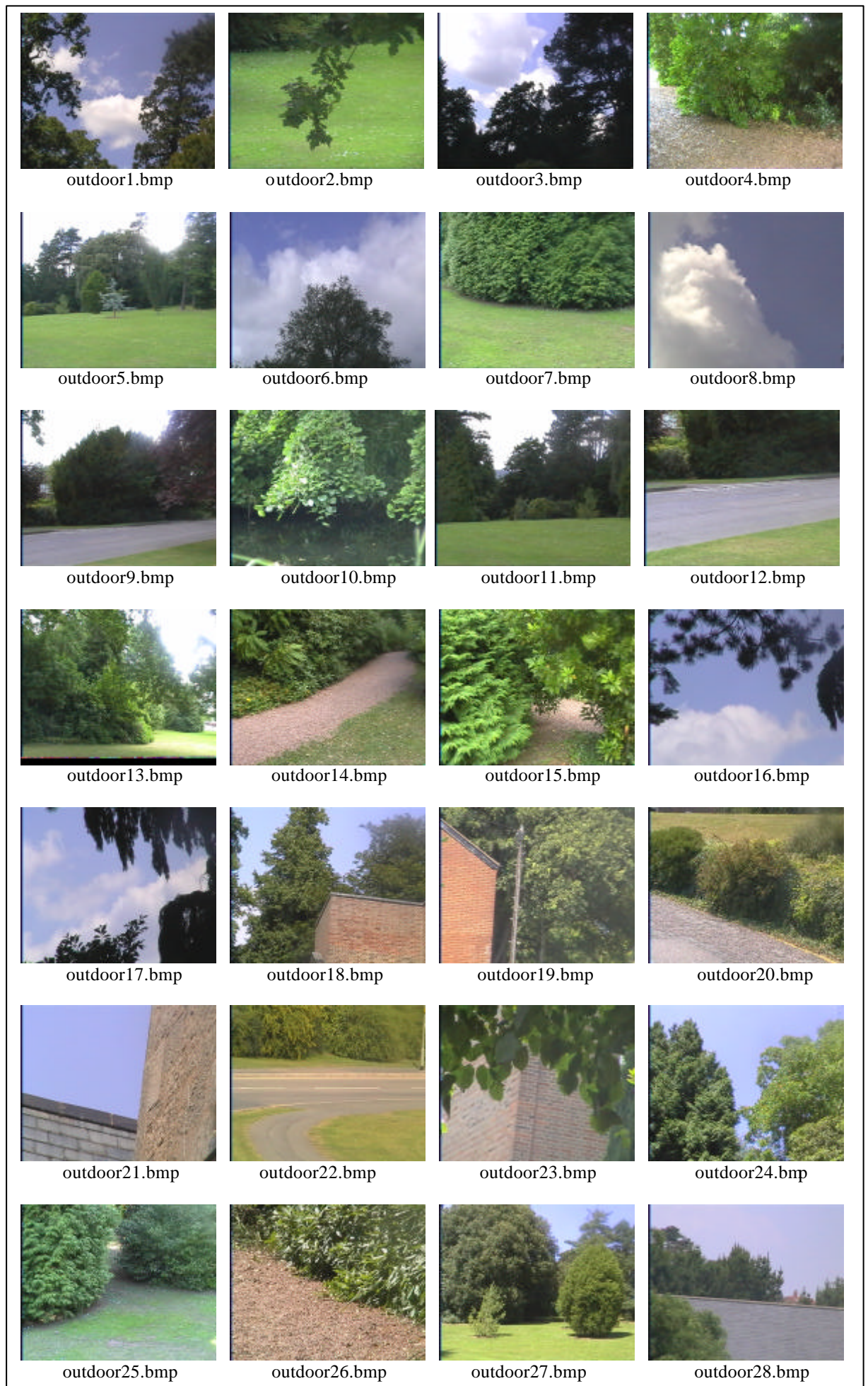


Figure 3.5 Sample images from the PANN Scene Analysis Benchmark.

The analysis is performed on all of the 448 images of the PANN database. The images contain nine different natural objects trees, grass, sky, clouds, bricks, pebbles, road, water, leaves and water. Water is not included for classification, as it is present only in four images, and therefore it is not possible to have enough samples for classification. As the images consist of different objects, they need to be segmented before feature extraction. Therefore, in the case of outdoor data, the images are first segmented and then texture features are extracted.

Class	FCM	Histogram Thresholding	Region Growing	Split and Merge
Trees	387	314	180	316
Grass	268	241	69	379
Sky	293	419	187	400
Clouds	247	303	176	315
Bricks	137	274	143	302
Pebbles	121	114	65	310
Road	152	196	59	215
Leaves	206	184	134	248
Total	1811	2045	1013	2485

Table 3.3 PANN benchmark data composition in terms of regions generated by different segmentation methods.

In Table 3.3 we show the total number of regions generated by different image segmentation methods. Each region yields one sample for our analysis. Split and merge generates the maximum number of samples and takes the longest to compute. Region growing generates the smallest number of samples. Vegetation categories, including trees, grass and leaves, have more samples than natural objects such as sky, clouds, bricks, pebbles and road.

3.2 Plan of work

We first present two flowcharts showing the methodology for analysing texture benchmarks and PANN database. The first flowchart shows the detailed methodology involved in the benchmark analysis. The second flowchart shows different steps in outdoor data analysis. We discuss the flowcharts and the process they include in sections 3.2.1 and 3.2.2.

3.2.1 Texture benchmark analysis

Figure 3.6 shows the various steps of analysis for texture benchmarks. For MeasTex benchmark, each image is divided into 16 equal parts. In the case of VisTex benchmark, images are first converted into Portable Greyscale format and then divided into four parts each. In the next step, texture features are extracted using five different texture extraction methods. The methods used are auto-correlation [178,199], co-occurrence matrices [86,176,178], edge frequency [178,199], Laws[178,126], and primitive run length [46,176,199]. After the feature extraction phase, the features extracted from each method are classified individually and also features from all the five methods are combined as a set and then classified. For classification, linear classifier and nearest neighbour classifiers are used. It is well known that nearest neighbour models compare well with neural networks and provide good classification rates on non-linear classification problems[189]. In this study, two different models of the nearest neighbour classifiers namely

Model1 and Model2 are used [195]. The detailed results for the benchmark analysis are presented in chapter 5.

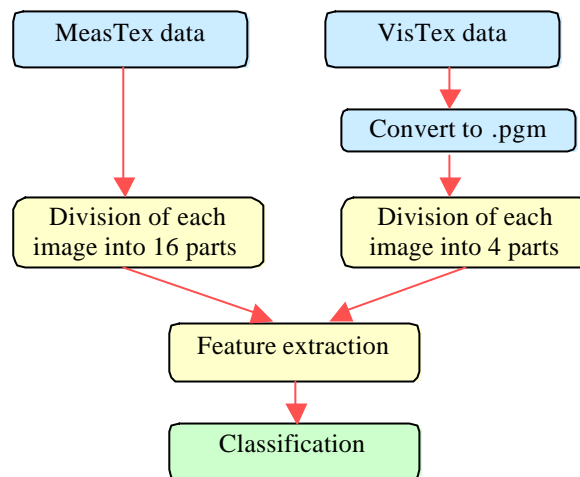


Figure 3.6 Flowchart for the methodology of benchmark analysis.

3.2.2 PANN database analysis

Fig 3.7 shows the flowchart for the analysis of the PANN database.

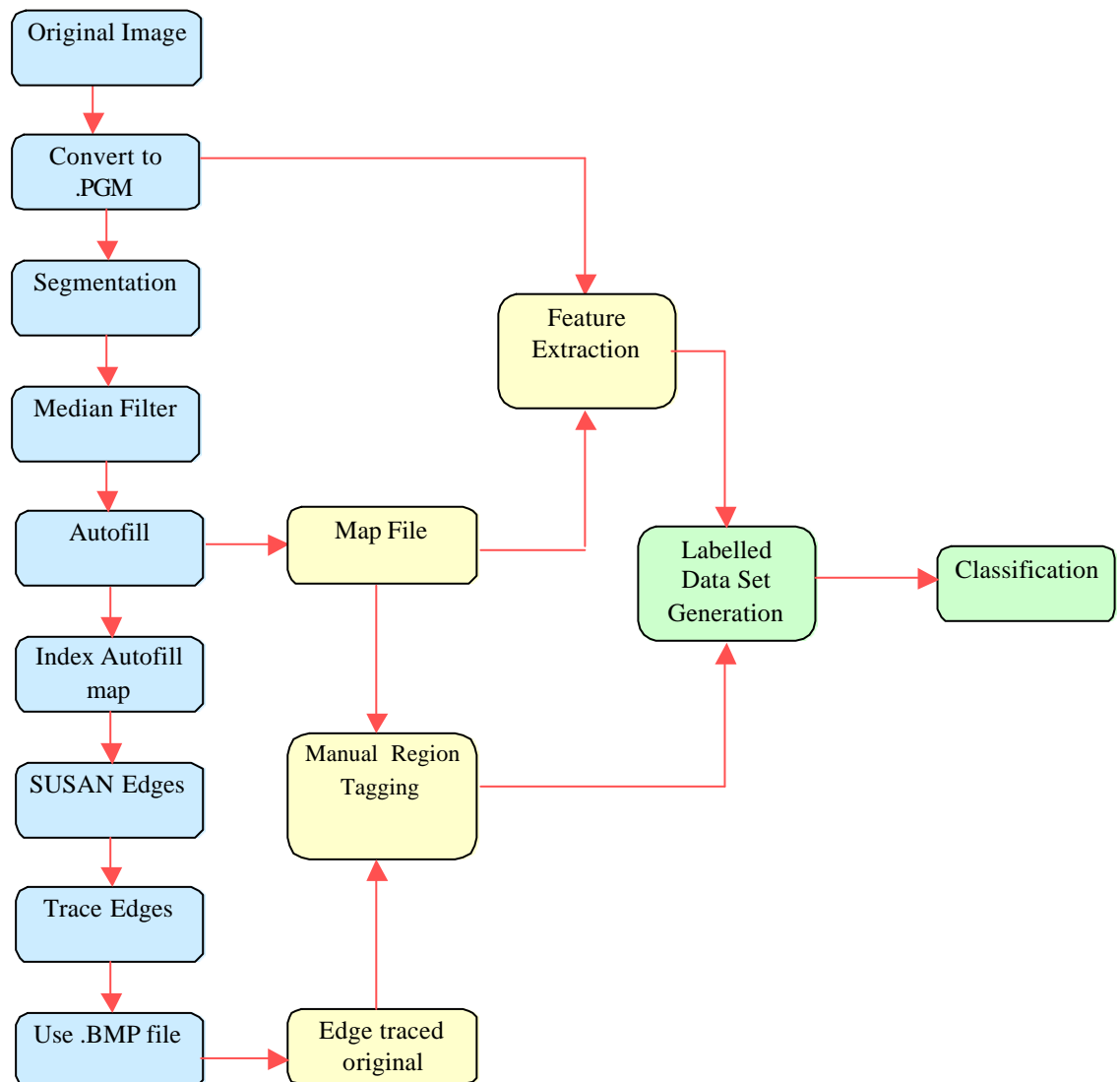


Figure 3.7 Flowchart showing various image processing steps for PANN benchmark analysis.

We explain below various steps shown in the flowchart. For the basics of image processing operations, please refer to Petrou [170] and Gose et al.[75].

Step 1. *Original images*

The entire set of original images from the Exeter PANN database is used for the analysis.

Step 2. *Convert to .PGM*

Using the convert command in Linux, each original colour image is converted to a 256 greyscale Portable Grey Map (.pgm) format of size 512x512 pixels. This format is a popular academic research image format and the choice for many tools and techniques used in PANN laboratory.

Step 3. *Enhancement*

When the images are converted to pgm format, some information is lost because of the conversion. For only those images that are of very poor quality, enhancement is performed as a pre-processing step before image segmentation. First, noise is removed and then spatial domain and frequency domain filters are used as needed to improve the image quality. For noise removal, a median filter is used in this study. The median filter enhances a given image by reducing noise. The grey level of each pixel is replaced by the neighbourhood median and the result is a smoother image without noise. In this study, a 3x3 pixel neighbourhood is used for median filtering. In order to perform median filtering to an image in the neighbourhood of a pixel, we first sort the values of the pixel and its neighbours, determine the median, and assign this value to the pixel. For example, in a 3x3 pixel neighbourhood, the median is the 5th largest value, in a 5x5 neighbourhood it is the 13th largest value, and so on. When several values in a neighbourhood are the same, all equal values have to be grouped. For example, suppose that a 3x3 neighbourhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus the principal function of median filtering is to force points with distinct intensities to be more like their neighbours, thus eliminating small regions that appear isolated in the area of the filter mask.

Bright images are smoothed using average filter and lowpass-butterworth filter, while dull images are sharpened using highpass-butterworth filters. Some images that appear too bright can be smoothed using a lowpass filter. In the case of a lowpass filter, the grey level of each pixel is replaced by the average in a neighbourhood and the result is a smooth image. In this study a 3x3 pixel neighbourhood is used for lowpass filtering. Image blurring is also done by the reduction of high-frequency components from the image. For this purpose a lowpass-butterworth filter is used in this study.

Some dull images are sharpened using a highpass filter. This filter is used to improve the edge information and enhance object contrast with the background. In our study a 3x3 pixel neighbourhood is used for highpass filtering. Image sharpening is also done by increasing the magnitude of the high frequency components relative to the low-frequency components. For this purpose a highpass-butterworth filter is used in this study.

The image enhancement step improves the visibility of the images and makes them better suitable for further analysis, such as image segmentation. Image enhancement is performed on 32 images from the PANN database. After enhancement, the images are ready for the segmentation process, which is described below.

Step 4. Segmentation

Division of an image into its constituent objects is called segmentation. An image can be divided into a number of objects depending on the objects of interest in any problem i.e. segmentation is stopped when the objects of interest. Segmentation carries a risk of generating wrong object boundaries. Splitting the image into too many objects, which is termed as over-segmentation, generate many small regions which is problematic for good texture analysis and time consuming to label when generating ground truth data. Under-segmentation is a much more serious problem, as in that case different objects can be merged into one cluster [118].

We find that it is better to risk over segmentation than under segmentation because in the first case all of the pixels belonging to a region belong to the same class. On the other hand, in the case of under-segmentation different regions are merged together, so one region can contain pixels belonging to more than one class.

In our study, we have used four commonly used techniques for image segmentation. The techniques used are: fuzzy c-means clustering[206], histogram based thresholding, region growing and split and merge[74]. All of the segmentation techniques are performed on the greyscale (.pgm) version of the original images. After segmentation, the results produced by each method are processed individually for feature extraction and classification phase. The segmentation techniques are described below. Their detailed algorithms appear in the next chapter.

a. Fuzzy c-means clustering based segmentation

Fuzzy c-means clustering describes an image in terms of fuzzy classes. This means that each pixel in an image is assigned a membership to class clusters using a fuzzy function. Each class can consist of many disjoint segments, and depending on the nature of the image, this number can be a large figure. In this study, a version of the fuzzy c-means clustering algorithm implemented by the Robotics and Image Analysis Laboratory at the University of West Florida is used. For this we use the following parameters: number of classes is set to 5, the fuzzy factor used is 2.0, termination threshold is 0.5 and the maximum number of iterations used is 15. We have used a maximum of 5 clusters for each image as for a given image we always find the total number of classes present to be five or less.

b. Histogram thresholding based image segmentation

This method segments the image into a number of object regions based on the number of peaks in the image histogram. Histogram thresholding performs well only if different objects in the image have considerably different grey levels. In this method, the segmentation is automatic and there is no need to specify in advance the number of objects in which the image is to be segmented. In case of images with a large number of small peaks, histogram smoothing is required for the algorithm to perform well. In our case, as the number of objects is never more

than five in any image and the algorithm performs well without any form of pre-processing of the histogram.

c. Region growing based image segmentation

Region growing groups pixels into larger regions by the process of pixel aggregation. The process starts with the first pixel and merges the neighbouring pixels to it based on whether a certain condition is satisfied. In this case, the merging is based upon the grey level difference between the pixel and its neighbours. This threshold value determines the quality of segmentation. If this threshold is set to a small value, the result is over-segmentation because not many neighbouring pixels satisfy the merge criterion. On the other hand if the threshold is set to a large value, it results in under-segmentation and two or more regions are merged into a single region. Again, it is better to over-segment rather than under-segment. Various experiments were performed to find the best value of the threshold. In this study, this threshold is set to 30.

d. Split and merge segmentation

Split and merge segmentation initially subdivides an image into a set of arbitrary, disjoint regions and then merges them iteratively to satisfy a given criterion. At each step, the image is split into four disjoint quadrants and then each quadrant is again split until all of the pixels belonging to that quadrant are homogenous. We need to however specify the maximum splitting possible. The node size corresponds to the minimum size that an image quadrant can have after recursive splitting. When the splitting is complete, the neighbouring regions are merged until a certain condition is satisfied. For each quadrant obtained at the end of the split process, its average grey level is calculated. This is compared to the grey level average of neighbouring quadrants, and if the different between them is less than the merge factor, which is user specified, then the quadrants are merged together. For this a node size and a merge factor needs to be specified. In this study we have used a node size of 4 and a merge factor of 10. A set of preliminary experiments was performed to select the optimal values of the node size and the merge size.

Step 5. Median filtering

After the segmentation step, because of the complex detail in the outdoor scenes, image segmentation results in several very small regions, some of which have only a single pixel. Such regions are too small to be manually labelled when generating ground truth data, and therefore some form of filtering is required to eliminate regions that are less than a certain size. A filtering method is needed that removes these tiny regions while preserving image structure. Median filter is used to eliminate small regions that appear isolated in the area of the filter mask. A 5x5 pixel mask is used for median filtering in this case. In Figure 3.8(a) we show an original image and in Figure 3.8(b-d) median filtered images using different mask sizes. We finally settles on using a mask size of 7x7 pixels.

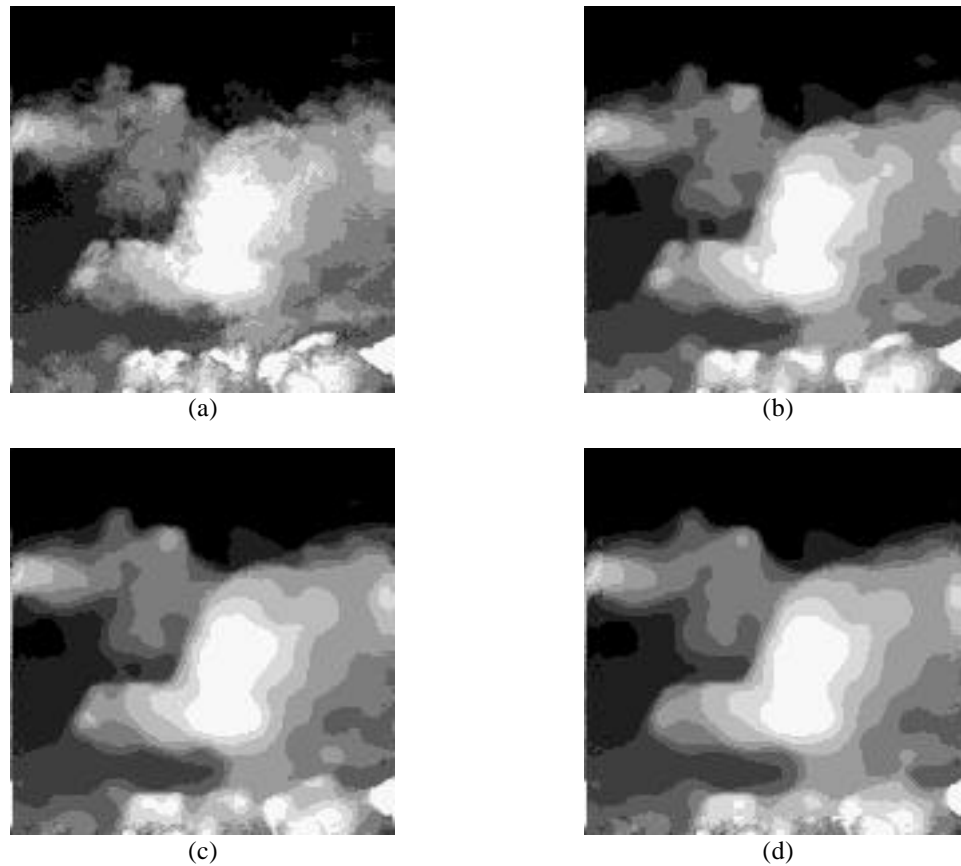


Figure 3.8 (a) Original segmented image; (b) median filtered output with 3x3 pixel mask; (c) median filtered output with 5x5 pixel mask; (d) median filtered output with 5x5 pixel mask.

Step 6. Autofill

The next aim is to construct a map file from the median filtered image that contains all regional information we need for textural feature extraction. The construction of a map file requires some processing of the median filtered image to generate the desired result. To extract texture features from a region, the main question is “what minimum sized area constitutes a valid region?” The quest for an answer to this question led us to the conclusion that the area should be at least 500 pixels in order to extract meaningful textural features. This may sound like quite a large figure, but it is not when the image size totals 262144 (512x512) pixels. Considering the median filtered images, we notice that even after applying a 7x7 mask to the segmented output we still have regions containing less than 500 pixels. Additional processing of the median filtered image is required that considers each region in the image. If a given region consists of less than 500 pixels, then it is merged within the region in which it is enclosed.

For the map file to be useful, it must uniquely identify each region. This will allow us to trace back individual regions later in the project to help us to analyse case by case performance is needed. The regions should be labelled 1 to N , where N is the total number of regions in the map file. In theory, this is a relatively simple procedure, which involves scanning the median filtered image and labelling each homogeneous region with a unique number. In practice however, this is far from a trivial task. We eventually decided on designing a recursive technique to both eliminate the small regions and assign each to a unique identifier. The autofill algorithm eliminates all regions with less than 500 pixels and creates the map file by assigning a unique

value to every region. This map file is used to identify different regions for manual tagging and also for feature extraction.

Step 7. *SUSAN*

The output of the autofill procedure produces a map file from the median filtered image that contains all of the regions needed for textural feature extraction. This map file is used not only for feature extraction. It is also used as a guide for generating the outlines of regions used when manually tagging the images to produce ground truth data. The borders of these regions are detected and traced over the original colour image. These edge-traced images are used when generating ground truth data by simply clicking within a bordered region with the mouse and labelling the region as of a particular class. The technique used for this is called SUSAN.

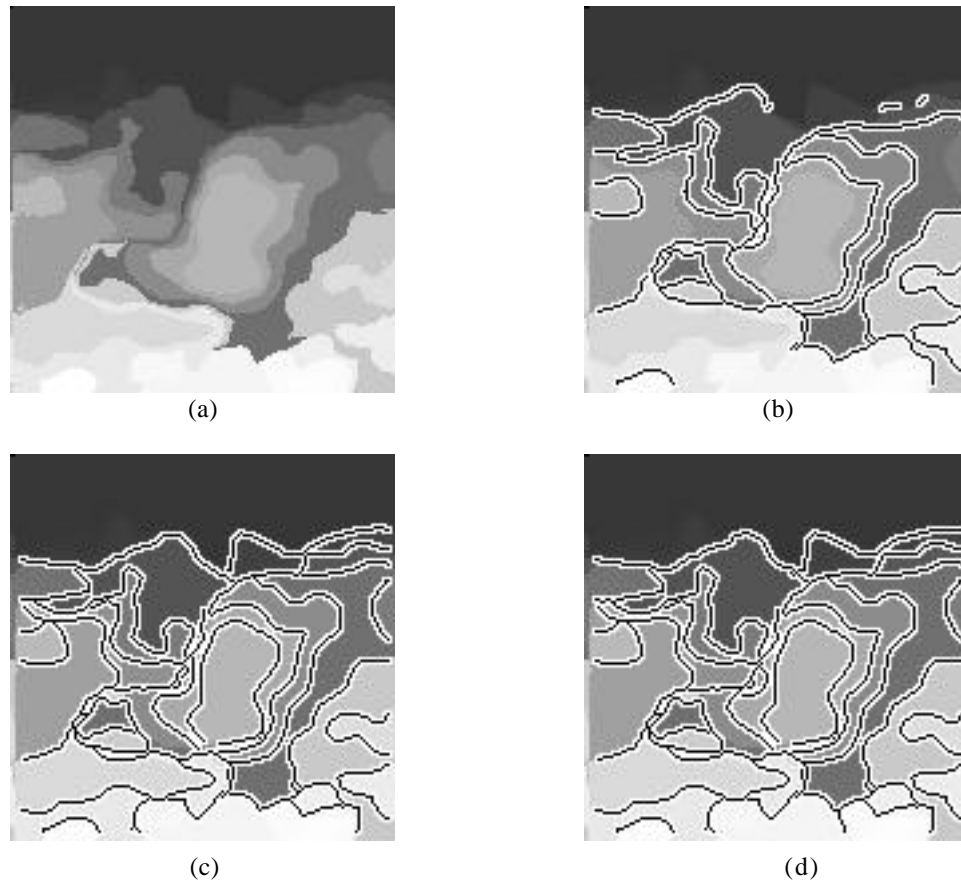


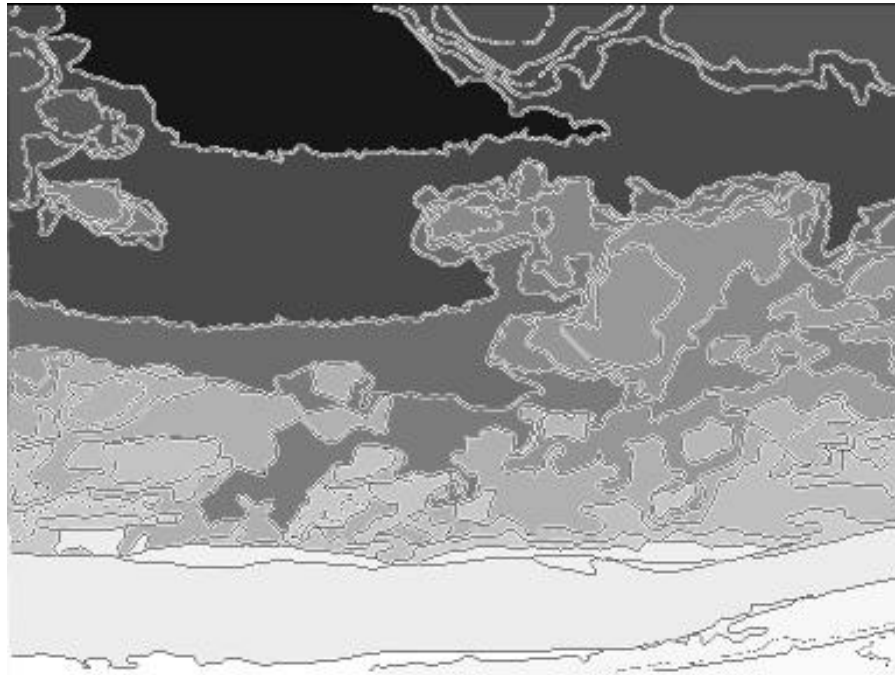
Figure 3.9 The effect of brightness parameter on detecting edges in SUSAN; (a) Equalised map file; (b) output with brightness parameter $b=20$; (c) output with $b=10$; (d) output with $b=5$.

Although there are many edge detection techniques available, SUSAN is chosen for this study because of two reasons. First, SUSAN is very flexible as it allows a variety of parameter changes, which means that it can be customised to suit several different applications. Second, the detected edges with SUSAN are black edge pixels surrounded by a white border. This means that once the edges are found, it is a very simple process to retrieve these edges. In addition to these points, SUSAN is a very fast algorithm and it has been used successfully with other research at PANN research laboratory. Figures 3.9(a-d) show the example output of SUSAN using different brightness parameter settings. It is clear that Figure 3.9(d) detects more edges

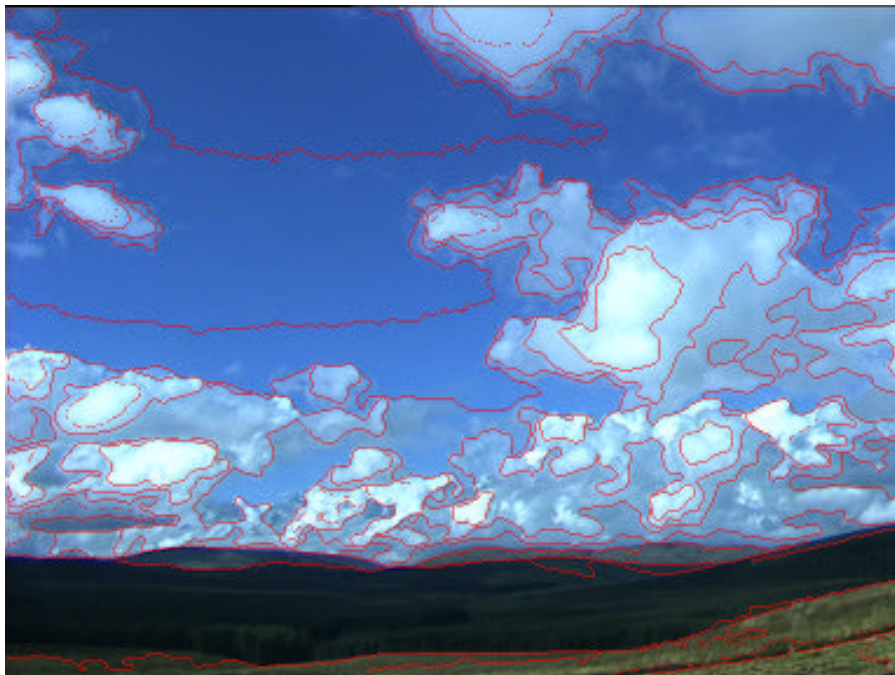
than other examples. This example uses the value 5 for the brightness parameter and this was the default value we used for all images in the project data set.

Step 8. Trace edges

With the edges of the map file detected successfully, it is a relatively simple process to trace these edges over the original. A simple procedure was written to scan the SUSAN edge-detected image for edges (black pixels surrounded by a white border). When edges are found we draw a red pixel at the corresponding location in the original colour image. The following example shows a edge-detected map image and its corresponding traced image (Figure 3.10(a) and (b)).



(a)



(b)

Figure 3.10 (a) SUSAN edge detected image; (b) the same edges overlaid on the colour original.

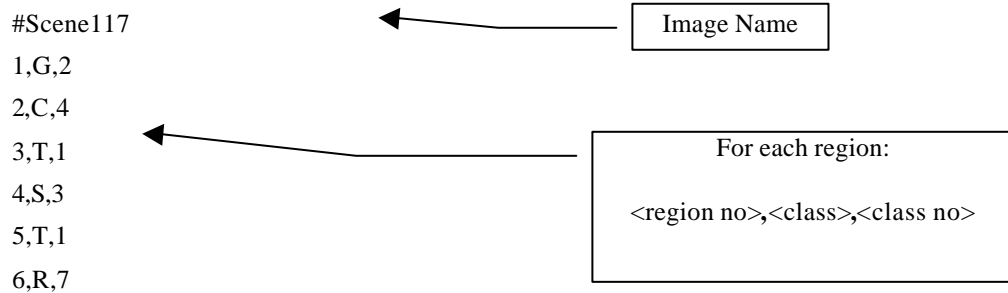
Step 9. Use .BMP file

The edge traced images in the .bmp format are used. This is required for manual region tagging to have the coloured images for adequate ground truth data generation.

Step 10. Manual region tagging

One of the greatest difficulties in scene analysis research is the generation of ground truth data. It is essential that ground truth data is available because it forms the basis of the training set(s) and provides a means of judging the classification performance when testing. To make the generation of this data as simple as possible, a software is designed in PANN lab to allow particular regions of an image to be labelled as one of the available classes, effectively generating our own ground truth. This manual tagging software consists of an image viewer that is used to display the edge-traced image and a point and click interface for manually tagging regions of the image. The idea is to open the edge traced images in sequence and for each image manually tag regions as of a known class. The traced edges from the SUSAN image are used as a guide to visualise the boundaries between regions. When an edge-traced image is displayed, its corresponding map file is automatically opened in the background. This map file is used as a reference for the pixels that fall within a region when the user clicks on a location in the image viewer. When the user clicks inside a region on the edge-traced image and chooses to label it as a particular class, the (x, y) coordinates of the mouse pointer are used to find the value of the corresponding location in the map file. The value of this location in the map file represents a unique region identifier, and all pixels sharing this identifier in the map file are marked as tagged. All of these corresponding pixels in the traced image are then updated to show that they have been tagged. The image viewer changes these pixels with a class-specific colour so that the user can determine which regions are left to be tagged.

The motivation for performing this real-time, interactive image tagging is to generate our ground truth data as easily as possible. This ground truth data is appended to a file each time the user closes the current image. The generated file contains the ground truth information of each region in each image and is formatted as follows:



```
#Scene117
1,G,2
2,C,4
3,T,1
4,S,3
5,T,1
6,R,7
```

Image Name

For each region:
<region no>,<class>,<class no>

Figure 3.11 shows the file selection procedure used for tagging the images. This shows the interface for selecting the image to tag. When double clicked, each image file opens in the window shown in Figure 3.12. The example shows a partially tagged image. The different colours represent different class selections when tagging an image from the PANN database.

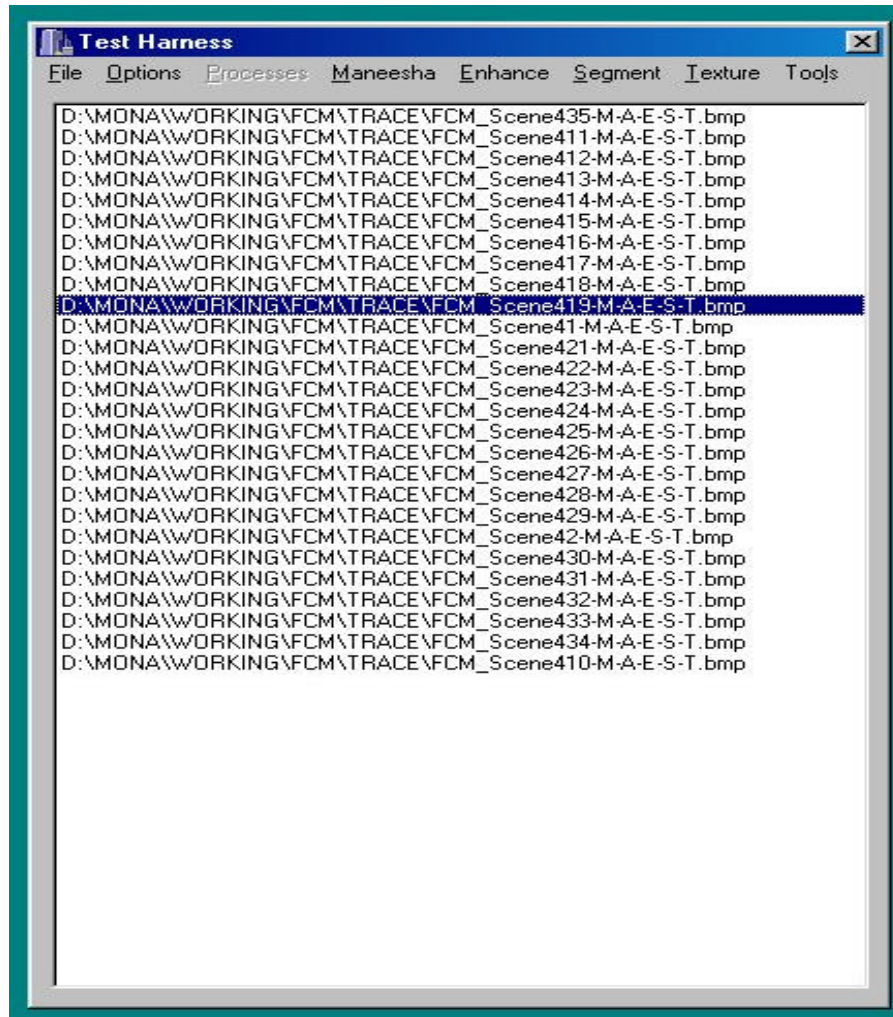


Figure 3.11 File selection for tagging images.

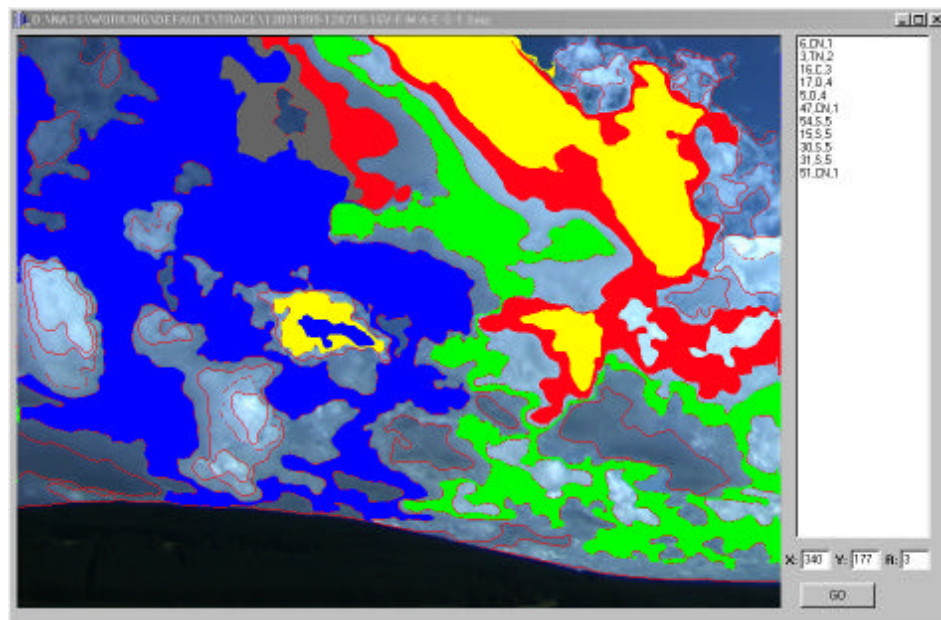


Figure 3.12 Region tagging for generating ground truth data.

Step 11. *Texture feature extraction*

Once we have labelled the regions, texture feature data needs to be extracted from them. This feature data will be used during classification to discriminate between different objects (classes)

from one another. Five different textural feature extraction techniques are used in this study. The five texture feature extraction methods used are: *auto-correlation*, *co-occurrence matrices*, *edge frequency*, *Laws masks* and *primitive run length*. For the results from each segmentation process, texture features are extracted individually. For each of the four segmentation methods, fuzzy c-means clustering, histogram thresholding, region growing and split and merge, features are extracted using five texture methods, so at the end we get 20 separate feature files (4 segmentation methods multiplied by 5 texture extraction methods). Each original feature extraction method was run as a separate batch process to generate texture features from each region in each image. For each feature extraction method to work, it requires both the original greyscale copy of the original image and the map file. The map file contains information about each region to extract features from, and the greyscale copy contains the image structure from which texture features are extracted. For each region in the map file, a separate row in the feature file is generated. Each row contains the following tab delimited information shown in Table 3.4.

<image1 name>	<region 1>	<texture features f_1, \dots, f_n for region 1>
"	<region 2>	<texture features f_1, \dots, f_n for region 2>
"	"	"
"	"	"
"	"	"
<image2 name>	<region 1>	<texture features f_1, \dots, f_n for region 1>
"	<region 2>	<texture features f_1, \dots, f_n for region 2>
"	"	"
"	"	"
"	"	"
<image3 name>	<region 1>	<texture features f_1, \dots, f_n for region 1>
"	<region 2>	<texture features f_1, \dots, f_n for region 2>
"	"	"
"	"	"

Table 3.4 Structure of the generated feature file.

In the case of autocorrelation features, a total of 99 features were extracted. For co-occurrence matrix features we used Haralick's 14 texture features for our texture benchmark analysis. In case of PANN scene analysis benchmark analysis, we use an additional set of six features namely homogeneity, entropy, mean grey level, standard deviation of grey level, kurtosis and skewness. So we get a feature set of 20 features. For edge frequency method, a total of 50 features have been used. For Law's features, a total of 25 masks have been used giving a total of 125 features. Finally, for primitive length we have used a total of 5 features with a primitive run length of 30 in all directions. All these features have been computed on all regions produced by the segmentation algorithm. The only exception to this lies with a small amount of difference with the co-occurrence matrix features where the algorithm did not compute reasonable features from very small regions which means that we have slightly less samples in some of the cases with this feature set for the scene analysis database.

Step 12. Labelled data set generation

At this stage in the methodology, we have the ground truth data for each region and texture features for each region, both in separate files. A small procedure was written to take both files and generate a new, labelled data set. This procedure scans each row of the texture feature file and stores the <image name> and <region no>. It then scans each line of the ground truth data until the <image name> matches an image name entry. Once the feature entry is matched to its corresponding ground truth entry, each feature entry is written to a new file with its ground truth tag added on the end. Each row of this new re-labelled file contains the tab delimited information as shown in Table 3.5. For each of the five feature extraction methods, a file in the above format is available after ‘true feature labelling’. It is this class-labelled texture feature file that we use for classification. A file in the above format is available for each of the four segmentation methods for all the five texture feature extraction processes.

<image1 name>	<region1>	<texture features f_1, \dots, f_n for region 1>	<image1, region1, class n^0 >
.	<region2>	<texture features f_1, \dots, f_n for region 2>	<image1, region2, class n^0 >
.	.	.	<image1, region3, class n^0 >
.	.	.	.
.	.	.	.
<image2 name>	<region1>	<texture features f_1, \dots, f_n for region 1>	<image2, region1, class n^0 >
.	<region2>	<texture features f_1, \dots, f_n for region 2>	<image2, region2, class n^0 >
.	.	.	<image2, region3, class n^0 >
.	.	.	.
.	.	.	.
<image n name>	<region1>	<texture features f_1, \dots, f_n for region 1>	<image n, region1, class n^0 >
.	<region2>	<texture features f_1, \dots, f_n for region 2>	<image n, region2, class n^0 >
.	.	.	<image n, region3, class n^0 >

Table 3.5 Structure of the generated feature file with labelling.

Step 13. Classification

After the features are extracted, they are subjected to a classification stage. There are two important considerations. First, how do we present the data to the classifier? Random data split into training and test classes can result in either over-optimistic or over-pessimistic results. Hence for the purposes of our study we have used leave-one-out cross validation so that all available data is tested. Second, we need to consider the classifier to be used. For our study, we have used the linear classifier and the nearest neighbour classifier. The input to the classifier is the feature file that has been labelled and the output of the classifier is a confusion matrix showing how the samples are classified as well as an average recognition rate.

In this chapter we have presented the methodology for our study giving details of the various data we will be using. The chapter has not, however, detailed the algorithms of different processes shown in the flowcharts. For example, we have not discussed any image segmentation or texture analysis algorithms. This forms the topic of detail in our next chapter.

Chapter 4

Algorithms for image analysis and pattern recognition

This chapter describes image processing and pattern recognition algorithms used in this study in algorithmic level of detail. These algorithms include methods used for image enhancement, image segmentation, texture feature extraction, and classification. These brief details are important to highlight the exact nature of tools used for processing images. We have not used image enhancement for all images. Only 32 images in the PANN database have been enhanced as it was considered worthwhile to improve their quality for feature extraction. For the other 416 images, no preprocessing is performed. For segmentation and texture extraction, we detail the algorithms used. Their actual implementation is sometimes more complex than the detail presented here. The same applies to feature extraction methods such as principal components analysis. Classifier algorithms have also been provided except for in-depth details of the linear discriminant analysis that is widely available as a part of statistical packages.

4.1 Algorithms for image enhancement

In some cases before any analysis can be performed on natural images, it is necessary to pre-process them to make them better suited for analysis. Enhancement is a technique that emphasises salient features of the original image such as object edges and removes background noise. The main aim of image enhancement is to improve the visibility of fine patterns and object details. In our analysis we have tried our best not to use image enhancement since it is not clear how this may affect the quality of the texture features extracted. However, we have used it for a small number of images in the scene analysis database where the images are of extremely poor quality and need enhancement before decent segmentation or texture analysis can be performed.

We have used both smoothing and sharpening filters as described by Gonzalez and Woods[74]. In the frequency domain, these are referred to as lowpass and highpass filters respectively. We briefly detail these filters in the spatial and frequency domain.

4.1.1 Filters in spatial domain

The term spatial domain refers to the aggregate of pixels composing the image and spatial domain methods are the procedures that operate directly on these pixels. Image enhancement functions in the spatial domain are expressed as: $g(x, y) = T(f(x, y))$, where $f(x, y)$ is the input image, $g(x, y)$ is the processed image, and T is the image enhancement operator. It is the enhancement operator that determines how images are pre-processed.

The use of spatial masks for image enhancement is usually called spatial filtering and the masks used are called spatial filters. A square neighbourhood is defined and the centre of this area is moved from pixel to pixel. At each location, the mask is convolved with the image pixels. The

original pixel value is replaced by the convolved value to generate the enhanced image. This is shown in Figure 4.1.

a_1	a_2	a_3	w_1	w_2	w_3	$a_1 * w_1$	$a_2 * w_2$	$a_3 * w_3$
a_4	a_5	a_6	w_4	w_5	w_6	$a_4 * w_4$	$a_5 * w_5$	$a_6 * w_6$
a_7	a_8	a_9	w_7	w_8	w_9	$a_7 * w_7$	$a_8 * w_8$	$a_9 * w_9$
(a)	(b)	(c)						

Figure 4.1 (a) The pixel neighbourhood; (b) mask; (c) convolved values that are averaged.

In Figure 4.1, the centre pixel value a_5 is replaced by $\frac{1}{9} \sum_{i=1}^9 a_i \cdot w_i$.

The masks can be so designed to attenuate the low or high frequency components of the resultant image. Smoothing filters cause image regions to blur and remove noise. Sharpening filters enhance sharp edges and object boundaries. These filters are described below.

Smoothing Filters

These filters are used for blurring and for noise reduction. These filters are also used for bridging small gaps in lines or curves prior to image segmentation. Smoothing filters used in this study are averaging and median filters. In the case of averaging filters, the centre pixel value is replaced by the average of its neighbourhood. In the case of median filter, it is replaced by the median of the neighbourhood.

Sharpening filters

These filters are used to highlight fine details in an image or to enhance the details that have been blurred either by mistake or while image acquisition. We have used a highpass filter in this study. Highpass filter is used to enhance the edge details in the image. The mask has a positive coefficient at its centre, $w_5 = 8$ and all other $w_i = -1$. The results are scaled to have a range of [0,255].

4.1.2 Filters in frequency domain

In case of enhancement in the frequency domain, we first compute the Fourier transform of the image to be enhanced. The transform is then multiplied with a transfer function that determines the quality of enhancement. The inverse Fourier Transform is then performed on the resultant image to produce the enhanced image. Frequency domain filters used in this study include lowpass Butterworth filter and highpass Butterworth filter. These are described below.

Lowpass Butterworth filter

This is a smoothing filter. The transfer function of the Butterworth lowpass filter, of order n and with the cutoff frequency locus at a distance D_0 from the origin, is defined by the relation:

$$H(u, v) = \frac{1}{1 + [\sqrt{2} - 1][D(u, v)/D_0]^{2n}} \quad \dots \text{Eqn (1)}$$

where $D(u, v) = \sqrt{u^2 + v^2}$

Highpass Butterworth filter

This is a sharpening filter. The transfer function of the Butterworth highpass filter, of order n and with the cutoff frequency locus at a distance D_0 from the origin, is defined by the relation:

$$H(u, v) = \frac{1}{1 + [\sqrt{2} - 1][D_0]/D(u, v)]^{2n}} \quad \dots \text{Eqn (2)}$$

4.2 Algorithms for image segmentation

Image segmentation subdivides an image into its constituent parts or objects. It entails the separation of the image into regions of similar attribute or homogeneity with respect to a given characteristic. Autonomous segmentation is one of the most difficult tasks in image processing. The aim of this section is to describe the different techniques used and state why they were deemed applicable within the study. Four commonly used techniques for autonomous image segmentation are applied in this study and their results are compared. Performance evaluation of segmentation methods is a tough task as different parameter settings can affect the results significantly. The problem of over-segmentation and under-segmentation is also quite crucial in this context. All of the segmentation methods used in this study are selected so that there are very few user-defined parameters required. We have used four most commonly used traditional methods for image segmentation, namely fuzzy c-means clustering based segmentation, region split and merge segmentation, region growing, and histogram thresholding based segmentation. These are described in the following sections. The FCM algorithm has been taken from the implementation of Bezdek's code and the split and merge algorithm has been taken from Gonzalez and Woods[74]. The two other pieces of code were written by myself.

4.2.1 Fuzzy c-means clustering based segmentation

Fuzzy c-means clustering is a process designed to assign each sample to a cluster based on cluster membership. The segmentation of the image into different regions can be thought of as the assignment of pixels to different clusters. The algorithm is based on iterative minimization of the following function:

$$J(U, V) = \sum_{i=1}^c \sum_{k=1}^n u_{ik}^m |x_k - v_i|^2 \quad \dots \text{Eqn. (3)}$$

where,

- x_1, \dots, x_n are n data sample vectors;
- $V = \{v_1, \dots, v_c\}$ are cluster centres;
- $U = [u_{ik}]$ is a $c \times n$ matrix, where u_{ik} is the i^{th} membership value of the k^{th} input sample x_k , and the membership values satisfy the following conditions

$$0 \leq u_{ik} \leq 1 \quad i = 1, 2, \dots, c; \quad k = 1, 2, \dots, n \quad \dots \text{Eqn. (4)}$$

$$\sum_{i=1}^c u_{ik} = 1 \quad k = 1, 2, \dots, n \quad \dots \text{Eqn. (5)}$$

$$0 < \sum_{k=1}^n u_{ik} < n \quad i = 1, 2, \dots, c \quad \dots \text{Eqn. (6)}$$

- $m \in [1, \infty]$ is an exponent weight factor.

The objective function is the sum of the squared Euclidean distances between each input sample and its corresponding cluster centre, with the distances weighted by the fuzzy memberships. The algorithm is iterative and makes use of the following equations:

$$v_i = \frac{1}{\sum_{k=1}^n u_{ik}^m} \sum_{k=1}^n u_{ik}^m x_{ik} \quad i = 1, 2, \dots, c \quad \dots \text{Eqn. (7)}$$

$$u_{ik} = \frac{\left[\frac{1}{|x_k - v_i|^2} \right]^{1/(m-1)}}{\sum_{j=1}^c \left[\frac{1}{|x_k - v_j|^2} \right]^{1/(m-1)}} \quad i = 1, 2, \dots, c; \quad k = 1, 2, \dots, n \quad \dots \text{Eqn. (8)}$$

For calculation of a cluster centre, all input samples are considered and the contributions of the samples are weighted by the membership values. For each sample, its membership value in each class depends on its distance to the corresponding cluster centre. The weight factor m reduces the influence of small membership values. The larger the value of m , the smaller the influence of samples with small membership values. The fuzzy c-means clustering procedure consists of the following steps.

Algorithm FCM

- 1 Initialise $U^{(0)}$ randomly or based on an approximation; initialise $V^{(0)}$ and calculate $U^{(0)}$. Set the iteration counter $\infty = 1$. Select the number of class centres c and choose the exponent weight m .
- 2 Compute the cluster centres $V^{(\infty)}$ according to Eqn. 7.
- 3 Update the membership values $U^{(\infty)}$ according to Eqn 8.
- 4 Stop the iteration if $\max |u_{ik}^{(\infty)} - u_{ik}^{(\infty-1)}| \leq \epsilon$
else let $\infty = \infty + 1$ and go to Step 2, where ϵ is the pre-specified small number representing the smallest acceptable change in U .

Figure 4.2(b) shows the FCM segmented image corresponding to the original in Figure 4.2(a).

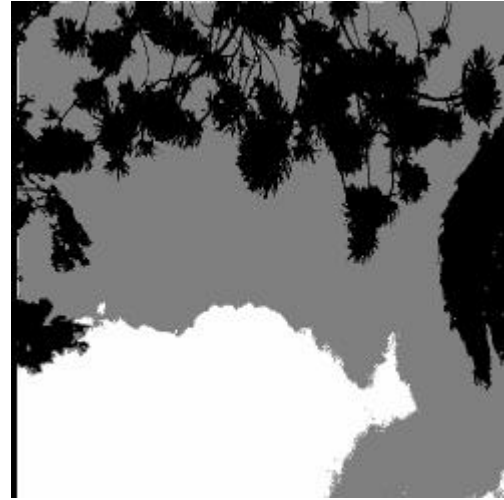


Figure 4.2 (a) Original image.

Figure 4.2(b) FCM segmented image.

4.2.2 Split and merge segmentation

Split and merge is an image segmentation procedure that initially subdivides an image into a set of arbitrary, disjoint regions and then merges and/or splits the regions in an attempt to satisfy the conditions below [110]. Let R represent the entire image region. Segmentation can be viewed as a process that partitions R regions into n subregions, R_1, R_2, \dots, R_n , such that:

- 1 $\bigcup_{i=1}^n R_i = R$,
- 2 R_i is a connected region, $i=1,2,\dots,n$,
- 3 $R_i \cap R_j = \Phi$ for all i and j , $i \neq j$,
- 4 $P(R_i) = \text{TRUE}$ for $i=1,2,\dots,n$, and
- 5 $P(R_i \cup R_j) = \text{FALSE}$ for $i \neq j$,

where $P(R_i)$ is a logical predicate over the points in set R_i and Φ is the null set. Condition 1 indicates that the segmentation must be complete, that is, every pixel must be in a region. The second condition requires that points in a region must be connected. Condition 3 requires that the regions must be disjoint. Condition 4 deals with the properties that must be satisfied by the pixels in a segmented region. For example $P(R_i) = \text{TRUE}$ if all pixels in R_i have the same intensity. Finally, condition 5 requires that regions R_i and R_j are different in the sense of predicate P .

Algorithm Split and Merge

- 1 Split into four disjoint quadrants any region R_i where $P(R_i) = \text{FALSE}$;
- 2 Merge any adjacent regions R_i and R_k for which $P(R_i \cup R_k) = \text{TRUE}$; and
- 3 Stop when no further merging or splitting is possible.

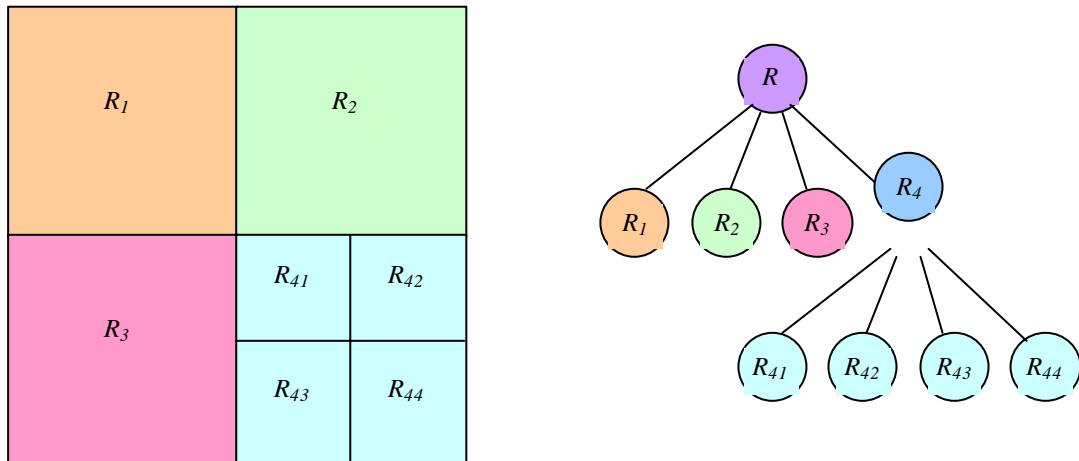


Figure 4.3 The partitioned image and its corresponding quadtree representation.

The split and merge algorithm iteratively works towards satisfying these constraints as follows. Let R represent the entire image region. Select a predicate P as above. For a square image, divide it successively into smaller and smaller quadrant regions so that, for any region R_i , $P(R_i) = \text{TRUE}$. That is, if $P(R) = \text{FALSE}$, divide the image into quadrants. If P is FALSE for any quadrant, subdivide that quadrant into sub-quadrants, and so on. This splitting technique has

a convenient representation in the form of a so-called *quadtree* representation (that is, a tree in which each node has exactly four descendants) as illustrated in Figure 4.3. The root of the tree corresponds to the entire image and each node corresponds to a subdivision. The image is now a partition of adjacent regions with identical properties. The algorithm next merges regions but only those whose combined pixels satisfy the predicate P above. The region merge and split algorithm may be summarised as follows.

Figure 4.5 illustrates the split and merge algorithm. The image consists of a single object and background. For simplicity, both the object and background have constant grey levels and $P(R_i) = \text{TRUE}$ if all pixels in R_i have the same intensity. Then, for the entire image region R , $P(R) = \text{FALSE}$, so the image is split as shown in Figure 4.4(a). In the next step, only the top left region satisfies the predicate so it is not changed, while the other three quadrants are split into sub-quadrants, as shown in Figure 4.4(b). At this point several regions can be merged, with the exception of the two sub-quadrants that include the lower part of the object; these do not satisfy the predicate and must be split further. Figure 4.4(c) shows the results of the split and merge operation. At this point all regions satisfy P , and merging the appropriate regions from the last split operation yields the final, segmented result shown in Figure 4.4(d).

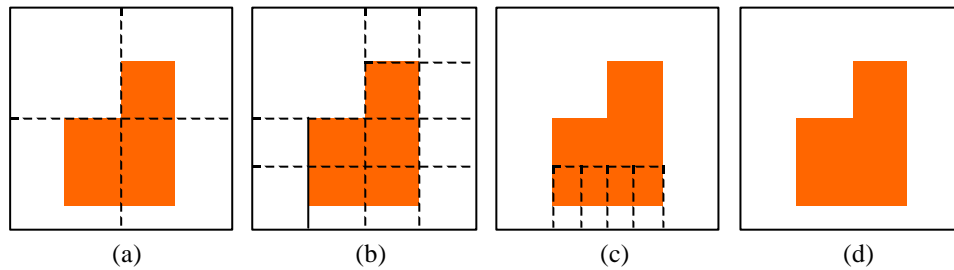


Figure 4.4 Split and merge methodology.

Figure 4.5(b) shows the region merge and split segmented image corresponding to the original in Figure 4.5(a).



Figure 4.5 (a) Original image.



Figure 4.5 (b) Split and merge segmented image.

4.2.3 Region growing segmentation

As its name implies, *region growing* is a procedure that groups pixels or subregions into larger regions. The simplest of these approaches is *pixel aggregation*, which starts with the first pixel of the image then starts growing that region by appending its neighbouring pixels that have similar properties (such as grey level, texture, colour, etc.). After growing the first region it moves to the next pixel in the image that is not allocated to any region before. This process is continued until all of the pixels have been assigned to a region. This procedure is easily illustrated with the simple example below.

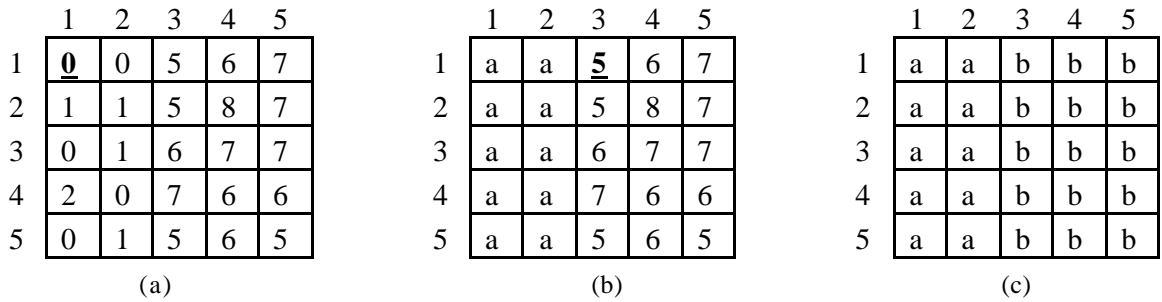


Figure 4.6 A simple example showing region growing.

The numbers inside each cell of Figure 4.6(a) represent the grey level values of a small 5×5 pixel image. The first point with co-ordinates (1,1) represents the first seed point and is indicated here in bold and underlined. Using this starting (seed) point, the segmentation results in a region R_1 and the rest of the image pixels remain the same. Then we move on to the next pixel in the image that has not been allocated to any region before i.e. the pixel with co-ordinates (1,3), and start appending the pixels to this new seed point until the segmentation results in two regions R_1 and R_2 . The property P to be used to include a pixel in either region is that the absolute difference between the grey level of that pixel and the grey level of the seed should be less than a threshold T . Any pixel that satisfies this property for a seed is (arbitrarily) assigned to region R_1 or R_2 . Figure 4.6(b) shows the result obtained using $T = 3$. In this case, the segmentation consists of two regions, where the points in region R_1 are denoted a and the points in region R_2 by b . The example image segmented using this algorithm is shown in Figure 4.7.



Figure 4.7 (a) Original image.



Figure 4.7 (b) Region growing segmented image.

4.2.4 Histogram thresholding based segmentation

In histogram based thresholding, the image histogram is used for setting various thresholds to partition the given image into distinct regions. It is expected that each region within the image will have some mean grey level intensity and a small spread around this central value that pixels in this region will take. By examining the various peaks of the histogram denoting grey levels that occur with the highest frequency, we can use them as thresholds to partition the image. The algorithm for segmentation is given below.

Algorithm Histogram Thresholding segmentation

1. For an image I with pixel grey levels $[0, L-1]$, first compute the frequency with which different grey levels occur. Let us denote the grey level variable as x , i.e. x lies in the range 0 to $L-1$ where L is 256 for our study. Hence $x_0=0$ and $x_{255}=255$. The frequency value corresponding to grey level x_i is denoted as $f(i)$.
2. Divide the histogram into a total of $n=L/y$ segments, where y is a user set parameter equal to the size of the segment. There are a total of n segments. For each segment j find its peak. Peak corresponds to grey level corresponding to the largest frequency. We can denote this grey level as x_j and frequency $f(j)$. Store these values in an array P .
3. Set $N=0$.
4. Now for $j=0$ to $n-1$ do
 If $((f(j+1) > f(j))$ and $(f(j+1) > f(j+2))$
 Then put $f(j+1)$ and x_{j+1} in an array called *final*.
 Increment N by 1 each time a new value is added to *final*.
5. N is the total number of objects found in the image. Each member can be relabelled as frequency $h(k)$ corresponding to grey level x_k for a total number of elements equal to N .
6. For each pair of successive x_k values, compute the lowest frequency in the original histogram. This is done as follows:
 For each pair x_k and x_{k+1} where $0 \leq k \leq size-1$, let $a = x_k$ and $b = x_{k+1}$
 Find the gray level that corresponds to the least frequency between a and b
 A total of $N-1$ thresholds are derived at the end
7. Segment the image using these thresholds into different regions.

The identification of the final peaks is shown in Figure 4.8.

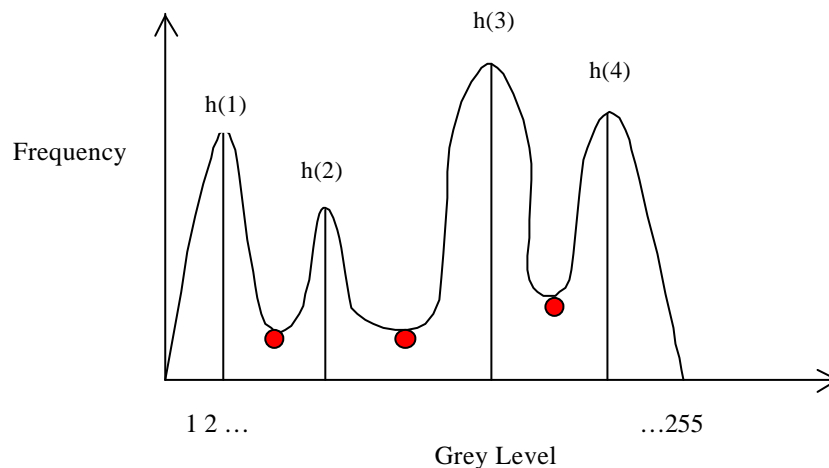


Figure 4.8 The identification of three thresholds from the four peaks in histogram thresholding.



Figure 4.9 (a) Original image.



Figure 4.9 (b) Histogram thresholded image.

4.3 Autofill

The segmented file contains some very small regions that are less than 500 pixels in size that cannot be eliminated using median filter. So there is a need of some additional processing to manage these small regions. Also, to extract features from the segmented file, a Mapfile needs to be generated which identifies each region uniquely. A recursive technique was adapted to solve these two problems prior to feature extraction. We named this technique 'Autofill'. This technique merges all regions less than 500 pixels to the regions in which they are enclosed and also provides a unique label to every region in the Mapfile. Autofill is described by the following algorithm.

Algorithm Autofill

For each pixel in the image,

- 1 If it is part of an existing region goto 1.
- 2 If it is not part of an existing region
 - 2.1 Initialise counter to 0.
 - 2.2 Recursively fill the region in eight directions, labelling each pixel with the value of current iteration, and for each direction stop when a pixel is encountered that is not equal to the current pixel. For each recursive call increment counter by 1.
 - 2.3 If region is less than 500 pixels, merge with surrounding region.
- 3 Provide a unique label of every region in mapfile.

On completion, this technique achieves the desired result and because of its recursive nature, it is very compact and efficient. The following example gives a simplistic representation of how autofill works. For the sake of demonstration, let the minimum size of a region be 2.

Figure 4.10(a) represents the top right corner of a median filtered image. Cell (3,2) represents a single pixel region valued 6. Figure 4.10(b) shows the result of autofill eliminating the pixel (3,2) by merging it with the adjacent region and indexes the regions from 1 to N , where N is the total number of region in the map file.

	1	2	3	4	5
1	5	5	5	9	9
2	5	5	5	9	9
3	5	6	5	9	9
4	5	5	5	9	9
5	5	5	5	9	9

(a)

	1	2	3	4	5
1	1	1	1	2	2
2	1	1	1	2	2
3	1	1	1	2	2
4	1	1	1	2	2
5	1	1	1	2	2

(b)

Figure 4.10 A simple example showing how Autofill works.

4.4 SUSAN

The acronym SUSAN (Smallest Univalue Segment Assimilation Nucleus) is described as an entirely new non-linear low-level image processing technique. It offers *edge detection* (one dimensional feature detection), *corner detection* (two dimensional feature detection including corners, junctures etc.) and structure preserving noise reduction. The basis of SUSAN is the image pixel and its associated local area of similar brightness. This local area is referred to as USAN (Univalue Segment Assimilation Nucleus) and is said to contain important information about the image. From the size, centroid, and measure of pixel distribution values across USAN, two dimensional features and edges can be detected. The SUSAN algorithm is based on desired criteria for feature detecting: good detection (there should be a minimum number of false positives and false negatives); good localisation (the edge location must be reported as close to the correct location as possible); single detection (it must respond only to a single edge); and good processing speed (should be fast enough for image processing).

To meet these requirements, the SUSAN algorithm offers three possible uses.

1. *Edge detection* – Edges are found and drawn as links of black pixels surrounded by white pixels each side of the edge.
2. *Corner detection* – Corners are found and drawn as black pixels surrounded by white pixels.
3. *Smoothing* – Noise removal produces an enhanced image similar in appearance to a median filtered image.

Although there are many edge detection techniques available, we chose SUSAN because of two reasons. First, SUSAN is very flexible as it allows a variety of parameter changes, which means that it can be customised to suit several different applications. Second, the detected edges with SUSAN are black edge pixels surrounded by a white border. This means that once the edges are found, it is a very simple process to retrieve these edges. The details of the SUSAN framework are available from: <http://www.fmrib.ox.ac.uk/~steve/susan>.

4.5 Texture based feature extraction

Texture features provide a measure of the underlying texture within a given region. A variety of methods are used to extract texture characteristics. Some of the commonly used approaches include the use of spatial frequencies, edge frequencies, run lengths, pixel's joint probability distribution, and special masks such as Law's masks. It has been demonstrated in different studies that different feature extraction methods yield different results based on the application

domain and it is for this reason that a diverse range of techniques were investigated. Here we describe the above approaches of feature extraction.

4.5.1 Autocorrelation

The textural character of an image depends on the spatial size of texture primitives. Large primitives give rise to coarse texture (e.g. rock surface) and small primitives give fine texture (e.g. silk surface). An autocorrelation function can be evaluated that measures this coarseness. This function evaluates the linear spatial relationships between primitives. If the primitives are large, the function decreases slowly with increasing distance whereas it decreases rapidly if texture consists of small primitives. However, if the primitives are periodic, then the autocorrelation increases and decreases periodically with distance. The set of autocorrelation coefficients C shown below are used as texture features:

$$C_{ff}(p, q) = \frac{MN}{(M-p)(N-q)} \frac{\sum_{i=1}^{M-p} \sum_{j=1}^{N-q} f(i, j) f(i+p, j+q)}{\sum_{i=1}^M \sum_{j=1}^N f^2(i, j)}$$

where p, q is the positional difference in the i, j direction, and M, N are image dimensions.

4.5.2 Co-occurrence matrices

Statistical methods use second order statistics to model the relationships between pixels within the region by constructing Spatial Grey Level Dependency (SGLD) matrices. A SGLD matrix is the joint probability occurrence of grey levels i and j for two pixels with a defined spatial relationship in an image. The spatial relationship is defined in terms of distance d and angle q . If the texture is coarse, and distance d is small compared to the size of the texture elements, the pairs of points at distance d should have similar grey levels. Conversely, for a fine texture, if distance d is comparable to the texture size, then the grey levels of points separated by distance d should often be quite different, so that the values in the SGLD matrix should be spread out relatively uniformly. Hence, a good way to analyse texture coarseness would be, for various values of distance d , some measure of scatter of the SGLD matrix around the main diagonal. Similarly, if the texture has some direction, i.e. is coarser in one direction than another, then the degree of spread of the values about the main diagonal in the SGLD matrix should vary with the direction q . Thus texture directionality can be analysed by comparing spread measures of SGLD matrices constructed at various distances d and direction q .

From SGLD matrices, a variety of features can be extracted. From each SGLD matrix, fourteen statistical measures are extracted using the following nomenclature:

$p(i, j) = p(i, j) / R$	(i, j) th entry in a normalised SGLD matrix, $p(i, j, d, \mathbf{q})$ where i, j are the grey scale values of pixels at distance d pixels apart, and angle \mathbf{q} is the angle of the line joining the centres of these pixels in the creation of the SGLD matrix.
$p_x(i) = \sum_{j=1}^{N_g} p(i, j)$	i th entry in the marginal-probability matrix obtained by summing the rows of $p(i, j)$.
N_g	Number of grey levels in the image.
$p_y(j) = \sum_{i=1}^{N_g} p(i, j)$	j th entry in the marginal-probability matrix obtained by summing the columns of $p(i, j)$.
$p_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j); k = 2, 3, \dots, 2N_g$	The result of adding the i th and j th entries calculated above from the marginal-probability matrix.
$p_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j); k = 0, 1, 2, \dots, 2N_g - 1$	The result of subtracting the i th and j th entries calculated above from the marginal-probability matrix.

The fourteen texture features suggested by Haralick et al. [86] are given by:

Angular Second Moment (ASM)	$f_1 = \sum_{i=1} \sum_{j=1} (p(i, j))^2$
Contrast	$f_2 = \sum_{n=0}^{N_g-1} n^2 \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j) \right)$
Correlation	$f_3 = \frac{\sum_i \sum_j (i, j) p(i, j) - \mathbf{m}_x \mathbf{m}_y}{\mathbf{s}_x \mathbf{s}_y}$ <p>where $\mathbf{m}_x = \sum_{i=0}^{n-1} i p_x(i)$</p> $\mathbf{s}_x^2 = \sum_{i=0}^{n-1} (i - \mathbf{m}_x)^2 p_x(i)$ $\mathbf{m}_y = \sum_{j=0}^{n-1} j p_y(j)$ $\mathbf{s}_y^2 = \sum_{j=0}^{n-1} (j - \mathbf{m}_y)^2 p_y(j)$
Sum of Squares: Variance	$f_4 = -\sum_i \sum_j (i - \mathbf{n})^2 p(i, j)$
Inverse Different Moment	$f_5 = -\sum_i \sum_j \frac{1}{1 + (i - j)^2} p(i, j)$
Sum Average	$f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i)$
Sum Variance	$f_7 = \sum_{i=2}^{2N_g} (i - f_6)^2 p_{x+y}(i)$

<i>Sum Entropy</i>	$f_8 = - \sum_{i=2}^{2N_g} p_{x+y}(i) \log(p_{x+y}(i))$
<i>Entropy</i>	$f_9 = - \sum_i \sum_j p(i, j) \log(p(i, j))$
<i>Difference Variance</i>	$f_{10} = \text{variance of } p_{x-y}$
<i>Difference Entropy</i>	$f_{11} = - \sum_{i=0}^{N_g-1} p_{x-y}(i) \log(p_{x-y}(i))$
<i>Information Measure of Correlation I</i>	$f_{12} = \frac{HXY - HXY1}{\max(HX, HY)} \text{ where}$ $HXY = - \sum_i \sum_j p(i, j) \log(p(i, j)) \text{ and}$ $HXY1 = - \sum_i \sum_j p(i, j) \log(p_x(i) p_y(j)) \text{ and}$ $HXY2 = - \sum_i \sum_j p_x(i) p_y(j) \log(p_x(i) p_y(j))$ <p>and HX and HY are entropies of p_x, p_y.</p>
<i>Information Measure of Correlation II</i>	$f_{13} = (1 - \exp[-2.0(HXY2 - HXY)])^{\frac{1}{2}}$ <p>where</p> $HXY = - \sum_i \sum_j p(i, j) \log(p(i, j)) \text{ and}$ $HXY1 = - \sum_i \sum_j p(i, j) \log(p_x(i) p_y(j)) \text{ and}$ $HXY2 = - \sum_i \sum_j p_x(i) p_y(j) \log(p_x(i) p_y(j))$ <p>and HX and HY are entropies of p_x, p_y.</p>
<i>Maximal Correlation Coefficient</i>	$f_{14} = (\text{second largest eigenvalue of } Q)^{1/2}$ <p>where $Q(i, j) = \sum_k \frac{p(i, k) p(j, k)}{p_x(i) p_y(k)}$</p>

Table 4.1 Haralick's 14 texture measures based on co-occurrence matrices.

The *angular second moment* is a measure of homogeneity of the image. In a homogenous image there are very few dominant grey tone changes and the SGLD matrix will have only a few entries of a large magnitude. Conversely, in a less homogenous image there will be many entries in the SGLD matrix of smaller magnitude and hence the ASM will be smaller in magnitude. The *contrast* feature is the difference moment of the SGLD matrix and a measure of the amount of local variation in the image. A low value of contrast results from uniform images whereas images with large variation produce a high value. *Correlation* is a measure of grey level linear-dependency within the image. It measures the degree to which the rows and columns of SGLD matrix resemble each other. High values are obtained when the matrix elements are uniformly equal and low values are obtained for a matrix with large differences in element values. *Entropy* of the image is a measure of the randomness of grey pixel values. When the SGLD matrix is equal, the entropy is the highest. Low values for entropy are obtained when matrix elements are very different from each other (large variability). Hence higher values for entropy indicate greater randomness in the image.

4.5.3 Edge frequency

A number of edge detectors can be used to yield an edge image from an original image. We can compute an edge dependent texture description function E as follows:

$$E(d) = |f(i, j) - f(i + d, j)| + |f(i, j) - f(i - d, j)| + |f(i, j) - f(i, j + d)| + |f(i, j) - f(i, j - d)|$$

This function is inversely related to the autocorrelation function. Texture features can be evaluated by choosing specified distances d (*pixel distance*).

4.5.4 Law's method

Laws observed that certain gradient operators such as Laplacian and Sobel operators accentuated the underlying microstructure of texture within an image. This was the basis for a feature extraction scheme based a series of pixel impulse response arrays obtained from combinations of 1-D vectors shown in Figure 4.11. Each 1-D array is associated with an underlying microstructure and labelled using an acronym accordingly. The arrays are convolved with other arrays in a combinatorial manner to generate a total of 25 masks, typically labelled as $L5L5$ for the mask resulting from the convolution of the two L5 arrays [127].

$$\begin{array}{ll} \text{Level} & L5 = [1 \quad 4 \quad 6 \quad 4 \quad 1] \\ \text{Edge} & E5 = [-1 \quad -2 \quad 0 \quad 2 \quad 1] \\ \text{Spot} & S5 = [-1 \quad 0 \quad 2 \quad 0 \quad -1] \\ \text{Wave} & W5 = [-1 \quad 2 \quad 0 \quad -2 \quad 1] \\ \text{Ripple} & R5 = [1 \quad -4 \quad 6 \quad -4 \quad 1] \end{array}$$

Figure 4.11 Five 1-D arrays identified by Laws.

These masks are subsequently convolved with a texture field to accentuate its microstructure giving an image from which the energy of the microstructure arrays is measured together with other statistics. The energy measure for a neighbourhood centred at $F(j, k)$, $S(j, k)$, is based on the neighbourhood standard deviation computed from the mean image amplitude:

$$S(j, k) = \frac{1}{W^2} \left[\sum_{m=-w}^w \sum_{n=-w}^w [F(j+m, k+n) - M(j+m, k+n)]^2 \right]^{\frac{1}{2}}$$

where $W \times W$ is the pixel neighbourhood and the mean image amplitude $M(j, k)$ is defined as:

$$M(j, k) = \frac{1}{W^2} \sum_{m=-w}^w \sum_{n=-w}^w F(j+m, k+n)$$

4.5.5 Primitive length (run length) encoding

A large number of neighbouring pixels of the same grey level represent a coarse texture, a small number of these pixels represents a fine texture, and the lengths of texture primitives in different directions can be used for texture description. A primitive is a maximum contiguous set of constant grey-level pixels located in a line. These can then be described by grey level, length and direction. The texture description features is then based on computation of continuous probabilities of the length and the grey level of primitives in the texture.

Let $B(a, r)$ be the number of primitives of all directions having length r and grey level a . Let A be the area of the region in question, let L be the number of grey level within that region and let N_r be the maximum primitive length within the image. The texture description features can then be determined as follows. Let K be the total number of runs:

$$K = \sum_{a=1}^L \sum_{r=1}^{N_r} B(a, r)$$

The features are defined as:

a) Short primitives emphasis

$$\frac{1}{K} \sum_{a=1}^L \sum_{r=1}^{N_r} \frac{B(a, r)}{r^2}$$

b) Long primitives emphasis

$$\frac{1}{K} \sum_{a=1}^L \sum_{r=1}^{N_r} B(a, r) r^2$$

c) Grey level uniformity

$$\frac{1}{K} \sum_{a=1}^L \left[\sum_{r=1}^{N_r} B(a, r) r^2 \right]$$

d) Primitive length uniformity

$$\frac{1}{K} \sum_{a=1}^L \left[\sum_{r=1}^{N_r} B(a, r) \right]^2$$

e) Primitive percentage

$$\frac{K}{\sum_{a=1}^L \sum_{r=1}^{N_r} r B(a, r)} = \frac{K}{A}$$

4.6 Classifiers

Classification is the process where a given test sample is assigned a class on the basis of knowledge gained by the classifier during training. We have used two classifiers: linear discriminant analysis and nearest neighbour classifier. To make the classification results comparable and for exhaustive data analysis we have used leave-one-out classification for both classifiers. Two different models of the nearest neighbour classifier are used in our study as proposed by Singh et al.[195]. The results for both the models are shown.

4.6.1 Linear Discriminant Analysis (LDA)

Linear analysis uses linear boundaries between data distributions to discriminate between samples. Linear combinations of the independent, sometimes called predictor, variables are formed and serve as the basis for classifying cases into one of the groups. In essence, linear discriminant analysis can be used for classifying samples that are linearly separable. If classes are linearly separable, then it is a good indication that the features that have been selected are discriminatory (Figure 4.12(a)). The type of distribution shown in Figure 4.12(b) would not allow a 100% classification result using standard linear discriminant analysis because the data distributions are overlapping (i.e. data not linearly separable).

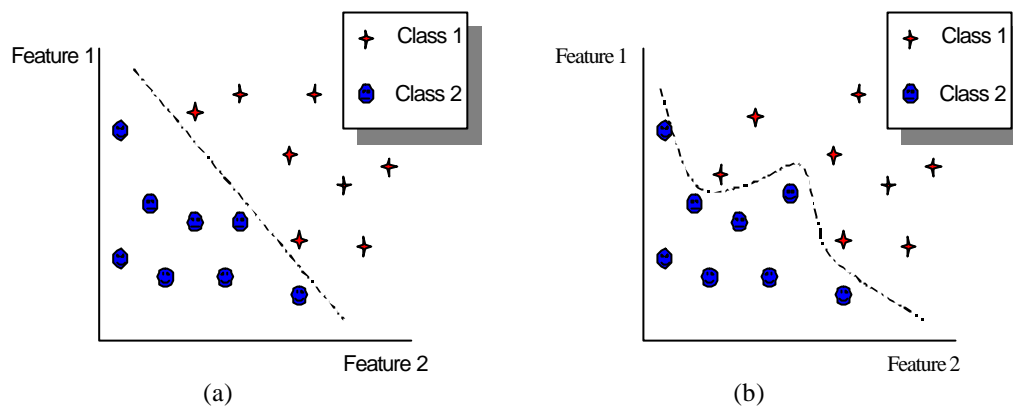


Figure 4.12 Decision boundaries for LDA: (a) linearly separable data; (b) non-linearly separable data.

On the basis of the training data available, LDA generates the boundaries that separate different class distributions. The linear discriminant equation assigns weights to each variable to give the “best” separation between groups. The linear discriminant boundary for separating data of different classes is given by:

$$D = B_0 + B_1 X_1 + B_2 X_2 + \dots + B_p X_p$$

where X_i are the values of the variables or features and the B_i are the coefficients estimated from the data. For successful classification, two classes must differ in their D values. Unknown samples are classified using the coefficients determined from training data. To perform LDA classification, we use a statistical analysis software package called SPSS for Windows (www.spss.com). SPSS incorporates a simple data set import and allows many common statistical methods to be applied.

4.6.2 Nearest neighbour classifier

Nearest neighbour methods have been used as an important pattern recognition tool. In such methods, the aim is to find the nearest neighbours of an unidentified test pattern within a hypersphere of pre-defined radius in order to determine its true class. The traditional k nearest neighbour rule has been described as follows.

Algorithm traditional kNN

- Out of N training vectors, identify the k nearest neighbours, *irrespective* of class label. k is chosen to be odd.

- Out of these k samples, identify the number of vectors, k_i , that belong to class \mathbf{w}_i , $i=1, 2, \dots, M$. Obviously $\sum_i k_i = k$.
- Assign x to the class \mathbf{w}_i with the maximum number k_i of samples.

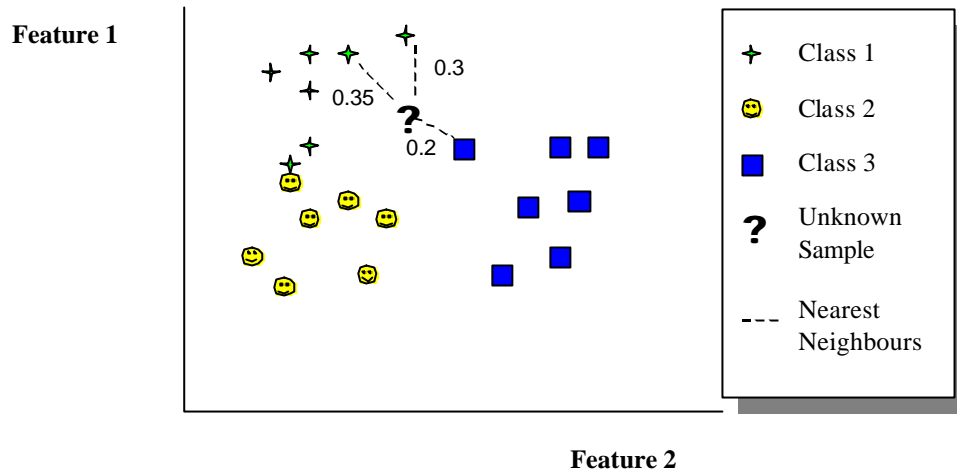


Figure 4.13 Nearest neighbour classifier in action on a two class problem.

As shown in Figure 4.13, we need to determine the distances of a test pattern from surrounding neighbours. Nearest neighbour methods can detect a single or multiple number of nearest neighbours. A single nearest neighbour method is primarily suited to recognising data where we have sufficient confidence in the fact that class distributions are non-overlapping and the features used are discriminatory. In most practical applications however the data distributions for various classes are overlapping and more than one nearest neighbours are used for majority voting. In k -nearest neighbour methods, certain implicit assumptions about data are made in order to achieve a good recognition performance. The first assumption requires that individual feature vectors for various classes are discriminatory. This assumes that feature vectors are statistically different across various classes. This ensures that for a given test data, it is more likely to be surrounded by data of its true class rather than of different classes. The second assumption requires that the unique characteristic of a pattern that defines its signature, and ultimately its class, is not significantly dependent on the interaction between various features. In other words, nearest neighbour methods work better with data where features are statistically independent. This is because nearest neighbour methods are based on some form of distance measure and nearest neighbour detection of test data is not dependent on their feature interaction. In this study, two models are used for the nearest neighbour classification. The algorithms for these are described below.

Model-1 kNN rule

- Out of n training vectors, identify the k nearest neighbours, *irrespective* of class label. k is chosen to be odd.
- Out of these k samples, identify the number of vectors, k_i , that belong to class \mathbf{w}_i , $i=1, 2, \dots, M$. Obviously $\sum_i k_i = k$.
- Assign x to the class \mathbf{w}_i with the maximum number k_i of samples.
- If two or more classes \mathbf{w}_i , $i \in [1 \dots M]$, have an equal number E of maximum nearest neighbours, then we have a tie (*conflict*). Use conflict resolution strategy.
- For each class involved in the conflict, determine the distance d_i between test pattern $x = \{x_1, \dots, x_N\}$ and class \mathbf{w}_i based on the E nearest neighbours found for class \mathbf{w}_i . If the m^{th}

training pattern of class w_i involved in the conflict is represented as $y^{im} = \{y_1^{im}, \dots, y_N^{im}\}$ then

the distance between test pattern x and class w_i is: $d_i = \frac{1}{E} \sum_{j=1}^N |x_j - y_j^{im}|$

- Assign x to class C if its d_i is the smallest, i.e. $x \in w_C$, if $d_C < d_i$ for $\forall i$, such that $C \in [1..M]$ and $i \neq C$.

Model-2 kNN rule

- Out of n training vectors, identify the k nearest neighbours, *irrespective* of class label. k is chosen to be odd.
- Out of these k samples, identify the number of vectors, k_i , that belong to class w_i , $i=1, 2, \dots, M$. Obviously $\sum_i k_i = k$.
- Find the average distance d_i that represents the distance between test pattern $x = \{x_1, \dots, x_N\}$ and E_i nearest neighbours found for class w_i , $i = 1..M$. Only include classes for which samples were detected in the first step. If the m^{th} training pattern of class w_i found within the hypersphere is represented as $y^{im} = \{y_1^{im}, \dots, y_N^{im}\}$, then the distance between test pattern x

and class w_i is: $d_i = \frac{1}{E_i} \sum_{j=1}^N |x_j - y_j^{im}|$

- Assign x to class C if its d_i is the smallest, i.e. $x \in w_C$, if $d_C < d_i$ for $\forall i$, such that $C \in [1..M]$ and $i \neq C$. The decision in this model does not depend on the number of nearest neighbours found but solely on the average distance between the test pattern and samples of each class found.

The two models can be explained using Figure 4.14. In Figures 4.14 (a) to (d), we have assumed a total of five classes ('a' to 'e'). The samples of each class are represented by symbols 'a' to 'e'. The test pattern is shown as the square block around which a hypersphere is drawn to determine the number of neighbours included in the analysis. In the traditional nearest neighbour implementation, Figure 4.14(a) would assign the test pattern to class 'a' as there are two samples of class 'a' and only one sample of class 'b' within the boundary. Such decisions are based only on the number of nearest neighbours found. In Figure 4.14(b) we show the problem of neighbour conflict. In such cases, equal number of neighbours is found for more than one class when determining the class of the test pattern. We term this a *conflict*. Conflicts can be resolved by either increasing the size of the hypersphere, i.e. involving more neighbours for a clear-cut decision, or by using conflict resolution described in model-1 above. Figure 4.14(c) shows model-2 process of finding the true class of the test pattern. Here the distance from a given class to the test pattern represents the averaged distance of all samples of that class found within the hypersphere. If all samples of a given class, e.g. 'e' in Figure 4.14(c), lie outside the hypersphere, then these are not included in the analysis. In all nearest neighbour methods, the number of neighbours analysed has a very important effect on the results of the analysis. This is shown in Figure 4.14(d). In this figure, when using the inner sphere, the class assignment for the unknown test pattern is 'a'. When we consider more neighbours with the outer sphere, the class assignment changes to 'd'. Thus, one important parameter to optimise in nearest neighbour methods is the number of neighbours included in the analysis.

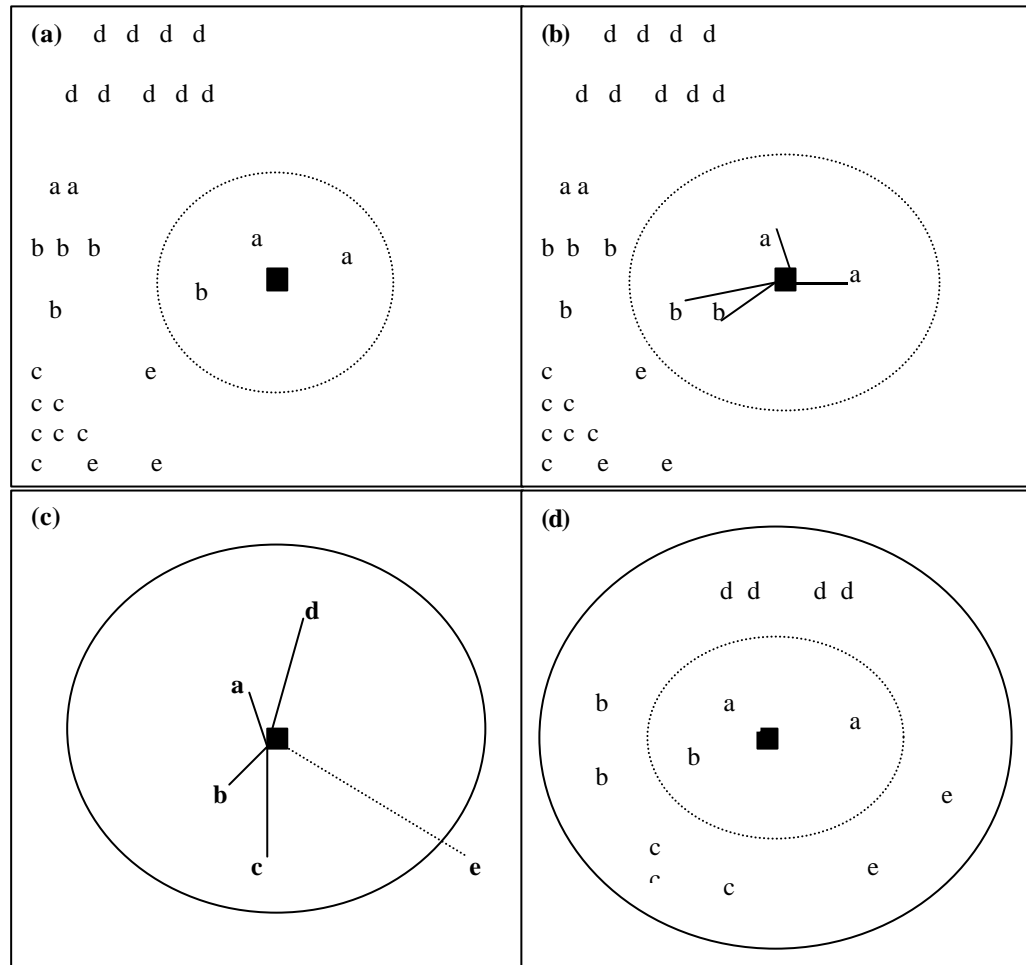


Figure 4.14 k-nearest neighbour models and strategies: (a) Traditional k-nearest neighbour model; (b) Conflict resolution; (c) Closest average distance model; and (d) Hypersphere size effect.

4.6.3 Data dimensionality reduction

In order to reduce the computational time, it is desirable to reduce the original number of features into a smaller data set with reduced dimensionality such that the original data variance is retained with the new smaller set of features. Also, the use of such data can improve classification results as the effect of those features that are least variable on the classification process is much reduced. Principal Component Analysis is a popular technique for this. Its basics are explained below.

Principal Components Analysis (PCA)

PCA is a means of reducing the dimensionality of data. Essentially this reduces the number of features that we are using for classification. In PCA, linear combinations of the observed variables are formed. The first principal component is the combination that accounts for the largest amount of variance in the data. The second principal component accounts for the next largest amount of variance and is uncorrelated with the first. Successive components explain progressively smaller portions of the total variance, and all are uncorrelated with each other.

If we have variables $V_1 \dots V_n$, then performing PCA will give us resultant variables $P_1 \dots P_n$. It is possible to compute as many principal components as there are variables (features) however in

practice only the first c components are chosen with eigenvalues greater than 1. The first few principal components contribute more to the total variance and hence the selection of fewer components may be sufficient to represent the data. Principal components with negligible variance can be removed from any calculation to give us a resultant list of principal components, $P_1 \dots P_c$, where $c \ll n$.

The algorithm for determining principal components is as follows.

Algorithm PCA

- 1) Normalise the features X_1, X_2, \dots, X_p to have zero means and unit variances (i.e. scale all features relative to one another so that they now lie in the range $[-1$ to $1]$).
- 2) Calculate the covariance matrix C .
- 3) Find the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_p$ and the corresponding eigenvectors a_1, a_2, \dots, a_p . The coefficients of the i th principal component are then given by a_i and λ_i is its variance. Calculate the principal component score from the above.
- 4) Discard any principal components that only account for a small variation in the data. The best method of finding how many principal components to choose is based on plotting the cumulative variance explained by the components. A typical plot will show steep decline earlier on and the curves flattens rapidly as the addition of more principal components does not provide much of an advantage. In most cases, principal components that taken together explain 95% or above are chosen. They can also be chosen from the above plot by leaving principal components in the flat region of the plot.

The above algorithmic descriptions give a basic insight into how the various image processing and pattern recognition tools have been used in this study. Now with this description given along with the methodology described in the previous chapter, we move on to describe our first set of results on MeasTex and VisTex benchmarks in the next chapter.

Chapter 5

Texture benchmark analysis

5.1 Texture benchmarks

The evaluation of image analysis on artificial and natural texture benchmarks is important. Texture benchmarks allow a range of feature extraction techniques to be compared without involving any image segmentation step in the process. One of the most popular examples of texture benchmarks is the Brodatz album [25] that contains photographs of well-known textures. A number of studies have used this benchmark. There are also other well-known repositories. We have used two of the most popular that are called MeasTex [138] and VisTex [214]. The details of these benchmarks have been presented in section. In this chapter we present the results of evaluating the performance of five feature extraction methods on these two benchmarks. These features include *autocorrelation*, *co-occurrence matrices*, *edge frequency*, *Law's measures*, and *primitive length*. We evaluate the performance of these methods on a complete set of selected features without using any feature selection method. In order to evaluate the recognition performance, we use two classifiers. The first classifier used is the linear discriminant classifier. The method is supposed to work best when the data distributions can be linearly separated. From our preliminary data analysis we know that data distributions are not linearly separable. However, the results will show us quantitatively how easy it is to classify data using a simple linear method and on the basis of such results the need for using a non-parametric classifier can be justified. The second classifier used is k nearest neighbour classifier. As the data is not linearly separable, the results tend to improve by using non-linear classifier.

5.2 Benchmark analysis

We discuss the results of applying our five selected feature extraction techniques on MeasTex and VisTex benchmark data using leave-one-out cross-validation. The linear analysis has been applied using the standard statistical package SPSS. For the linear analysis, for each feature extraction method, a single analysis is performed yielding its confusion matrix. For the k -nearest neighbour method, we experiment with $k=1, 3, 5$ and 7 neighbours and for each analysis we obtain a different result in the form of a different confusion matrix and the average recognition rate. We have presented the confusion matrices of both classifier analyses in appendices A-D.

5.3 MeasTex analysis

In Table 5.1 we show the number of features extracted for each texture analysis method.

<i>Feature extraction method</i>	<i>No. of features</i>	<i>No. of classes</i>	<i>No. of samples</i>
Autocorrelation	99	4	944
Co-occurrence	20	4	944
Edge frequency	50	4	944
Law's	125	4	944
Primitive length	5	4	944

Table 5.1 Details of feature extraction methods used on MeasTex data.

Features are extracted for a total of four classes. As discussed previously, each MeasTex image has been divided into sixteen parts giving a total of 944 samples from all of the images. We first discuss the results of applying the linear classifier in section 5.3.1 and then in the following section 5.3.2 we present the results of applying the k -nearest neighbour classifier.

5.3.1 Linear classifier results

Data visualisation in 2D can be useful for the visual understanding of how well separated the data is for different classes. One of the easiest techniques for visualising the complexity of the classification problem is to plot the scores of the first and the second principal components for data samples labelled by their class. Since principal components compress the data in terms of its dimensionality without affecting its complexity, this plot can be useful in understanding how much overlap exists between the class distributions. The usefulness of the plots depends on how accurately the visible overlaps reflect the real situation that in turn depends on the percentage of data variance explained by the first two components. Plots of the first two principal components that explain the majority of variance give a realistic picture of actual data distribution overlaps. The relationship between the discriminant functions or variables is shown with the canonical discriminant functions plot. The graph plots the variate scores for each sample for different groups or classes. The group centroids are highlighted and by visualising the plot we can see which variate discriminates better between the data groups.

In Figure 5.1 we show the PCA plot and the canonical discriminant function plot for *autocorrelation* features.

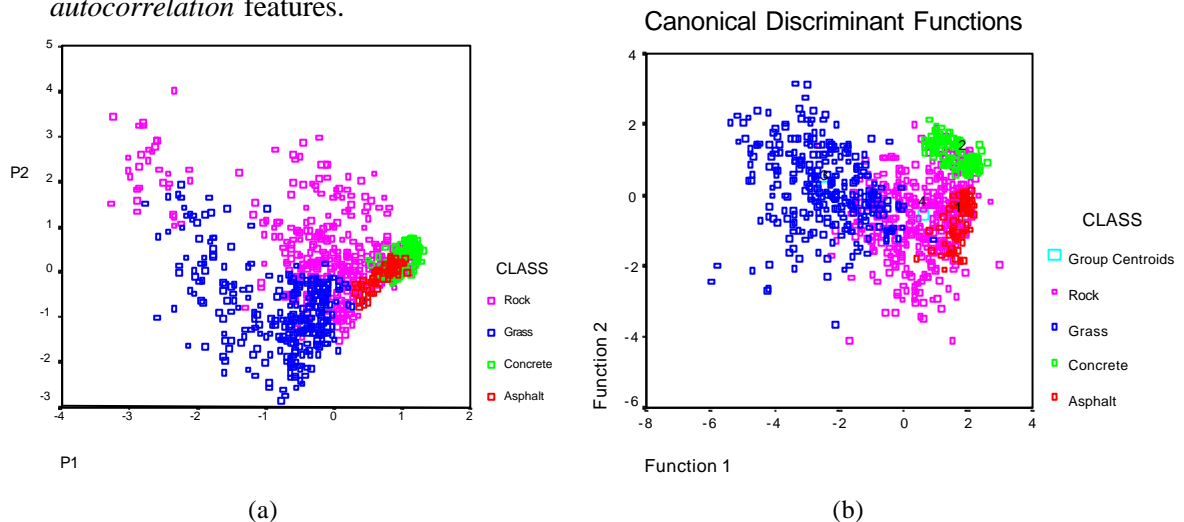


Figure 5.1 (a) Plot of the first two principal components for the *autocorrelation* features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *autocorrelation* features on MeasTex data.

In both plots we can see that asphalt samples are least separated from other class samples whereas grass samples are best separated. The plots also show the variability within each class. Classes such as asphalt and concrete are the most compact in terms of small inter-class variability whereas grass and rock samples are varied. Finally, the plots give some indication of the amount of outliers present in each class. Classes that are highly varied contain more outliers in Figure 5.1.

We obtain a correct recognition rate of 76.1% using leave-one-out validation method. For the first three classes, we obtain a recognition rate of 90% or above, however, the results are much poorer for rock for which we obtain just over 50% correct recognition.

We next evaluate the co-occurrence features. Figure 5.2 shows the PCA plot and the discriminant function plot. With these features, more compact clusters are formed by grass and asphalt data. Concrete and rock data is distributed in more than one cluster. The overall recognition rate of 79.2% correct is attributed to the improved ability of the classifier in recognising rock and asphalt samples. Inspection of the confusion matrix in Appendix A shows that the system is capable of correctly identifying asphalt and grass samples with 100% accuracy. Recognition rates for other classes are: concrete (68%), and rock (67%). Concrete samples are most confused with asphalt, and rock samples predominantly with grass and concrete.

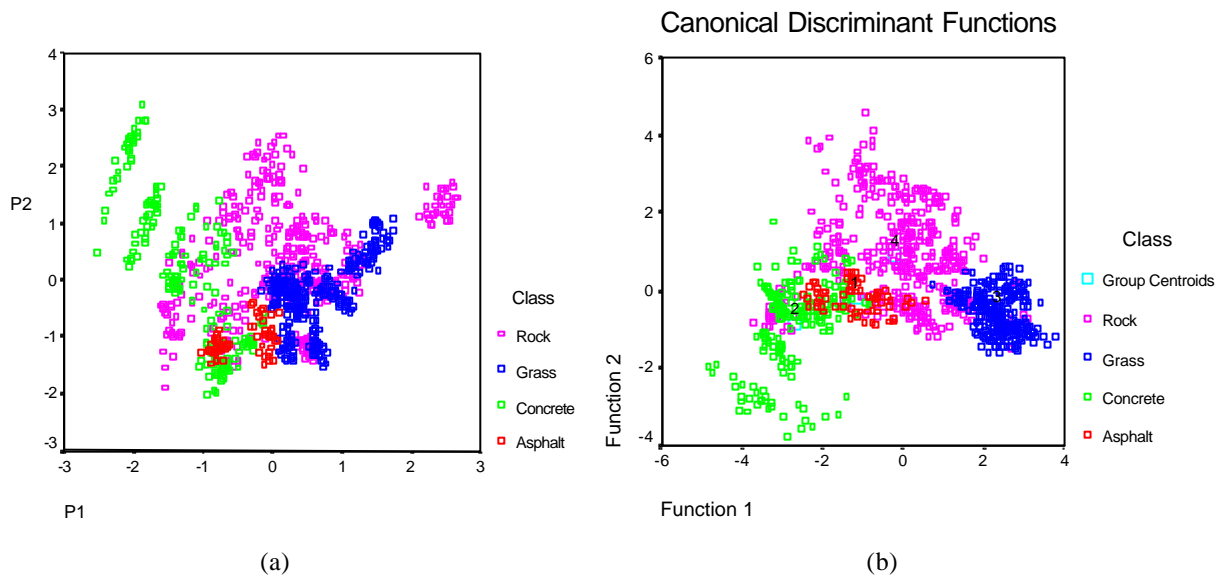


Figure 5.2 (a) Plot of the first two principal components for the *co-occurrence* features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *co-occurrence* features on MeasTex data.

In Figure 5.3, edge frequency measures have been used as input to the linear classifier. Rock samples are again by far the most varied along with grass samples. The concrete and asphalt yield the best compact clusters containing the least number of outliers. As a result of this large variability and significant overlap, it can be expected that the classification will not yield good results. The confusion matrix shows an overall recognition rate of 63.5% with the worst performances on grass and rock samples. The correct recognition rates on these four classes are: asphalt (86%); concrete (96%); grass (65%) and rock (42%). It is noticeable that grass samples are most confused with rock samples in this analysis.

Law's features have been obtained using 25 spatial masks and their PCA and discriminant function plots have been shown in Figure 5.4. These have by far the most compact clusters obtained using any other feature extraction method so far. In the PCA plot, the main boundaries of data overlap lie across grass and rock, rock and asphalt, asphalt and concrete, and concrete and rock samples. Rock samples are by far the most varied. The LDA method achieves the best recognition rate of 82.8% correct. This improved rate is a result of better recognition on all

classes, especially on rock and concrete. The individual correct recognition rates for all classes are: asphalt (89%); concrete (77%); grass (97%) and rock (74%).

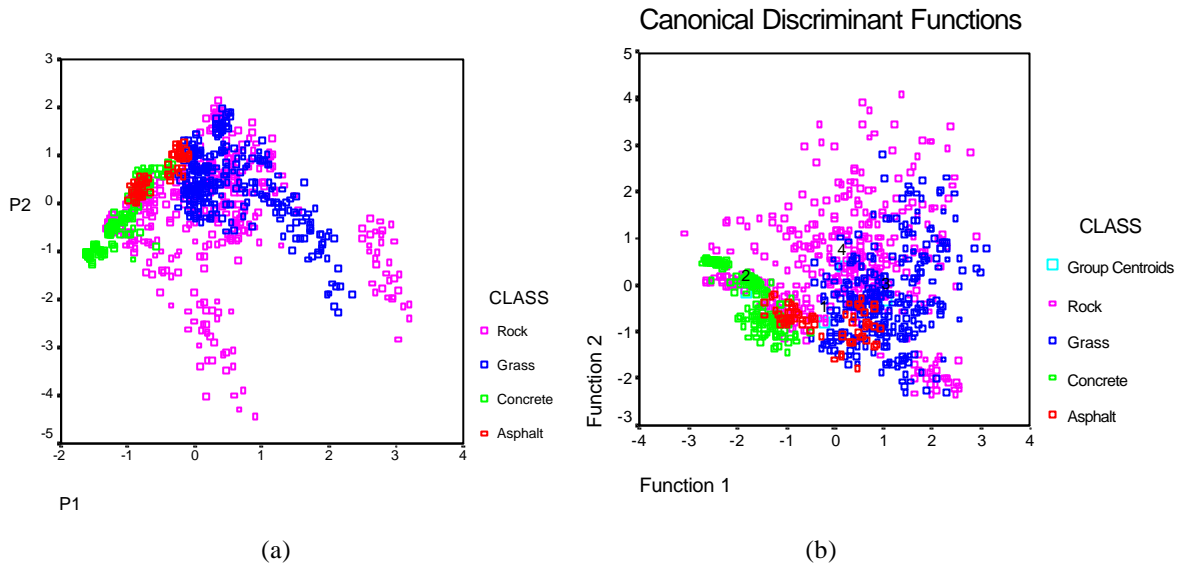


Figure 5.3 (a) Plot of the first two principal components for the *edge frequency* features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *edge frequency* features on MeasTex data.

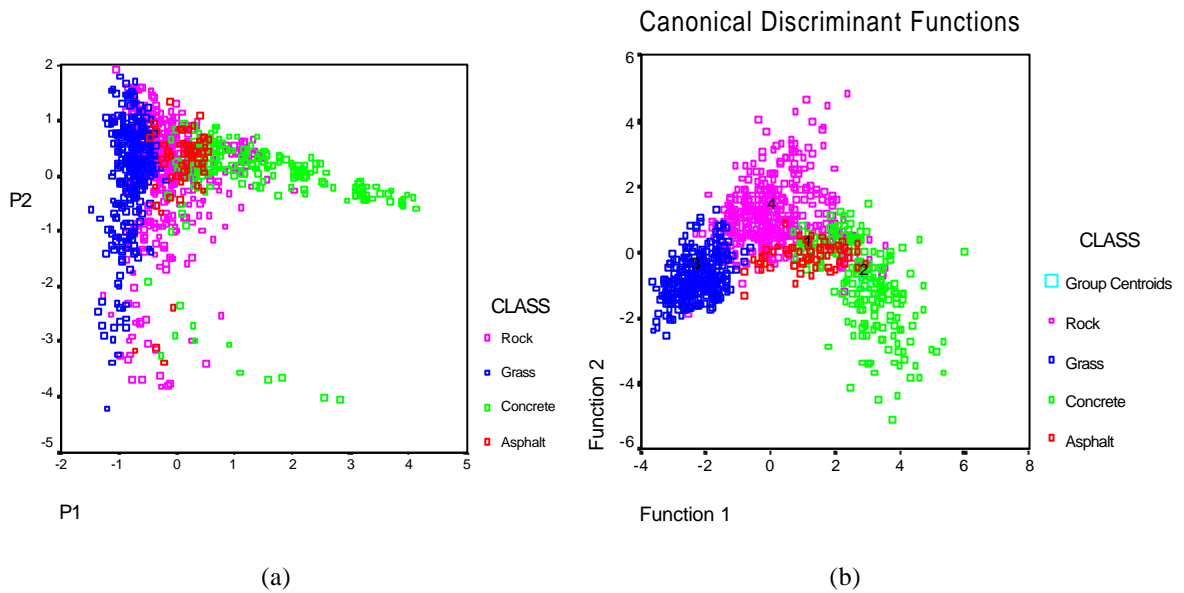


Figure 5.4 (a) Plot of the first two principal components for the *Law's* features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *Law's* features on MeasTex data.

Next we evaluate the performance of the primitive length features on MeasTex data using the linear classifier. The plot in Figure 5.5 shows that the data variability can be explained along one of the axes. Concrete samples are the most discriminatory whereas there is a strong overlap across data of other classes. The PCA plot shows how limited the features are in discriminating different textures for MeasTex analysis. Confusion matrix results are fairly poor: asphalt (33%); concrete (78%); grass (48%) and rock (25%).

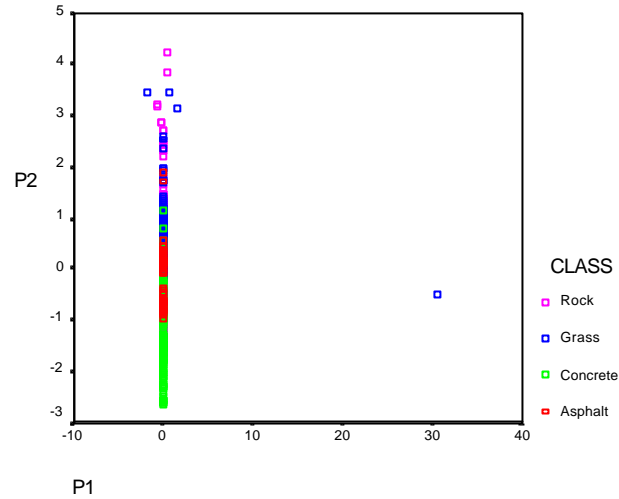


Figure 5.5 PCA plot for the *primitive length* features on MeasTex data.

Finally, we evaluate how well a combined feature set would perform on the given problem. Not all features from the five methods are useful in discriminating between the four texture samples and as such their combination without any feature selection process may not necessarily lead to better performance. The PCA and canonical function plots are shown in Figure 5.6. We find that compact but elongated clusters have been formed for data of all classes. By far, the plots show the best situation possible. The confusion matrix shows an overall recognition rate of nearly 88% correct with individual recognition rates for the four classes as follows: asphalt (100%); concrete (94%); grass (96%); and rock (75%). The main success has been high recognition rates on grass and concrete while maintaining good performance on recognising rock data.

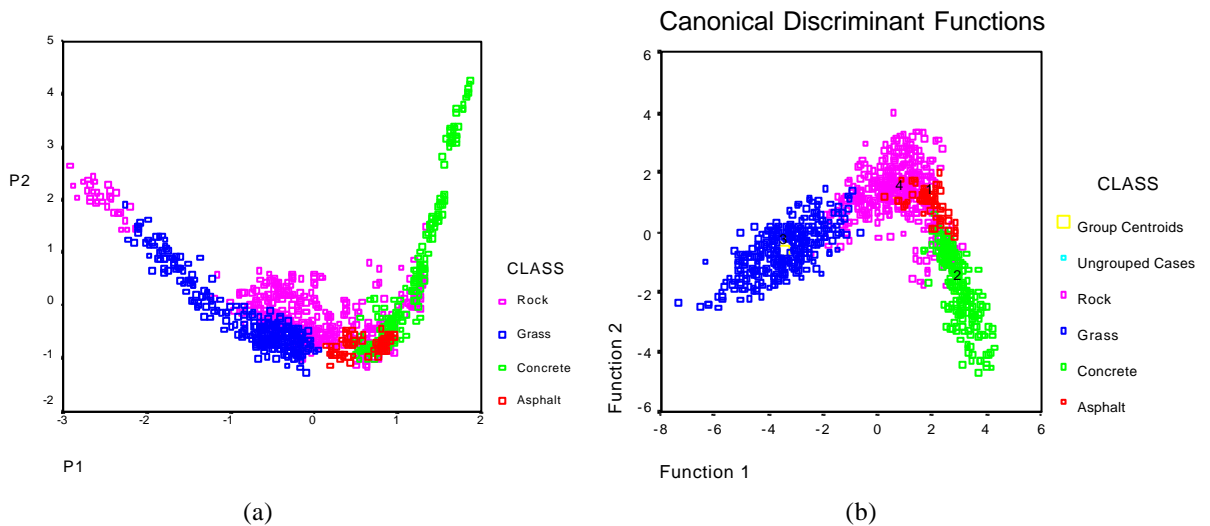


Figure 5.6 (a) Plot of the first two principal components for the *combined* features on MeasTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *combined* features on MeasTex data.

5.3.2 Nearest neighbour classifier results

We next discuss the results of using k nearest neighbour classifier on MeasTex data. We have used two models of nearest neighbour classifier. In the first model, the classifier assigns a test pattern to the class for which we find maximum number of neighbours within a hypersphere

surrounding the test pattern. In those cases where two or more classes tie as the winning classes, a conflict resolution strategy is adopted on the basis of estimating average sample distance. In the second model, only average distances rather than the number of samples are used for determining the class of a test pattern. The algorithms for these have been detailed in chapter 4.

The confusion matrices of MeasTex feature classification using nearest neighbour classifier appear in Appendix B. On each page we present the results of one feature extraction method using both nearest neighbour models. The results have been produced for $k = 1, 3, 5$ and 7 nearest neighbours (only odd values are used as prescribed by Devijver[52]). When $k = 1$, both models are equivalent. For *autocorrelation features*, we find that the best performance is achieved by the first model, for $k=5$, which gives the best performance of 79.5% correct recognition. Using a different value of k does not have much difference on the recognition rate. For the confusion matrix of this best performing classifier, recognising rock samples is the most difficult as it was the case with linear classifier. For misclassified patterns, the asphalt patterns are most confused as rock, concrete as asphalt and rock, grass as rock, and rock samples are confused as a bit of everything else.

Co-occurrence feature performance is much better for MeasTex data. The performance of the first model is better than the second model in general, and the value of k parameter has a more significant impact on model2 than model1. The best classifier performance is 86.9% correct recognition for samples from four classes. Even though similar mistakes are made on classification as autocorrelation method, the number of mistakes is much smaller.

Edge frequency results are not as good as the two previous methods. Model1 performance with $k=3$ is the best with a recognition rate of 70.7%. Rock samples are most confused with grass samples. The confusion matrix looks similar to other methods in terms of where the mistakes are being made leading us to conclude that edge frequency features have similar data overlap problems as the other two methods.

Law's features demonstrate similar performance to edge frequency method with a best recognition rate of 70.9% for model1 ($k=7$). The confusion matrix shows the difficulties because of rock samples that are classed as other objects and other class samples that are labelled as rock. Similar to other methods, for model2 classifier as k is increased, the performance drops (in this case from 63.4% for $k=1$ to 53.2% for $k=7$).

Out of all texture analysis methods discussed, *primitive length features* yield the poorest performance. The best performance is shown by the model1 with $k=7$ with a result of 54.1% correct recognition. The main reason for this poor performance is a significant overlap between rock and grass samples as more grass samples are confused as rock and vice-versa. The number of ties is larger than for other feature data and only 54% of ties get correctly resolved. In several cases, what should have been the winning class, turns out to be the second best.

Finally, we consider the *combined features* performance. The recognition performance is very good but only second best to the co-occurrence matrix performance. The best performance is displayed by the first model ($k=5$) at 83.3% correct recognition. Compared to the confusion

matrix for co-occurrence matrix for the same model and same k parameter, we find that the combined features are as good as the co-occurrence features in allowing us to assign grass samples correctly; on all other classes, the combined feature performance declines by a small amount.

Our main conclusion from this analysis is that on the whole, the first classifier model performs better than the second model. The performance of the second model decreases as the value of k is increased. In general, all feature sets except Law's and combined gave better performances with the nearest neighbour method compared to LDA analysis. For Law's feature set, LDA gave results of 82.8% and combined features yielded 87.5% recognition which is better than equivalent nearest neighbour best results of 70.9% and 83.3% respectively.

5.4 VisTex analysis

The analysis of VisTex data is more complicated than MeasTex. There are several reasons for this. First, there is a larger number of classes involved. The increase in the number of classes does not always increase the complexity of the classification problem provided that the class data distributions are non-overlapping. However, in our case we find that VisTex class distributions are overlapping and the classification problem is by no means entirely solvable using linear techniques alone. Second, VisTex data has much less number of samples for each class and it is expected that the unbalance between samples across different classes will make the classification more difficult. Third, and of most concern, is the significant variability across samples of the same class in VisTex benchmark. This aspect will be evident in the form of scattered clusters in PCA plots, irrespective of the features used.

<i>Feature extraction method</i>	<i>No. of features</i>	<i>No. of classes</i>	<i>No. of samples</i>
Autocorrelation	99	7	300
Co-occurrence	20	7	300
Edge frequency	50	7	300
Law's	125	7	300
Primitive length	5	7	300

Table 5.2. Details of feature extraction methods used on VisTex data.

In Table 5.2 above we show the number of features used for different texture extraction methods and summarise data details. We first present the results of applying linear classifier on the feature sets derived from previously discussed techniques. Next, we will apply the k nearest neighbour model on these feature sets and evaluate any changes in performance.

5.4.1 Linear classifier results

We first apply the linear discriminant analysis to autocorrelation features that have been extracted from VisTex images. As mentioned earlier, we are only considering the images of the following classes: water, tile, sand, metal, food, fabric and bark. In Figure 5.7 the PCA plot and the canonical discriminant function plots are shown. These two plots confirm the variability across samples from the same class. In particular classes such as fabric samples are particularly varied whereas water samples have small inter-class variability.

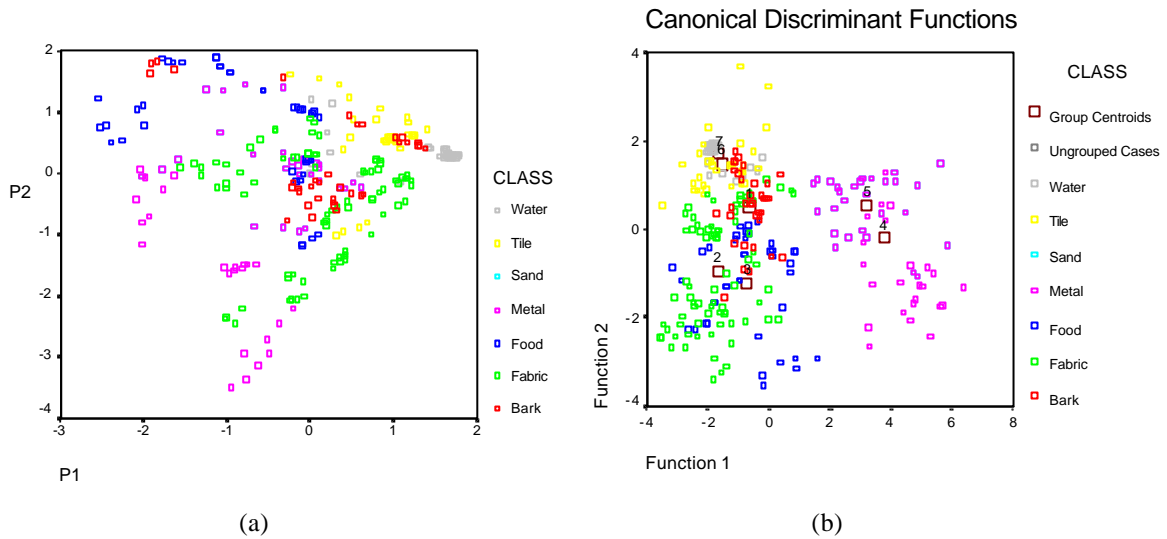


Figure 5.7 (a) Plot of the first two principal components for the *autocorrelation* features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *autocorrelation* features on VisTex data.

The detailed results in the form of the confusion matrices are shown in Appendix C. Overall, we obtain a recognition rate of 72.1%. We find that water, sand, and food samples are the easiest to recognise. The individual recognition rates for the seven classes are: bark (63.9%); fabric (65.0%); food (83.3%); metal (66.7%); sand (82.1%); tile (53.1%) and water (96.9%). The misclassification of one class as another is important to note. For example, misclassified bark samples are mostly confused as tile or water. Similarly, misclassified fabric samples are confused as either bark or food; misclassified food samples are confused as bark or fabric; misclassified metal samples are predominantly confused as sand; and misclassified tile samples are confused as either bark or water. Our immediate impression from such results is that there are not enough samples in this benchmark that would give a more comprehensive grouping of samples in different classes, and more importantly, the images available show high amounts of inter-class variability in terms of their extracted features thus yielding poorer classification results.

Figure 5.8 shows the co-occurrence feature distribution for the VisTex data in the form of PCA and discriminant function plots. Food and bark samples particularly overlap with others. However, these plots show better separation across samples of different classes which makes us optimistic that better results can be derived. Co-occurrence features can be discriminated for seven classes using the linear classifier with 73.9% accuracy. On the whole, fabric is the most difficult class to recognise, but all other class samples can be easily recognised. The recognition rates for samples of seven classes are: bark (72%); fabric (60%); food (81%); metal (96%); sand (86%); tile (78%) and water (70%). In particular, we can summarise the misclassifications as follows: most misclassifications for bark are confused as fabric or food; most misclassifications for fabric are confused as bark or food; most misclassifications for food are confused as fabric; most misclassifications for sand are confused as bark; most misclassifications for tile are confused as water, fabric or food; and most misclassifications for water are confused as bark, food or tile.

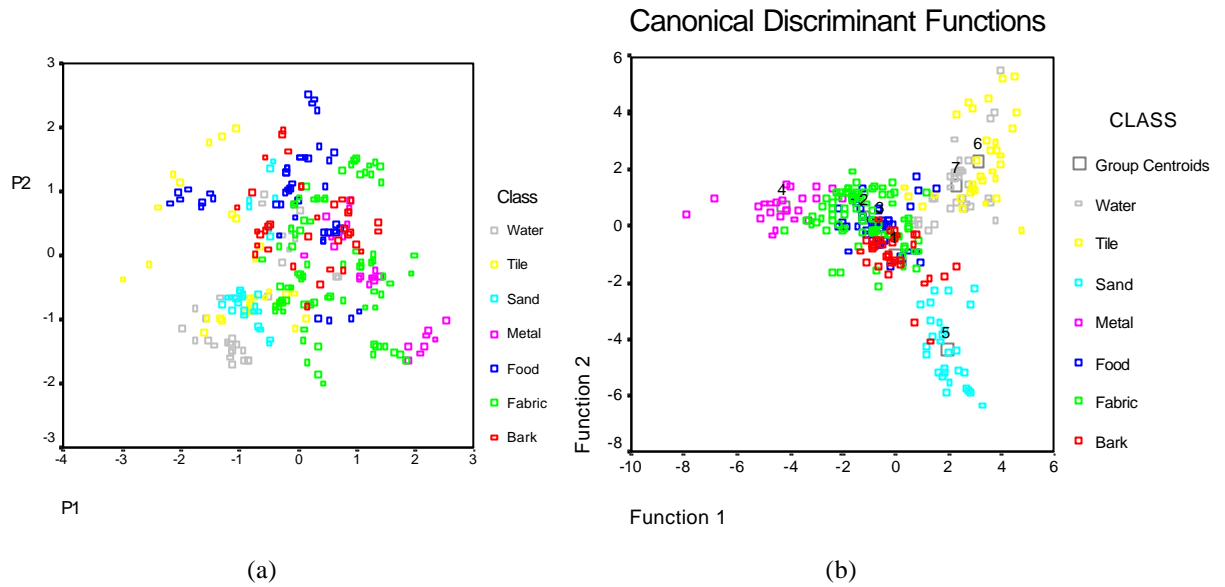


Figure 5.8 (a) Plot of the first two principal components for the *co-occurrence* features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *co-occurrence* features on VisTex data.

In Figure 5.9 we show the plots obtained with edge frequency measures on VisTex data using PCA and linear discriminant analysis.

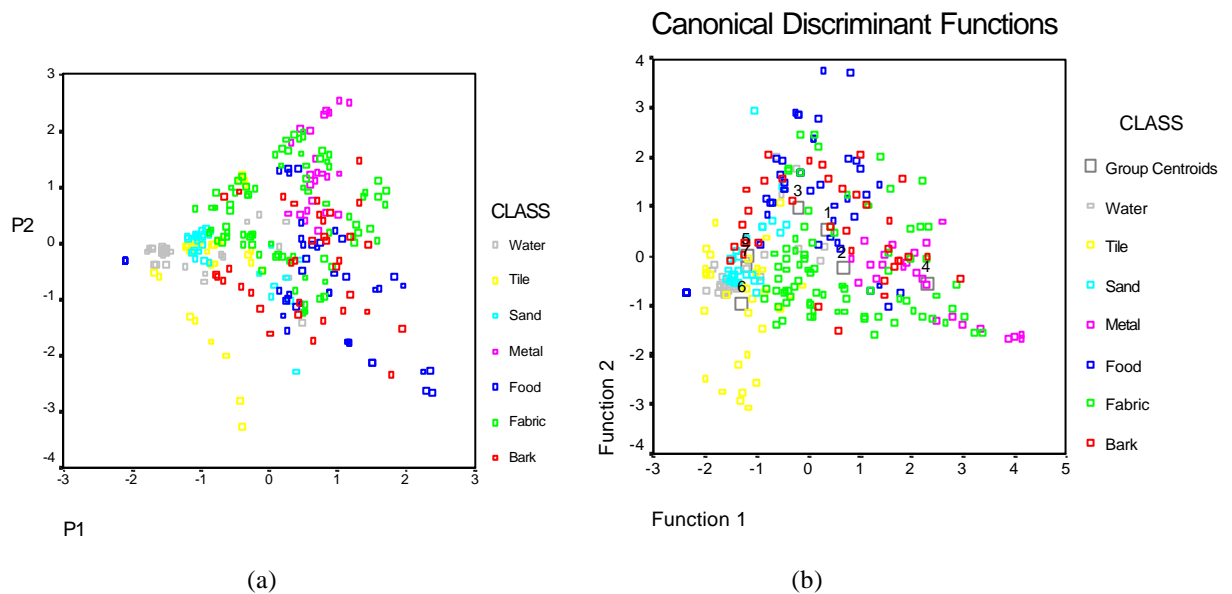


Figure 5.9 (a) Plot of the first two principal components for the *edge frequency* features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *edge frequency* features on VisTex data.

This is by far one of the most confusing situations where samples for each class appears scattered amongst samples of other classes. The recognition rate for the complete set is fairly poor at 53.2%. The individual recognition rates for different classes are as follows: bark (36%); fabric (31%); food (65%); metal (92%); sand (68%); tile (47%) and water (75%). The classifier makes a larger number of mistakes than before as: most misclassified bark samples are labelled as fabric, metal or sand; most misclassified fabric samples are labelled as food, metal or

water; most misclassified food samples are labelled as fabric; most misclassified metal samples are labelled as fabric; most misclassified sand samples are labelled as water; most misclassified tile samples are labelled as sand or water; and finally, most misclassified water samples are labelled as bark.

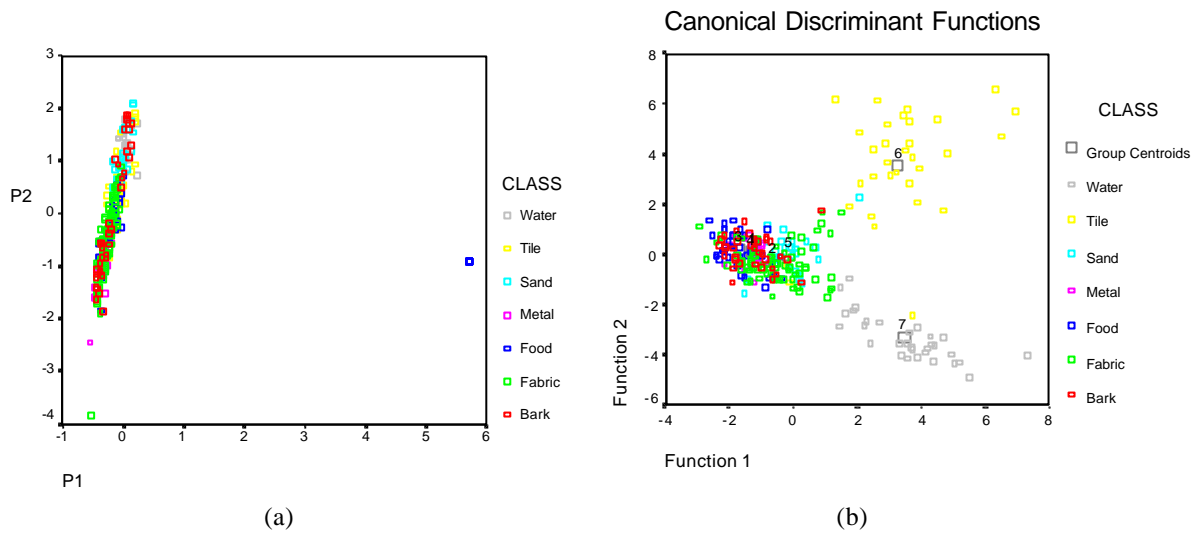


Figure 5.10 (a) Plot of the first two principal components for the *Law's* features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *Law's* features on VisTex data.

Law's features again prove to be very successful in characterising texture in VisTex images. Their PCA plot in figure 5.10 shows that the majority of data variability is explained by the first component. The discriminant function plot shows that it is relatively hard to discriminate between samples of classes except water and tile. An overall recognition rate of 68.8% is not as good as 73.2% for co-occurrence matrices but still quite good using a linear classifier. The individual recognition performances on the classes are: bark (50%); fabric (58%); food (65%); metal (79%); sand (82%); tile (81%); and water (91%). The majority of misclassified bark samples are confused as fabric, metal or sand; the majority of misclassified fabric samples are confused as food or metal; the majority of misclassified food samples are confused as bark or water; the majority of misclassified metal samples are confused as fabric; the majority of misclassified sand samples are confused as water; the majority of misclassified tile samples are confused as sand or water; and the majority of misclassified water samples are labelled as bark.

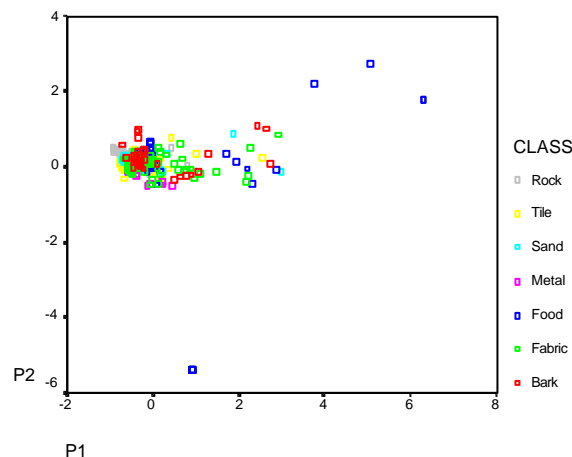


Figure 5.11 Plot of the first two principal components for the *primitive length* features on VisTex data.

Figure 5.11 shows the PCA plot for primitive length features. The data distribution is most overlapping for all classes. As a result of the significant overlap across different classes, a very poor overall recognition rate of 34.8% is obtained. A close inspection of the confusion matrix shows that for some classes, a greater percentage of samples are recognised as something else compared to being correctly classified. For example, more fabric samples are classed as bark and metal than fabric itself; more food samples are classed as bark than food itself; more sand samples are classed as water than sand itself; and more tile samples are classed as sand or water than tile. The individual recognition rates for the seven classes are: bark (42%); fabric (24%); food (25%); metal (100%); sand (21%); tile (25%); and water (50%).

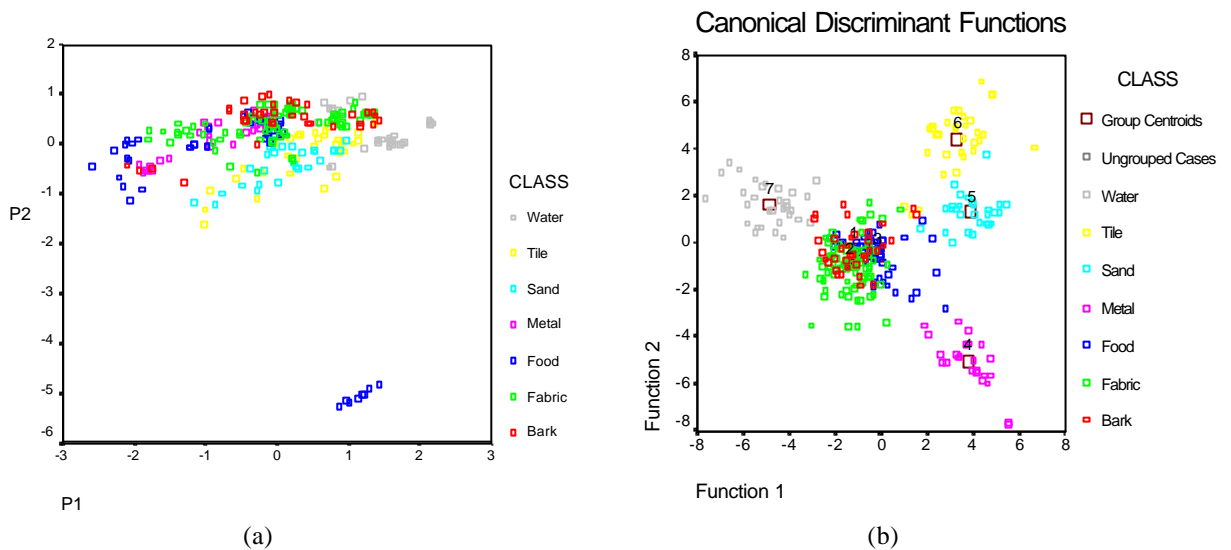


Figure 5.12 (a) Plot of the first two principal components for the *combined* features on VisTex data; (b) Plot of the discriminant functions generated by the linear classifier based on *combined* features on VisTex data.

Figure 5.12 shows the plots obtained when using the combined feature set. As a result of adding more information, the classes now appear separable. The confusion matrix shows radically improved results on VisTex data provided that features using different texture methods are used together. An overall recognition rate of 94.6% is by far the most impressive. The samples of individual classes have been recognised with the following accuracy: bark (94%); fabric (96%); food (83%); metal (100%); sand (96%); tile (100%) and water (97%). The only explanation for such a good performance can be based on the hypothesis that different feature extraction methods measure something different but when features are combined together, we get a very powerful set of descriptors. It is not a complete surprise to see good performance from the combined feature set as improvements were also noticed for MeasTex data. However, the magnitude of improvement in performance is significant and most noticeable.

5.4.2 Nearest neighbour classifier results

The k -nearest nearest neighbour models when applied to VisTex data show good performance. The confusion matrices are presented in Appendix D. On the whole, model-1 performances are better than model-2. Also, when we increase the number of nearest neighbours parameter k for model-2, we get inferior recognition rates. We now discuss the results obtained for individual feature sets.

For *autocorrelation features*, the best results of 85.7% correct recognition are obtained for the first model for $k=1$. This is much better than the linear classification of 72.1% correct. The individual classes are recognised with the following accuracy: bark (88.5%), fabric (97.5%), food (97.9%), metal (66.7%), sand (46.4%), tile (81.2%) and water (90.6%). On the whole it is relative easy to recognise samples for bark, fabric, food, tile and water. The confusion matrix shows that the mistakes are made when sand samples get confused as metal or fabric.

Co-occurrence features give a much better performance of 80.7% correct recognition for $k=1$. This better than the linear classifier performance of 73.9% correct. The individual classes are recognised with the following accuracy: bark (52.7%), fabric (87.5%), food (87.5%), metal (95.8%), sand (96.4%), tile (56.2%) and water (84.3%). The most difficult recognition is attributed to samples from tile that are confused as either bark or fabric, and samples from bark that are confused as fabric or food.

Edge frequency features show a best recognition rate of 66.8% correct for model-1, $k=3$. This is much better than the linear classifier performance of 53.2%. The classes are recognised with the following accuracy: bark (38.8%), fabric (76.2%), food (77.0%), metal (70.8%), sand (64.2%), tile (71.8%) and water (59.3%). Classes that could be recognised with near perfect accuracy with the co-occurrence approach are now confused with others, e.g. food, metal and sand. The most difficult classes to recognise are bark, tile and water.

Law's features do not perform well here and their best result is only as good as 56.1% correct recognition with model-1 ($k=7$). The performance is much worse than the linear recognition rate of 68.8%. The classes are recognised with the following accuracy: bark (22.2%), fabric (70.0%), food (50.0%), metal (54.1%), sand (57.2%), tile (50%) and water (75.0%). Most of the mistakes as made when the fabric and food samples are confused as the other and when bark gets confused as fabric, metal as fabric and tile as fabric. So the analysis is biased toward making mistakes in favour of fabric class.

For primitive length features, the best results are obtained using model-1 ($k=7$). The recognition performance of 42.4% is reasonably good compared to the linear classifier result of 34.8% correct. The classes are recognised with the following accuracy: bark (44.4%), fabric (61.2%), food (60.4%), metal (8.3%), sand (35.7%), tile (18.7%) and water (37.5%). Most of the mistakes are biased in favour of the first two classes, bark and fabric.

Finally, we consider the results on the combined data set. The best performance is given by the model-1 classifier with $k=5$. An overall recognition accuracy of 61.3% is obtained. This is nowhere close to 94.6% correct recognition obtained on the linear classifier. The classes are recognised with the following accuracy: bark (16.7%), fabric (63.7%), food (56.2%), metal (70.8%), sand (89.2%), tile (40.6%) and water (78.1%). It appears from the confusion matrix that there is a large likelihood that the misclassified patterns will be assigned to fabric class. This is proved by the mistakes made. Bark, food and tile are often confused as fabric. Fabric on the other hand is confused as bark and food.

In summary, the best performance is given by autocorrelation method showing a correct recognition rate of 85.7% correct. This is quite impressive for the seven class classification problem. Co-occurrence matrices come as a good second with an impressive performance of 80.7% correct. We also find that the use of the nearest neighbour classifier improves the performance significantly for half of the feature sets.

In this chapter we have discussed the performance comparison of different texture extraction methods. Some of this work is published in Singh and Sharma [197]. In the next chapter we discuss the PANN natural image benchmark that was developed as a part of this study. We discuss our experimental design for recognising image regions within this benchmark. This is quite important as this dictates the manner in which experiments are performed. This will be followed by experimental discussion in further chapters.

Chapter 6

Experimental design for PANN benchmark

The main purpose of this chapter is to detail the experimental design for PANN scene analysis database. PANN scene analysis database has been developed as a part of this study to provide a benchmark data set that can be used by researchers to test their natural object recognition schemes. In the following section, we discuss other scene analysis benchmarks and our reasons for not choosing them for analysis. Then we detail how our experiments are laid out that are described in the next two chapters.

6.1 Scene analysis benchmarks

There are a number of other data sets that other researchers have used for texture analysis. Some important databases have been discussed before. Unfortunately, there are not enough benchmark data sets on outdoor scene analysis. It would be inappropriate not to mention those available and their characteristics briefly to justify why it was felt necessary to develop a complete database of images as a part of this study. Most of these data repositories are available over the Internet. These datasets have been compiled by other researchers. Some of these are only for texture studies and hence only contain texture images. For such images, no segmentation is necessary. Data benchmarks on remote sensing and range data are available but not relevant to us. There are few good data repositories containing outdoor natural images. However, a number of repositories only contain very few images to be useful. For a comprehensive study, enough data is needed to generate meaningful results and as such these benchmarks are of limited use.

The two benchmarks that were found worth analysis are called Groningen Natural Stimuli collection and Bristol Image Database. The first database contains over 4000 large calibrated still images of outdoor scenes taken with a Kodak camera (<http://hlab.phys.rug.nl/archive.html>). The images have a file format of 1536x1024 pixels where each pixel is a 2 byte unsigned integer. The natural scenes contain a range of objects such as water, vegetation, buildings, etc. The Bristol Image Database consists of over 350 images of a range of urban and rural scenes. The images are digitised using a calibrated digitiser from the 35 mm colour transparency film to produce high quality 36 bit colour images. The statistics for image content and acquisition have been carefully controlled. Nearly half of the images are rural and the other half urban, spanning a variety of viewpoints. Environmental conditions during capture were dry, fully overcast, and good atmospheric visibility. For all images the camera was focussed at infinity. The images have been hand-labelled for providing ground truth. The objects have been categorised as sky, vegetation, road marking, road, pavement, building, fence/wall, road sign, signs/poles, shadow and mobile objects.

One of the key limitations of working with these databases is the unacceptably extreme variability with some of the objects present in different images. Also, some of the objects are not homogeneous in character, e.g. a building that contains windows, doors, etc. In such cases, an object is composed of several components that are individual objects in their own right. As such, segmentation and labelling would turn out to be too complex. Also, the images of rivers,

buildings, etc. have been taken at different locations, and the same natural object appears considerably different in different stills. Considering these problems, it was decided not to use these databases. PANN scene analysis benchmark data was developed with the following objectives: i) The database should contain sufficient samples of different natural objects for a meaningful analysis. The natural objects chosen for analysis were trees, grass, leaves, clouds, sky, pebbles, bricks, and road. ii) The natural objects considered should be uniform in the sense that they can not be broken into further individual components. iii) The stills should be captured directly in the digital format under similar lighting conditions, and both colour and grey scale images should be available for analysis. iv) The database should be collected from the same area or location to minimise extreme differences in samples. All data has been collected from the Exeter University campus. With these motivations, the PANN benchmark data was developed that contains 448 natural images taken by a Panasonic digital camera with a resolution of 768x570 pixels. The environmental conditions during data capture are dry, mostly overcast but good visibility.

6.2 Problem definition

In natural scene analysis, the main task is to develop sophisticated image analysis and classification algorithms that lead to better recognition of natural objects. Some of these natural objects have uniform texture, e.g. roads, brick walls, etc. Other natural objects have more of a fractal nature such as snow, leaves, trees, etc. A number of these objects are very rich in texture but the quality of texture we can extract depends considerably on the image quality and the manner in which images are taken. In this study we have tried to ensure that images are taken in good conditions and shadow effects are minimised. However, a limited number images in the database do suffer from these problems. In the majority of such images, only a few regions were affected by poor imaging conditions. Even though such images represent a more realistic view of what kind of environment an automated object recognition system must cope with, they add to the complexity of the classification task.

In our natural object classification task, the following objects have been considered: trees, grass, leaves, bricks, clouds, sky, pebbles, and road. A single image can contain more than one of these objects. Some of these classes are fairly homogeneous in their texture, e.g. bricks, sky, and road, whereas the others are more varied, e.g. trees, grass, leaves and clouds. Also, the vegetation has been imaged under different lighting conditions increasing the variability in samples. Clouds are by their very nature varied and thus their samples do not, by definition, form a very homogeneous class. An experiment on training the classifier on all classes was first performed using the different feature sets to determine the complexity of the classification task. Two points of observation soon became apparent from our experiments. First, we find that the data from different class distributions is highly overlapping. Second, we find that different clusters belonging to different classes have different amounts of variability. This is evident through the visualisation of compactness of data clusters through PCA plots. Also, it becomes apparent that there are a number of outliers present in data. An outlier can be defined as a sample of a class w that is considerably far in terms of its Euclidean distance from the mean feature vector. However, different feature sets have different data representations for the same image regions and as such different outlier samples. Their elimination for one data set may not correspond to the same samples being eliminated in other data sets. Hence, outlier elimination

for different data sets will yield different final sets because different samples have been eliminated from analysis, and that would make it impossible to compare our experiments across data sets on an even basis. For this reason, we do not remove any outliers for the purpose of this study. However, it is a reasonable hypothesis that their removal from our data sets will yield better results. The data distribution overlaps are shown in Figure 6.1.

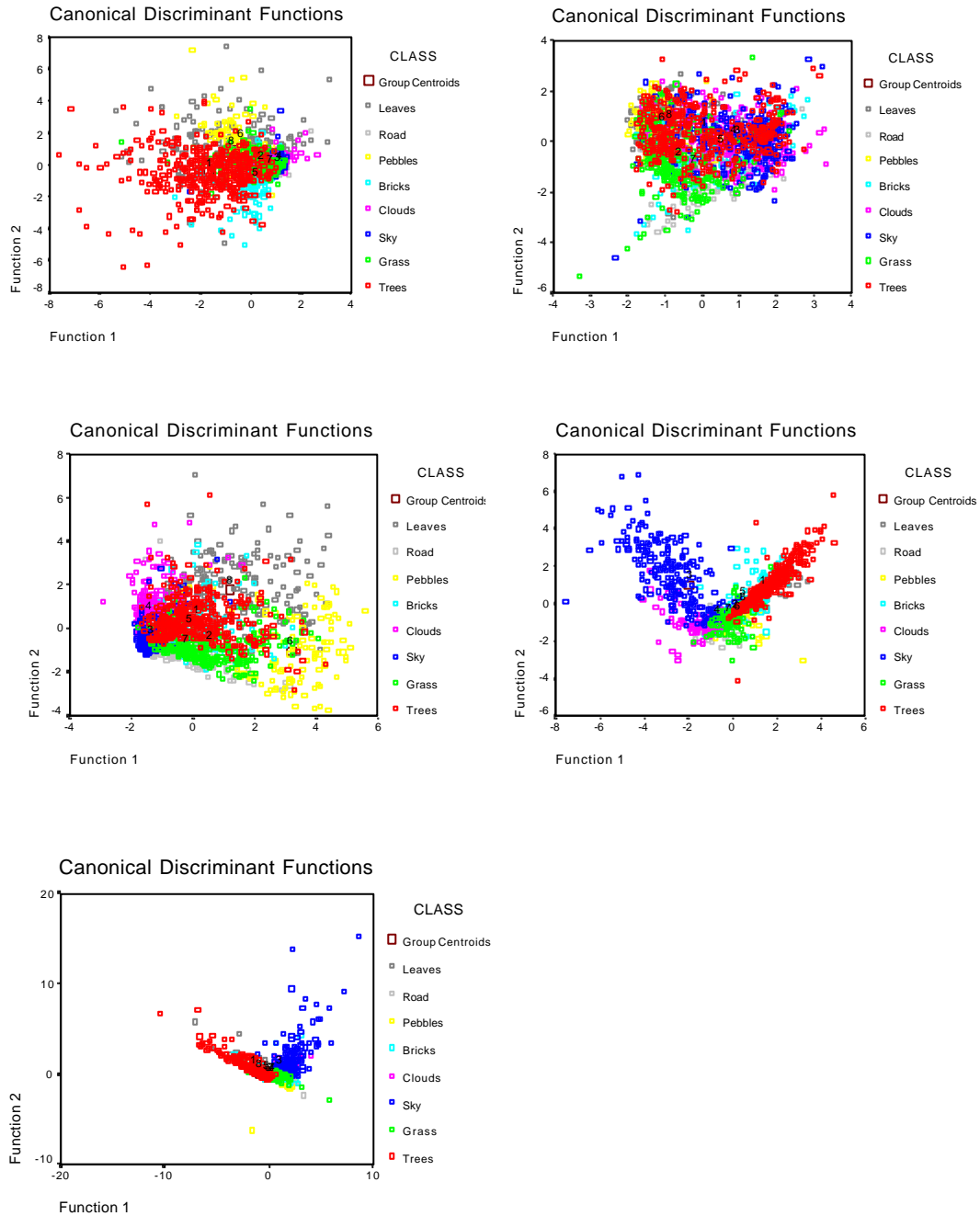


Figure 6.1 Data distribution overlaps during linear classification of natural scene analysis data containing 8 classes: vegetation (trees, grass, leaves) and natural objects (sky, clouds, bricks, pebbles, road). For all methods, FCM segmentation has been used.

There are several important observations to be made from Figure 6.1. First, in the first three feature extraction methods, there is considerable inter-class variability. Tree and grass samples appear as adjoining clusters masking the leaves cluster. Clusters, except for sky, are superimposed by vegetation clusters. For Law's and primitive length method, the clusters appear compact for trees, grass and sky masking other distributions. This situation can be exemplified by considering Figure 6.2.

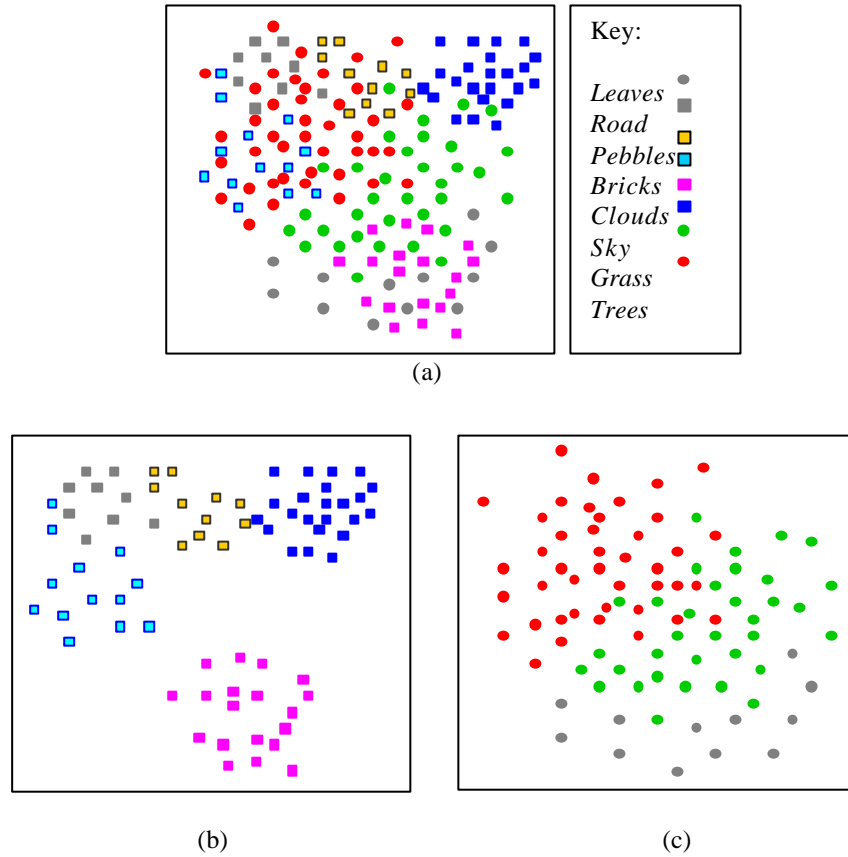


Figure 6.2 Data distribution overlaps for the natural object recognition problem: a) vegetation and natural object data overlaps showing difficulty in linear discrimination; b) natural object data after vegetation has been removed; c) vegetation after natural object data is removed.

In Figure 6.2(a) we show a graphical representation of how most class distributions overlap in our problem. A classifier would not perform well on such data. However, if we are to develop two classifiers for recognising natural object data and vegetation data separately, a much better performance can be achieved as visible from the almost linear separation between different classes shown in Figure 6.2(b) and 6.2(c).

The considerable data distribution overlap in our experiments is an important consideration in how our experiments should be designed. There are two important considerations for our experimental design. First, our primary purpose is to compare recognition success of natural objects across 20 different data sets that have been generated by using different combinations of image segmentation and feature extraction methods. This being the primary aim of the thesis, our objective is to demonstrate which data sets yield the best results using the same classifier for all of them. Second, even though this is not the primary goal, the results obtained must be reasonable enough to act as a basis of being used for a realistic system, if implemented, for

natural object recognition. What is a reasonable result is debatable. In the literature surveyed, most studies deal only with texture benchmark data rather than scene analysis data and the results are less frequently reported on scene analysis data. So it is difficult to set a baseline result for such as problem.

6.3 Experimental design

We propose that one of the two strategies proposed below will yield a good experimental design for our analysis. Either of these strategies can be implemented depending on the ease of implementation and our goals. We describe these strategies in detail below.

6.3.1 Experimental design based on two-phase classification

In this strategy, classification of natural objects can be made on the basis of separating out vegetation from the rest of the data. This is apparent from Figure 6.1(a) that shows how vegetation data is highly overlapping with others. It is hypothesised that two-stage classification will yield much better results compared to classifying data of all classes in one go. In the two-stage procedure, each image region is first identified as coming from vegetation category, or from other natural objects. This process can be handled only if the colour information contained in the images is used. In such a case, for each segmented region, based on its colour pixel values in the original image, the contribution of the green colour using colour histograms can be identified. It is relatively straightforward to identify whether an image region is from vegetation or natural objects class. Once this decision has been made, two separate classifiers can be trained to classify these separated samples. One of the classifier can be trained on recognising the three vegetation classes, and the other classifier can be trained on recognising the five natural object classes. This is shown in Figure 6.3 where classifier's accuracy is R1 and R2%.

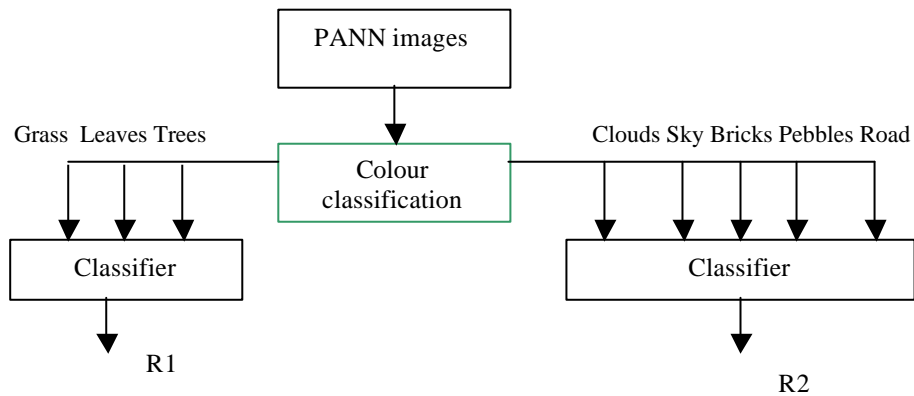


Figure 6.3 Colour classification for separating vegetation data from other natural objects.

One of the key advantages of using this two-stage scheme is that it is relatively simple to implement. Since colour image processing element acts only as a trigger as to which classifier should be used for a given test sample, we can achieve good performances using the two separate classifiers trained on grey-scale data. This hierarchical scheme ensures that we do not need to perform a full colour image analysis except for triggering different classifiers. Also, the two-stage scheme allows us to achieve better performances with data that is highly overlapping. Such an overlap is primarily as a result of vegetation data samples superimposed on the natural object clusters. Once these two categories of data are analysed with two separate classifiers, the overall performance of the recognition system improves. In our analysis, we make the

assumption that the colour image processing element is 100% accurate, i.e. we can filter out vegetation samples from others with 100% accuracy, before they are tested. The classifiers shown at the second level are based on working with the grey scale images. Hence the recognition performances of the two classifiers are a true reflection of the system performance.

6.3.2 Experimental design based on multi-stage classification

Parikh[162] has discussed the concept of multi-stage classifiers. In such a classification scheme, a number of classifier are developed. For example, for separating N different classes, we can construct a decision tree that represents different classifications at different levels. At each level i of the decision tree, a classifier is used based on a data partitioning scheme. Level $i=0$ denotes the root of the tree. How many levels are used for solving a problem depends on the number of classes N and the manner in which the experimentation is performed. We demonstrate multi-stage classification process in Figure 6.4. Consider that we have N classes ($w_1, w_2 \dots w_N$). These classes can be partitioned into a total of Ω_i groups at level i . At least each group must be of size 2 so that a classification is possible. The total numbers of combinations possible at level i and for the complete tree is considerably large. Each data group is considered as a homogeneous entity. In order to understand Figure 6.4, let us explain the symbols used. At each level, a partition or group of data ϑ can be formed.

We label this group as: $\vartheta_{\text{level } i, \text{ number in group}}^{\text{group number at level } i}$

Hence, the node ϑ_{21}^1 represents classification at level 2 with first data set in the first group. The tree can have a depth of L depending on the experimental design. In group j at level i , there are a maximum of ℓ^j nodes. At each node, the different leaves represent the data groups that are considered as separate classes and subjected to the classification performance.

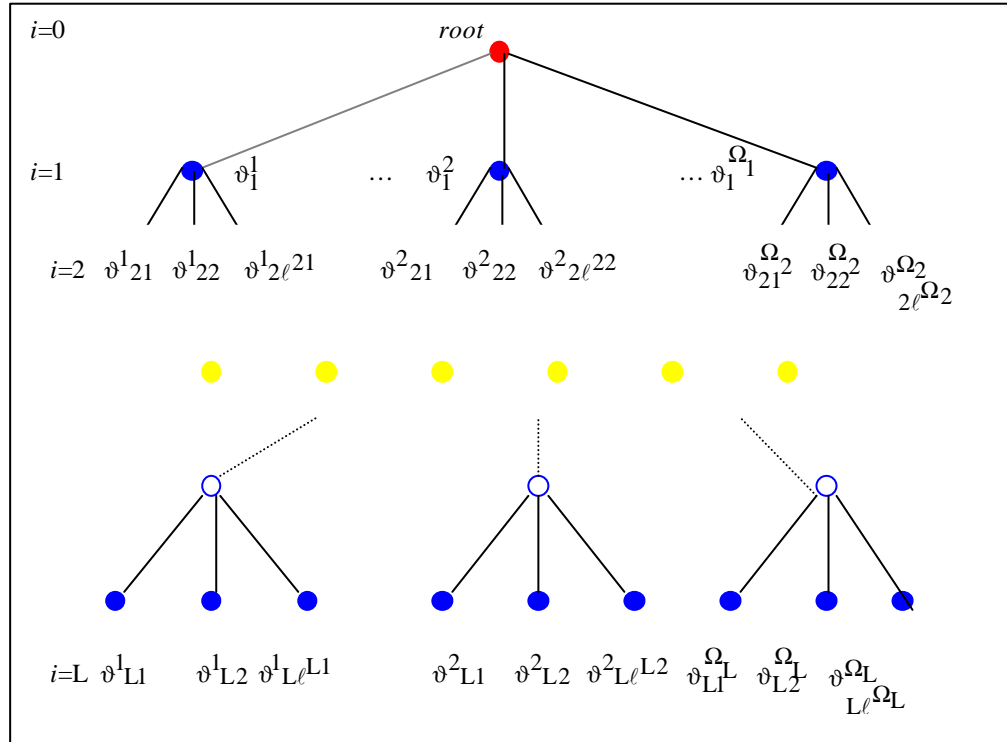


Figure 6.4 Multistage classification process where classifiers are used in a hierarchy.

Multistage classification is an important methodology where the separation of data from different classes in a hierarchical manner will yield better results rather than subjecting the complete data for classification into N classes. The recognition rate obtained finally is not a direct combination of recognition rates obtained at different levels. Only at the bottom of the tree, all misclassified samples can be collated and the overall success of the multistage classification scheme can be determined. How the experimental design is affected by the data depends on the amount of overlap between different classes. Similar to techniques used for optimal feature selection, search techniques can be employed to find the best possible decision tree for multi-stage classification process. An exhaustive approach would involve finding all possible combinations with which decisions are made. This is certainly not optimal and would waste a lot of time. A better approach can be adopted by considering class distribution distances from each other. It is common sense to separate out those classes first that have the maximal average distance across all features from other classes. On data plots, these classes should appear as separate and isolated from data of other classes. In fact, using Bhattacharya or Chernoff distance, we can rank classes as to how well they are separated from others. Using such a procedure, we can obtain only one optimal decision tree.

In our study, we have used the first described strategy for classification based on colour information. Colour processing has not been used but it has been assumed that using a simple strategy, vegetation samples can be pooled separately from other samples and therefore we can develop two classifiers and as such two separate classification studies. The first study will be based on classifying vegetation data alone and the other study will be based on classifying natural object data. We next present our results on these two data sets individually in the next two chapters: first on vegetation data and then on natural objects.

Chapter 7

Vegetation data analysis

The main purpose of this chapter is to detail the experiments performed on the PANN scene analysis database. PANN scene analysis database has been developed to provide a benchmark data set that can be used by researchers to test their natural object recognition schemes. As with MeasTex and VisTex data sets, the results on texture recognition of different objects have been generated using the linear classifier and the k -nearest neighbour method. In the experiments with PANN database however, we have a lot more data sets. Since we have used four segmentation methods and five feature sets, we have a total of 20 data sets. The analysis of these data sets has been carried out using the leave-one-out cross-validation procedure. The chapter is organised as followed. We discuss the results obtained on the vegetation analysis of the PANN database categorised as per each segmentation method followed by texture analysis. The overall best results for these are summarised at the end to draw some meaningful conclusions. We discuss the significance of these findings in the context of the differences observed on recognition rates using different combinations of segmentation and feature extraction procedures. These differences are present as each segmentation procedure is sub-optimal to a certain extent in its definition of object regions. Our study is to evaluate how the differences in segmentation can affect the quality of features for classification, i.e. poor segmentation will yield poor object regions and thus poor texture features from such regions. Obviously some feature extraction techniques would be more robust to the sub-optimal segmentation process. The quality of the features themselves is difficult to evaluate without teaching a classifier to differentiate between samples of different classes present in the images, and thus the performance evaluation of the experimental trials is based on percentage average recognition rates. The chapter concludes with some salient observations on our analysis.

7.1 Analysis of vegetation data

We first analyse vegetation data containing three classes namely trees, grass and leaves. Our results are based on a total of 20 data sets. The discussion is organised around four sections, each based on a different segmentation method. As discussed earlier, the quality of image segmentation is directly related to the final results obtained. A method that over-segments will generate texture features from a smaller region area, whereas a method that under-segments will have the texture feature contaminated by pixels of two or more regions. In our analysis we use the linear classifier and nearest neighbour classifier. As before, linear classification is discussed first followed by k nearest neighbour classification.

7.2 Linear classification

We discuss the results first for FCM clustering, followed by, in order, histogram thresholding, region growing and split and merge segmentation methods. All results have been generated using the leave-one-out cross-validation procedure.

7.2.1 Fuzzy c-means clustering segmentation

For this analysis, all images have been analysed using FCM clustering segmentation. Results for each of the five texture extraction methods using the linear classifier are detailed below. Detailed confusion matrices are available in Appendix E.

Autocorrelation features

The PCA plot and canonical discriminant plot for autocorrelation features extracted after FCM segmentation are shown in Figure 7.1.

From these plots we can clearly see that tree and leaves samples have more variability compared to grass data. Also, tree and leaves overlap grass data. Tree samples appear to have more outliers than the two other classes. In Appendix E we find that the linear classifier using autocorrelation features generates a recognition success of 71.1% correct. The recognition accuracy on individual classes is trees (61.8%), grass (93.7%), and leaves (59.2%). Most of the mistakes are made when misclassified tree samples are labelled as grass, or when the leaves samples are mistaken as trees or grass.

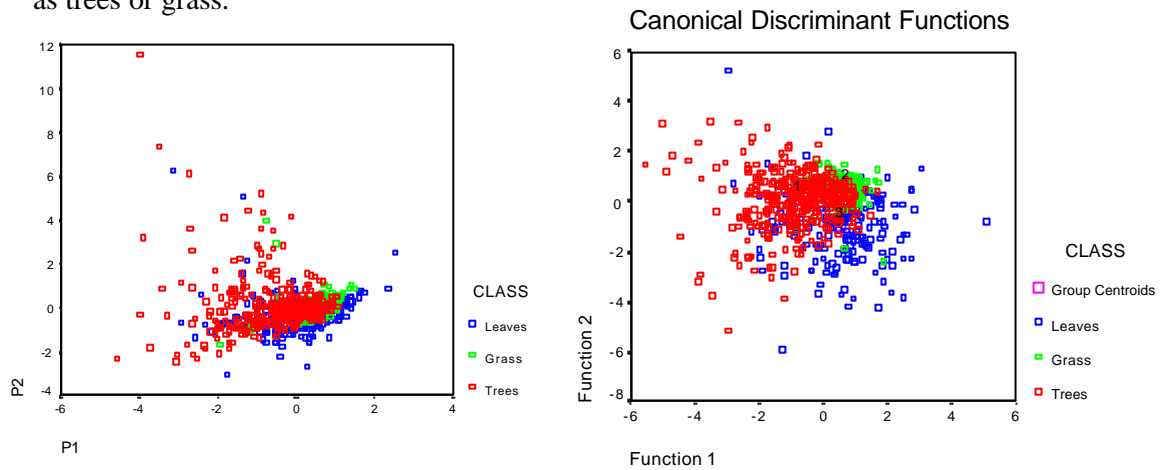


Figure 7.1 PCA plot and canonical discriminant function plot for the linear analysis using FCM and autocorrelation features for the discrimination of vegetation data.

Co-occurrence features

The PCA plot and canonical discriminant plot for co-occurrence features extracted after FCM segmentation are shown in Figure 7.2. As we can see, tree and leaves samples completely overlap. The separation however appears worse than Figure 7.1. As before, trees appear to have many more outliers compared to any other class. The PCA plot shows that using only the first two principal components, the samples are virtually indistinguishable. The confusion matrix in Appendix E shows an overall recognition rate of 57.3% correct. The best results are obtained for recognising grass and leaves, 70.5% and 72.5% correct recognition respectively. Trees are more difficult to recognise at 40.6% correct recognition. Trees are mostly misclassified as either leaves or grass. Most misclassified grass is classed as leaves and most misclassified leaves are classed as either trees or grass.

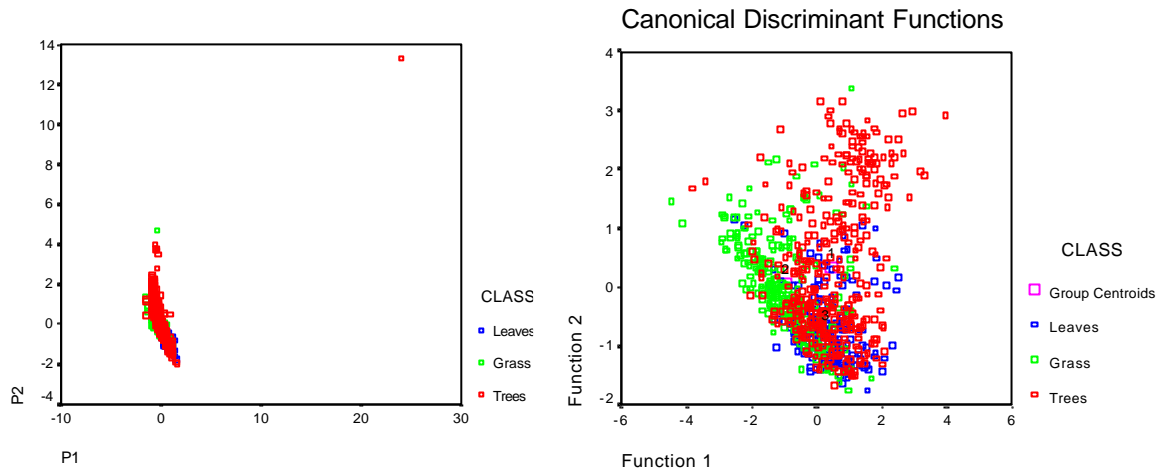


Figure 7.2 PCA plot and canonical discriminant function plot for the linear analysis using FCM and *co-occurrence* features for the discrimination of vegetation data.

Edge frequency features

The PCA plot and canonical discriminant plot for edge frequency features extracted after FCM segmentation are shown in Figure 7.3. The two plots show that trees and grass this time form a more compact cluster but there are a significant number of outliers for leaves. However since these outliers are not at, or across, class boundaries this should not affect the recognition of leaves samples. The confusion matrix in Appendix E shows the best recognition performance of 72.5%. The individual performances stand at trees (66.9%), grass (81.3%), and leaves (71.4%). Out of misclassified cases, the majority of samples from tree class are labelled as grass, most grass samples are labelled as trees, and most leaves are labelled as trees.

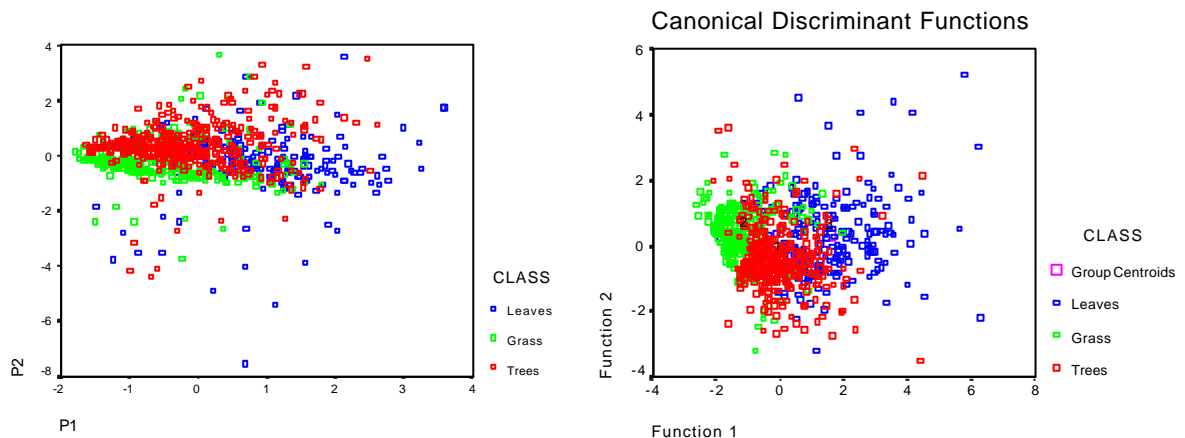


Figure 7.3 PCA plot and canonical discriminant function plot for the linear analysis using FCM and *edge frequency* features for the discrimination of vegetation data.

Law's features

The PCA plot and canonical discriminant plot for Law's features extracted after FCM segmentation are shown in Figure 7.4. There is considerable variability amongst the tree and grass samples. Tree samples virtually overlap most of the leaves samples in the plot. This situation is similar to the co-occurrence plots. The confusion matrix in Appendix E shows how the linear classifier performs on this feature set. An overall recognition rate of 62.6% is obtained with the best individual recognition performances for different classes as follows: trees (59.4%),

grass (54.1%), and leaves (79.6%). In terms of misclassifications, most misclassified patterns of class trees are labelled as leaves, most grass patterns are labelled as leaves, and most leaves patterns are labelled as trees.

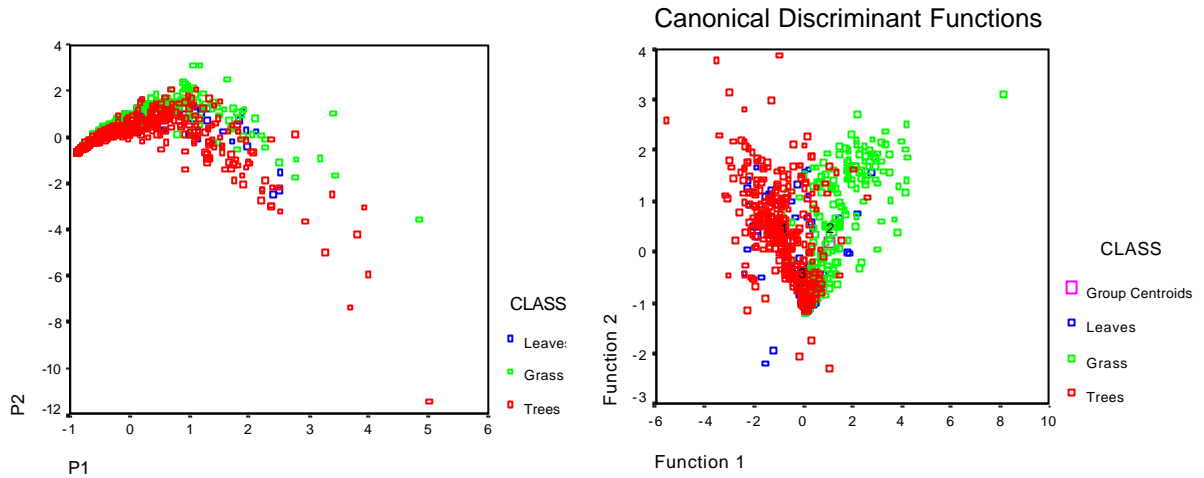


Figure 7.4 PCA plot and canonical discriminant function plot for the linear analysis using FCM and *Law's* features for the discrimination of vegetation data.

Primitive length features

These features are also called run length features. The PCA plot for primitive length features extracted after FCM segmentation are shown in Figure 7.5. The analysis fails to yield a canonical discriminant plot. Figure 7.5 shows little in terms of how data is distributed as all three classes lie linearly on top of each other. There is more variability along the second principal component axis. From Appendix E we see that the best overall performance of 36.7% correct recognition is obtained using linear analysis. The individual recognition rates are: trees (31.0%), grass (61.6%), and leaves (15%). Most mistakes are made when trees are misclassified as leaves, grass is misclassified as trees, and leaves are misclassified as trees and grass.

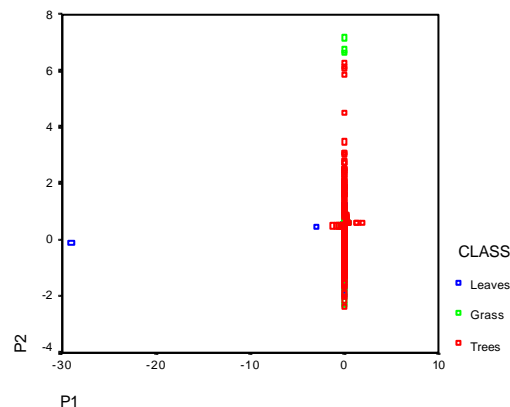


Figure 7.5 PCA plot for the linear analysis using FCM and *primitive length* features for the discrimination of vegetation data.

Summary

We find that on average the different feature sets perform relatively well except for primitive length method in discriminating between different vegetation classes. In Table 7.1 we summarise the overall results and highlight which class was best recognised for each feature set.

We find that tree samples are never the easiest to recognise. Also, three out of five times, grass is the easiest to recognise, and leaves is the best recognised class the other two times. The best result of 72.5% correct is obtained by the edge frequency method.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	71.1%	Grass (93.7%)
Co-occurrence	57.3%	Leaves (72.5%)
Edge frequency	72.5%	Grass (81.3%)
Law's	62.6%	Leaves (79.6%)
Primitive length	36.7%	Grass (61.6%)

Table 7.1 Summary of linear classifier performance with FCM.

Another manner in which the results can be summarised is to consider where the classifier makes most of the mistakes. This information present in confusion matrices is not easily visually interpreted. In Table 7.2 we show a better presentation of these mistakes. For each method of feature extraction, if the class in row *M* gets misclassified as class in row *N*, by more than 10% then we put one asterisk in that matrix position. We put two asterisks for more than 25% misclassification, and three asterisks for more than 50% misclassification.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		**	*
	<i>Grass</i>			
	<i>Leaves</i>	*	*	
<i>Co-occurrence</i>	<i>Trees</i>		*	**
	<i>Grass</i>			*
	<i>Leaves</i>	*	*	
<i>Edge frequency</i>	<i>Trees</i>		*	*
	<i>Grass</i>	*		
	<i>Leaves</i>	*		
<i>Law's</i>	<i>Trees</i>			**
	<i>Grass</i>			**
	<i>Leaves</i>	*		
<i>Primitive length</i>	<i>Trees</i>		***	*
	<i>Grass</i>	**		
	<i>Leaves</i>	***	**	

Table 7.2 Description of mistakes made by the linear classifier with FCM.

Table 7.2 shows that for most methods, trees are more prone to be misclassified as grass than vice-versa. Trees and leaves are misclassified almost interchangeably. Only a few times grass is misclassified as leaves and vice-versa. Also grass is more likely to be classified as leaves than vice-versa. Leaves and tree get confused as each other quite often. The number of mistakes made by the primitive length method is the largest with considerable overlap between trees and grass.

7.2.2 Histogram thresholding segmentation

For this analysis, all images have been analysed using histogram thresholding based segmentation. Results for each of the five texture extraction methods using the linear classifier are detailed below. Detailed confusion matrices are available in Appendix G.

Autocorrelation features

The PCA plot and canonical discriminant plot for autocorrelation features extracted after histogram based segmentation are shown in Figure 7.6. There appears to be a considerable amount of overlap across trees and grass data. Leaves samples are quite variable. Confusion matrices in Appendix G show an overall recognition rate of 64.3% correct with the individual classes recognised with the following accuracy: trees (48.7%), grass (91.7%), and leaves (54.9%). In terms of misclassifications, most of the tree samples are misclassified as leaves or grass, and leaves samples are mistaken as grass or trees.

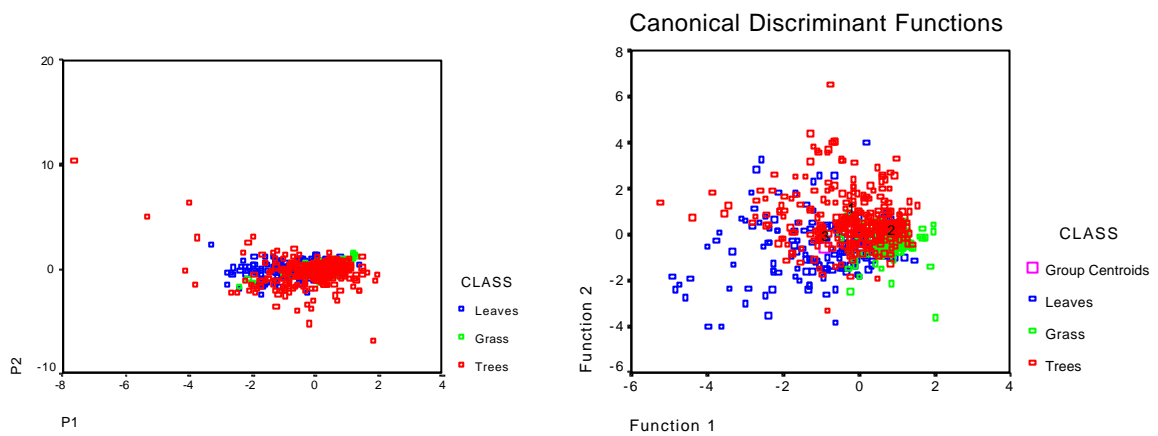


Figure 7.6 PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and *autocorrelation* features for the discrimination of vegetation data.

Co-occurrence features

The PCA plot and canonical discriminant plot for co-occurrence features extracted after histogram based segmentation are shown in Figure 7.7. There appears to be a considerable amount of overlap across trees and leaves data.

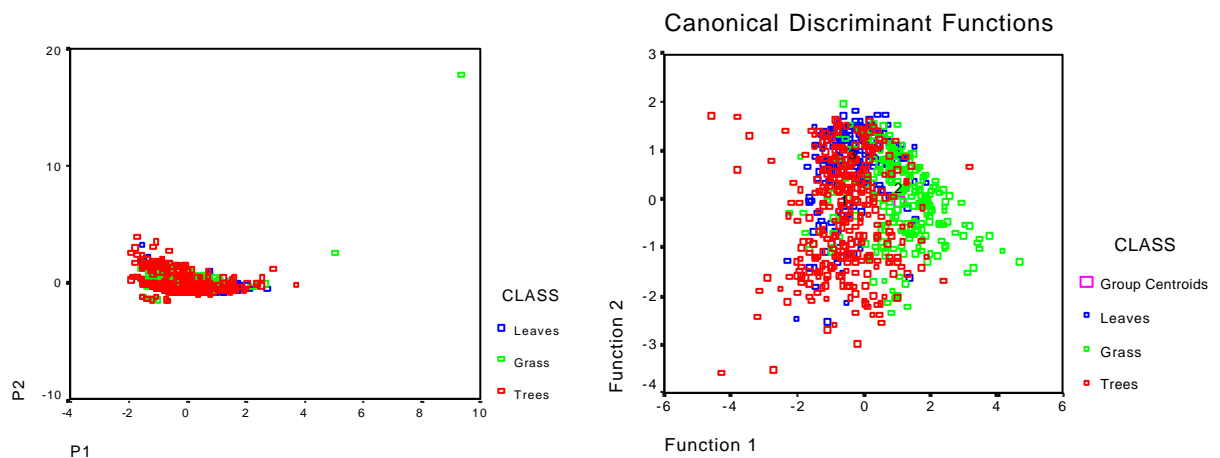


Figure 7.7 PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and *co-occurrence* features for the discrimination of vegetation data.

The confusion matrix in Appendix G shows the overall recognition result of 62.8% correct using leave-one-out cross-validation with the individual classes recognised with the following accuracy: trees (50%), grass (74%), and leaves (70.1%). For mistaken samples, the majority of mistakes can be attributed to tree samples being misclassified as leaves, grass samples mistaken as leaves, and leaves samples mistaken as grass and trees.

Edge frequency features

The PCA plot and canonical discriminant plot for edge frequency features extracted after histogram based segmentation are shown in Figure 7.8. The plots show a much better distinction between the three clusters with some of the tree samples overlapping with leaves samples. Tree and grass clusters appear compact in both of the plots.

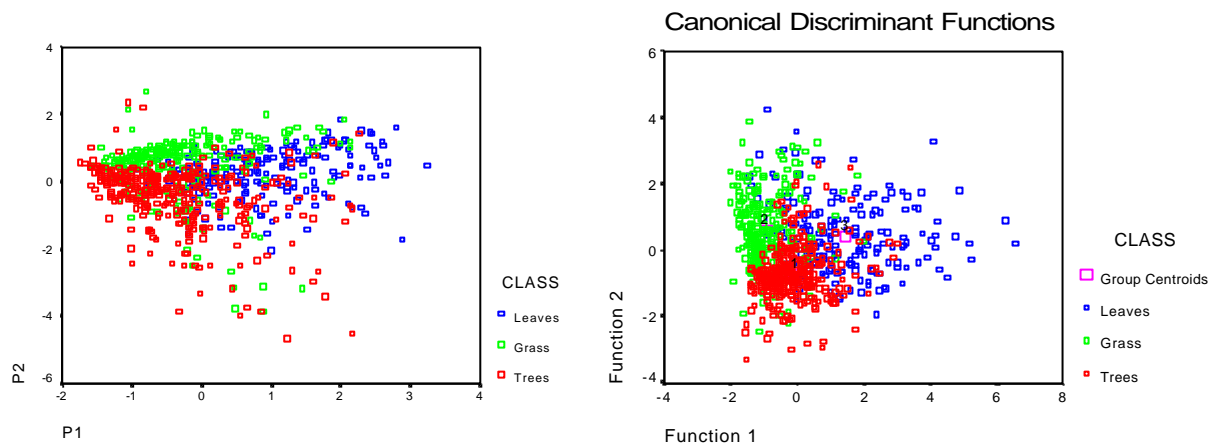


Figure 7.8 PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and *edge frequency* features for the discrimination of vegetation data.

The confusion matrices in Appendix G show an overall recognition accuracy of 76.5%. This is a very good result considering the similarity in the texture of various vegetation classes. The individual classes are recognised with the following accuracy: trees (81.2%), grass (78.0%), and leaves (66.3%). For misclassifications, grass samples have been misclassified as trees, and leaves have been misclassified as grass or trees.

Law's features

The PCA plot and canonical discriminant plot for Law's features extracted after histogram based segmentation are shown in Figure 7.9.

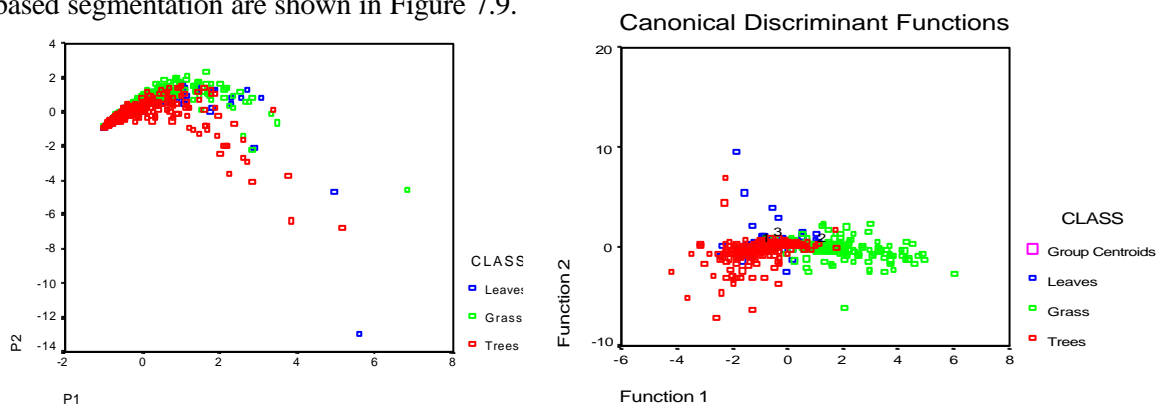


Figure 7.9 PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and *Law's features* for the discrimination of vegetation data.

The plots show a complete overlap of tree and leaves data. As a result of this, we can not expect the linear classifier to perform too well. The confusion matrix in Appendix G shows an overall leave-one-out recognition rate of 59.8% correct using leave-one-out cross-validation. The individual classes are recognised with the following accuracy: trees (49.7%), grass (63.5%), and leaves (72.3%). The misclassifications are mostly in cases of trees recognised as leaves, grass recognised as leaves, and leaves recognised as trees.

Primitive length features

The PCA plot for primitive length features extracted after histogram based segmentation are shown in Figure 7.10. The analysis fails to yield a canonical discriminant plot. Unfortunately there is not enough information from the plot except for the fact that the three classes are virtually impossible to classify on the basis of the two principal components alone.

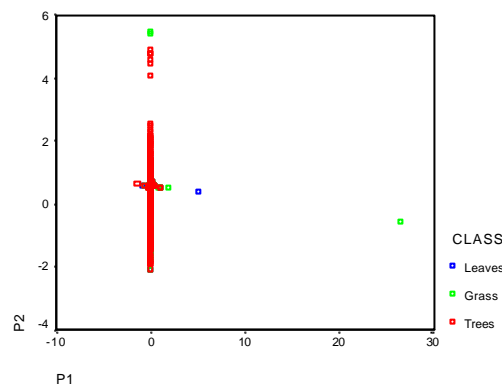


Figure 7.10 PCA plot for the linear analysis using Histogram Thresholding and *primitive length* features for the discrimination of vegetation data.

The confusion matrix in Appendix G shows an overall recognition rate of 46.1% correct. The individual classes are recognised with the following accuracy: trees (55.1%), grass (53.1%), and leaves (21.7%). The major misclassifications occur when trees are mistaken as grass, grass is mistaken as trees, and leaves are mistaken as both grass and trees.

Summary

In Table 7.3 we summarise the overall results and highlight which class was best recognised for each feature set. On the whole, the best performance is achieved using the edge frequency features. Different classes are best recognised using different texture methods.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	64.3%	Grass (91.7%)
Co-occurrence	62.8%	Grass (74.0%)
Edge frequency	76.5%	Trees (81.2%)
Law's	59.8%	Leaves (72.3%)
Primitive length	46.1%	Trees (55.1%)

Table 7.3 Summary of linear classifier performance with Histogram Thresholding.

Table 7.4 appears as described earlier. We find that in general there is a larger likelihood that tree samples are misclassified as grass rather than vice-versa. It is common for leaves samples to be mistaken as trees and vice versa.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		**	*
	<i>Grass</i>			
	<i>Leaves</i>	*	*	
<i>Co-occurrence</i>	<i>Trees</i>		*	**
	<i>Grass</i>	*		*
	<i>Leaves</i>	*	*	
<i>Edge frequency</i>	<i>Trees</i>			
	<i>Grass</i>	*		
	<i>Leaves</i>	*	*	
<i>Law's</i>	<i>Trees</i>			**
	<i>Grass</i>			**
	<i>Leaves</i>	**		
<i>Primitive length</i>	<i>Trees</i>		**	*
	<i>Grass</i>	**		
	<i>Leaves</i>	**	**	

Table 7.4 Description of mistakes made by the linear classifier with Histogram Thresholding.

7.2.3 Region growing segmentation

For this analysis, all images have been analysed using region growing segmentation. Results for each of the five texture extraction methods using the linear classifier are detailed below. Detailed confusion matrices are available in Appendix I.

Autocorrelation features

The PCA plot and canonical discriminant plot for autocorrelation features extracted after region growing segmentation are shown in Figure 7.11.

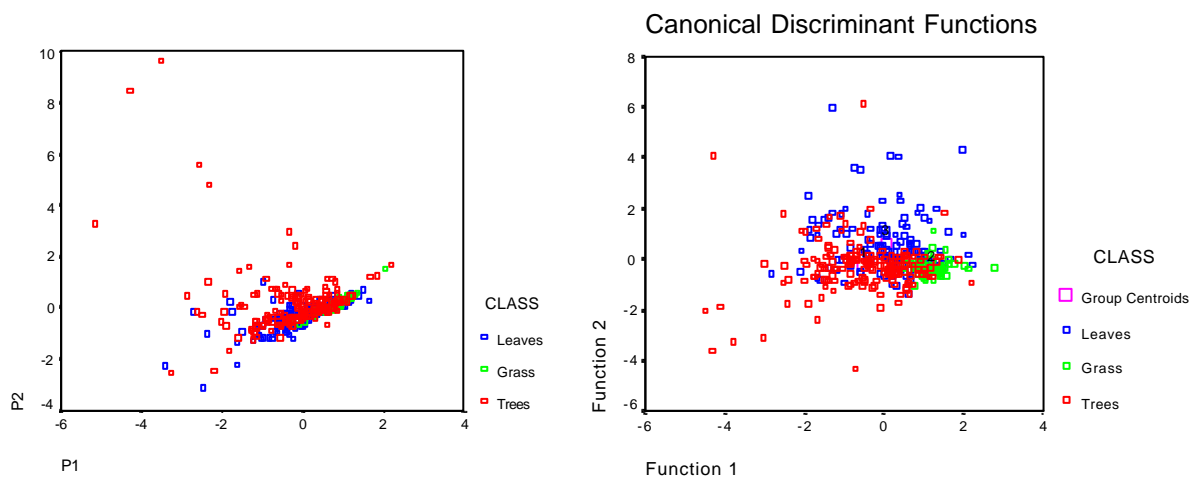


Figure 7.11 PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and *autocorrelation* features for the discrimination of vegetation data.

The plots show that there is considerable overlap across all three categories. An overall best result of 66.8% is shown in Appendix I. Trees are recognised with 63.9% accuracy, grass with 98.6% accuracy and leaves with 54.5% accuracy. The tree samples have been misclassified as grass or leaves, and leaves samples have been mistaken as trees and grass.

Co-occurrence features

The PCA plot and canonical discriminant plot for co-occurrence features extracted after region growing segmentation are shown in Figure 7.12. Co-occurrence features form elongated clusters as shown in Figure 7.12. With the high degree of overlap, the linear classifier can not be expected to perform too well. Appendix I shows the overall recognition rate of 54.3% correct with individual accuracy as follows: trees (50.3%), grass (52.0%), and leaves (60.6%). Each class has been roughly misclassified as something else with an error of around 20%.

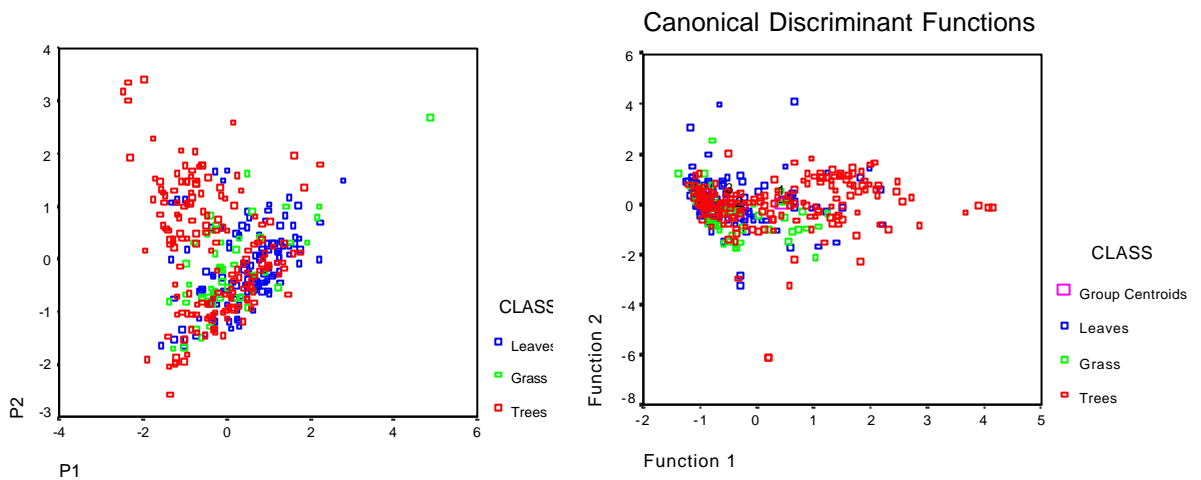


Figure 7.12 PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and *co-occurrence* features for the discrimination of vegetation data.

Edge frequency features

The PCA plot and canonical discriminant plot for edge frequency features extracted after region growing segmentation are shown in Figure 7.13.

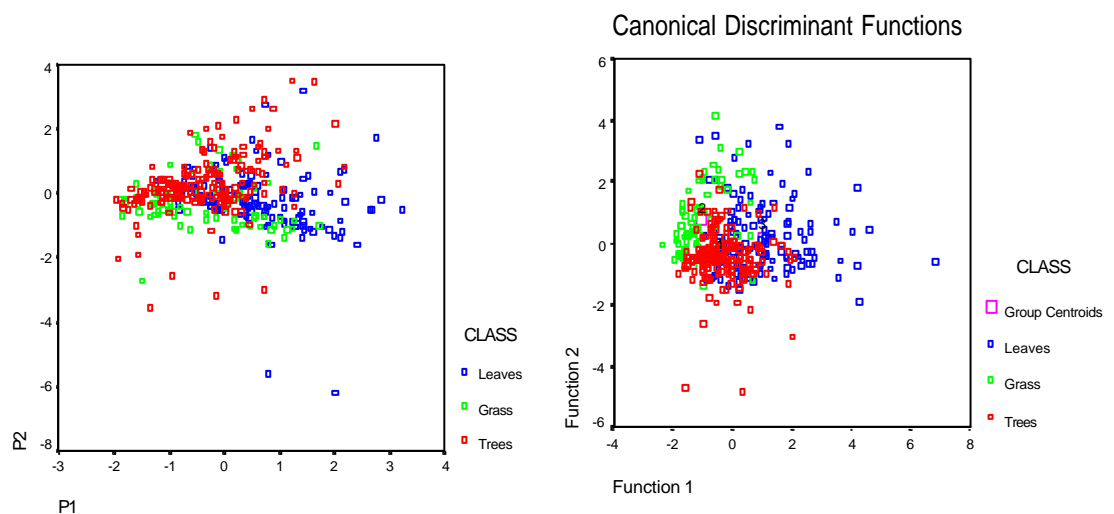


Figure 7.13 PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and *edge frequency* features for the discrimination of vegetation data.

In Figure 7.13, tree samples appear as a homogeneous cluster, whereas leaves are the most variable. An overall recognition accuracy of 69.5% is obtained using leave-one-out cross-validation. The confusion matrix in Appendix I shows individual accuracy of trees (74.4%), grass (73.9%), and leaves (60.4%). Tree samples are confused as grass or leaves with roughly an equal error rate of just over 10%. Leaves are confused as trees, and grass is confused as trees.

Law's features

The PCA plot and canonical discriminant plot for Law's features extracted after region growing segmentation are shown in Figure 7.14. The plots in Figure 7.14 show a significant degree of overlap between grass and leaves, and trees and leaves. The boomerang shaped data has trees on one end and grass at the other. In Appendix I, the confusion matrix shows an overall recognition rate of 61.6% with individual accuracy as follows: trees (55.0%), grass (56.5%), and leaves (73.1%). None of the tree samples are confused as grass but roughly half of them are confused as leaves. Similarly, hardly any grass samples are confused as trees and nearly half are confused as leaves. Leaves samples are only confused as trees.

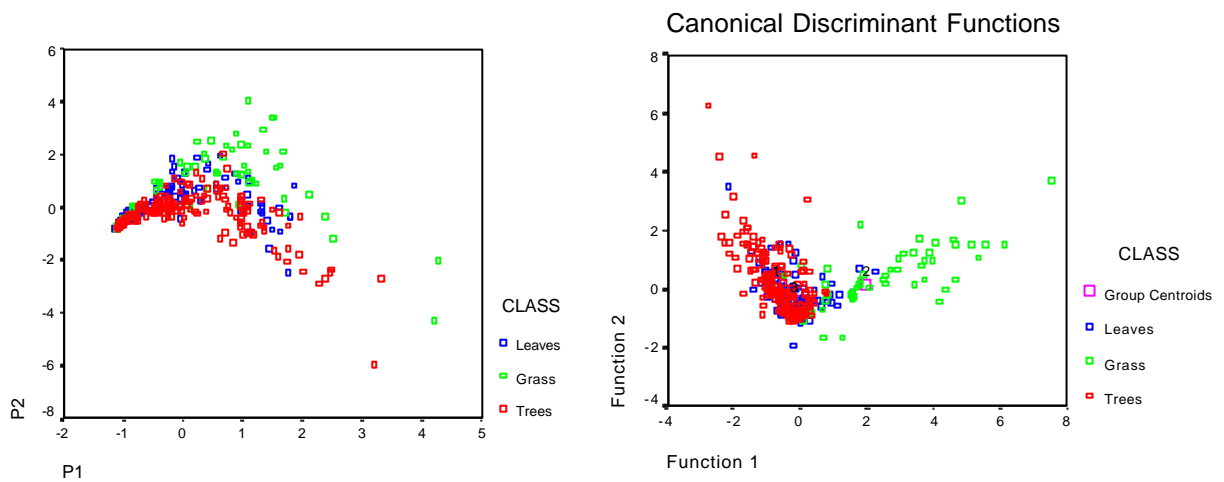


Figure 7.14 PCA plot and canonical discriminant function plot for the linear analysis using Region Growing and *Law's* features for the discrimination of vegetation data.

Primitive length features

The PCA plot for primitive length features extracted after histogram based segmentation are shown in Figure 7.15.

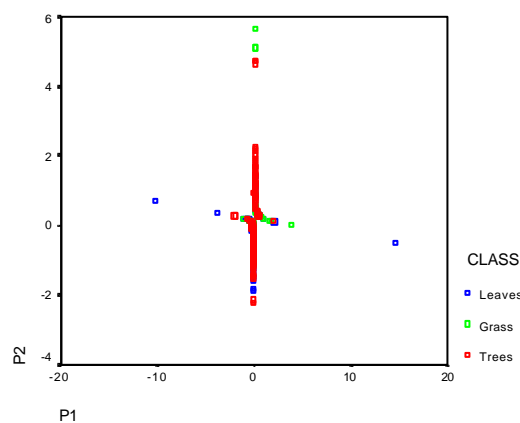


Figure 7.15 PCA plot for the linear analysis using Region Growing and *primitive length* features for the discrimination of vegetation data.

The analysis fails to yield a canonical discriminant plot. The confusion matrix in Appendix I shows an overall recognition rate of 31.3%. The recognition performances for trees and grass are abysmal at 8.9% and 14.5% respectively. Only leaves class achieves a reasonable performance of 70.1%. Tree samples are confused mostly as grass and leaves, and grass samples are confused as trees and leaves. The majority of mistakes made on leaves samples are in favour of trees.

Summary

In Table 7.5 we summarise the overall results and highlight which class was best recognised. In Table 7.6 we summarise the mistakes made by the linear classifier.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	66.8%	Grass (98.6%)
Co-occurrence	54.3%	Leaves (60.6%)
Edge frequency	69.5%	Trees (74.4%)
Law's	61.6%	Leaves (73.1%)
Primitive length	31.3%	Leaves (70.1%)

Table 7.5 Summary of linear classifier performance with Region Growing.

The best performance is achieved by the edge frequency method of texture extraction. In three out of five methods, leaves are the easiest to recognise. The ability of the autocorrelation method to recognise grass with complete accuracy is very impressive. We next discuss the mistakes made by the classifier in Table 7.6 that has been constructed as explained earlier. It appears that trees are more likely to be confused as leaves than grass. Also grass is more likely to be confused as leaves rather than trees. Leaves on the other hand are equally likely to be confused as either trees or grass.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		*	*
	<i>Grass</i>			
	<i>Leaves</i>	*	*	
<i>Co-occurrence</i>	<i>Trees</i>		*	**
	<i>Grass</i>	*		**
	<i>Leaves</i>	*	*	
<i>Edge frequency</i>	<i>Trees</i>		*	*
	<i>Grass</i>	*		
	<i>Leaves</i>	**	*	
<i>Law's</i>	<i>Trees</i>			**
	<i>Grass</i>			**
	<i>Leaves</i>	*		
<i>Primitive length</i>	<i>Trees</i>		**	**
	<i>Grass</i>	**		**
	<i>Leaves</i>	*		

Table 7.6 Description of mistakes made by the linear classifier with Region Growing.

7.2.4 Split and merge segmentation

For this analysis, all images have been analysed using split and merge based segmentation. Results for each of the five texture extraction methods using the linear classifier are detailed below. Detailed confusion matrices are available in Appendix K.

Autocorrelation features

The PCA plot and canonical discriminant plot for autocorrelation features extracted after split and merge segmentation are shown in Figure 7.16.

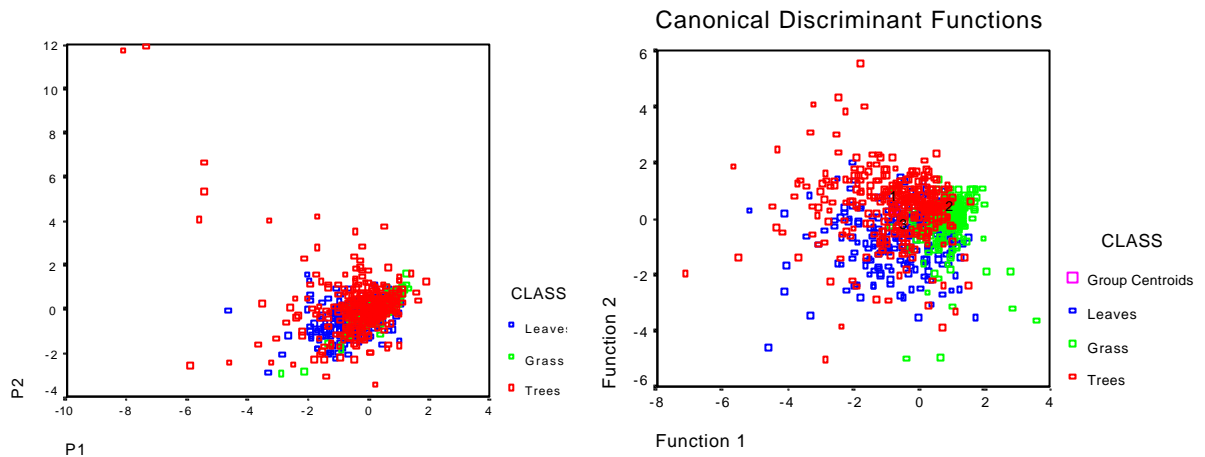


Figure 7.16 PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge and *autocorrelation* features for the discrimination of vegetation data.

The plots show a considerable number of outliers for all three classes. Grass forms the most compact cluster followed by tree data. Appendix K shows an overall recognition performance of 68.4% correct with the individual classes recognised with the following accuracy: trees (56.3%), grass (91.3%), and leaves (48.8%). It appears that this method is well suited to recognising grass texture. In terms of the classifier mistakes, these are made when mostly tree samples are with an equal likelihood assigned to grass or leaves, or when leaves are confused as trees or grass.

Co-occurrence features

The PCA plot and canonical discriminant plot for co-occurrence features extracted after split and merge segmentation are shown in Figure 7.17.

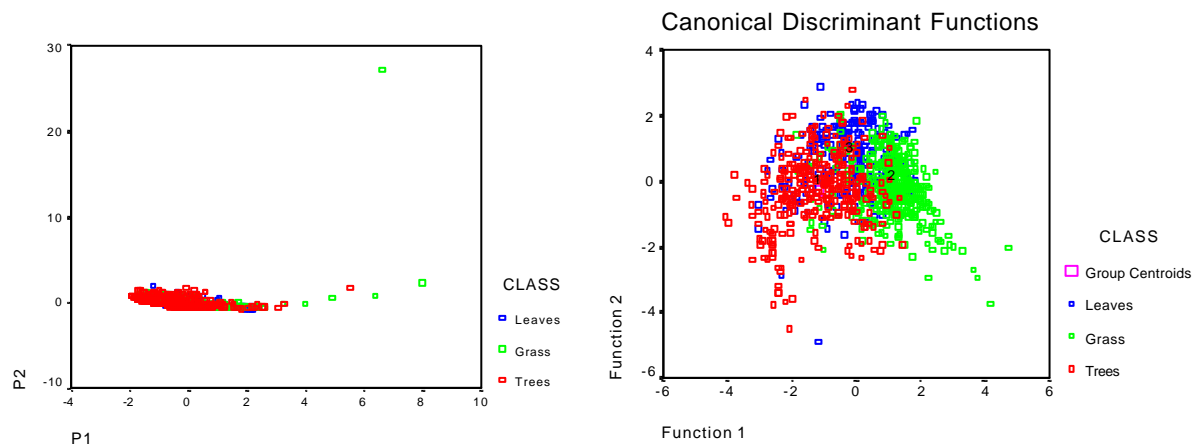


Figure 7.17 PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge and *co-occurrence* features for the discrimination of vegetation data.

In the above plot, the principal components appear totally overlapped and as such by themselves they will not yield good quality classification. Using the complete data, the canonical discriminant function plot shows a characteristic boomerang shape where tree and grass clusters appear disjoint but have considerable overlaps with leaves data. The results in Appendix K show an overall recognition accuracy of 69.8% with individual classes recognised as follows: trees (66.1%), grass (81.9%), and leaves (56.0%). The data plots justify the low recognition rate for leaves. In terms of mistakes, the tree samples are mostly mistaken as leaves, and leaves as both grass and trees.

Edge frequency features

The PCA plot and canonical discriminant plot for edge frequency features extracted after split and merge segmentation are shown in Figure 7.18.

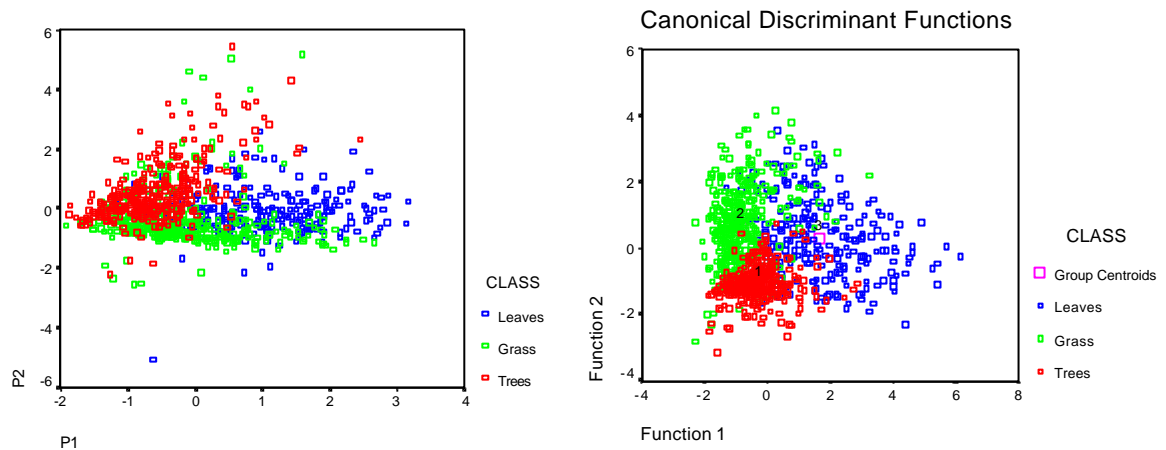


Figure 7.18 PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge segmentation and *edge frequency* features for the discrimination of vegetation data.

The above plot is the best seen so far. Trees, grass and leaves appear as three separate clusters and as such a good recognition performance can be expected using the linear classifier. This is confirmed by the very good accuracy on the data set of 79.4% correct as shown in Appendix K. The tree samples are the easiest to recognise (90.8% accuracy), something we have not seen in the previous analysis. Grass can be recognised with an accuracy of 74.1% and leaves with an accuracy of 73.0%. In terms of mistakes, grass is mostly confused as trees, and leaves as both grass and trees.

Law's features

The PCA plot and canonical discriminant plot for Law's features extracted after split and merge segmentation are shown in Figure 7.19. In Figure 7.19 we see a complete overlap of leaves data by the other two classes. All classes appear to have significant number of outliers. Appendix K shows a best recognition rate of 67.6% correct. The individual recognition accuracy of different classes are: trees (65.3%), grass (60.3%), and leaves (81.5%). In terms of mistakes made, the tree samples are mistaken as leaves, and grass samples are mistaken as leaves.

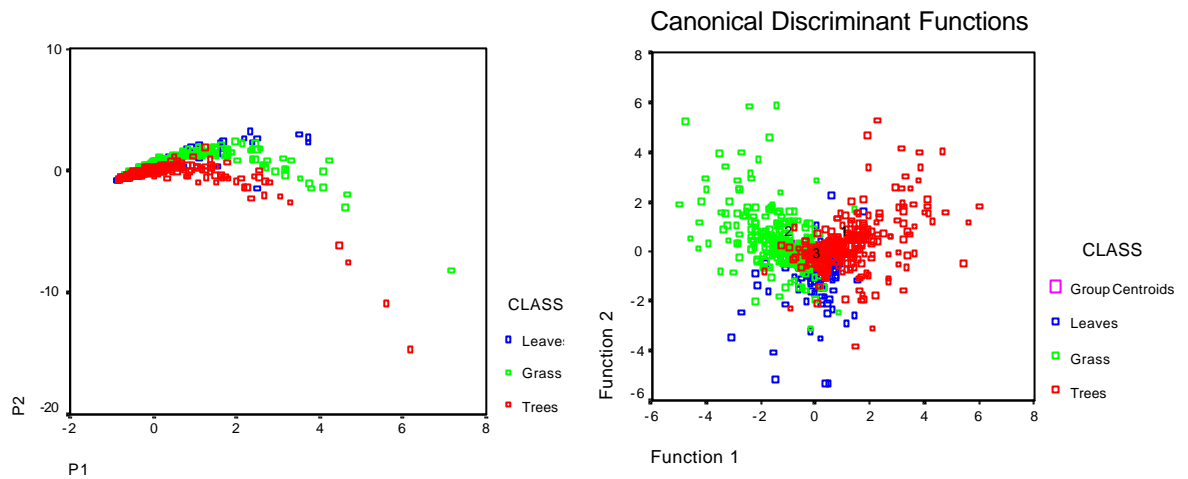


Figure 7.19 PCA plot and canonical discriminant function plot for the linear analysis using Split and Merge and *Law's* features for the discrimination of vegetation data.

Primitive length features

The PCA plot for primitive length features extracted after split and merge based segmentation are shown in Figure 7.20. The analysis fails to yield a canonical discriminant plot.

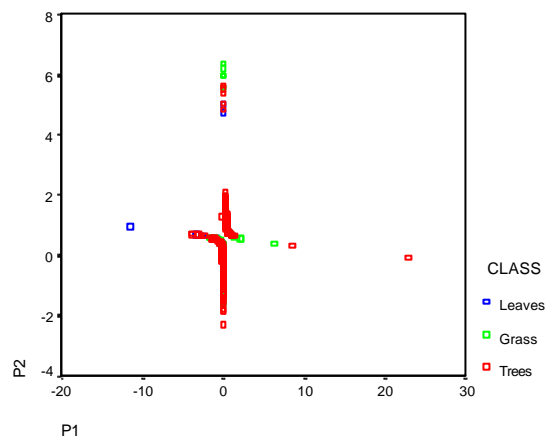


Figure 7.20 PCA plot for the linear analysis using Split and Merge and *primitive length* features for the discrimination of vegetation data.

There is very little discriminatory information available from either of the two principal components. The results in Appendix K show an overall leave-one-out recognition performance of 41.1%. The percentage accuracy on individual classes stands at: trees (55.7%), grass (17.4%), and leaves (58.9%). Tree samples have been mostly confused as leaves, grass has been confused as trees or leaves and leaves as grass and trees.

Summary

In Table 7.7 we summarise the overall linear classification results and highlight which class was best recognised. In Table 7.8 we summarise the mistakes made by the linear classifier. The best performance is achieved by the edge frequency method of texture extraction. In two out of five methods, leaves are the easiest to recognise, and twice grass is the easiest to recognise. The ability of the autocorrelation method to recognise grass with very high accuracy, and the same in the case of edge frequency for recognising trees, is very impressive. We next discuss the mistakes made by the classifier in Table 7.8 that has been constructed as explained earlier.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	68.4%	Grass (91.3%)
Co-occurrence	69.8%	Grass (81.9%)
Edge frequency	79.4%	Trees (90.8%)
Law's	67.6%	Leaves (81.5%)
Primitive length	41.1%	Leaves (58.9%)

Table 7.7 Summary of linear classifier performance with Split and Merge.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		*	*
	<i>Grass</i>			
	<i>Leaves</i>	*	**	
<i>Co-occurrence</i>	<i>Trees</i>		*	*
	<i>Grass</i>			*
	<i>Leaves</i>	*	*	
<i>Edge frequency</i>	<i>Trees</i>			
	<i>Grass</i>	*		
	<i>Leaves</i>	*	*	
<i>Law's</i>	<i>Trees</i>			**
	<i>Grass</i>			**
	<i>Leaves</i>			
<i>Primitive length</i>	<i>Trees</i>		*	**
	<i>Grass</i>	**		***
	<i>Leaves</i>	*	*	

Table 7.8 Description of mistakes made by the linear classifier with Split and Merge.

In Table 7.8 we find that for most methods, trees get misclassified as leaves and leaves get misclassified as trees. There is much less confusion between grass and trees. In the first four methods, grass hardly gets misclassified as leaves but this mistake is significant for the last two methods. Leaves are on the other hand more prone to be mistaken as grass.

7.2.5 Summarising linear classification results on vegetation data

We have already presented the summary of various analyses using the linear classifier. However, the summaries have been presented by grouping different feature extraction methods under a segmentation scheme. We can actually summarise the classifier mistakes in a reverse manner, i.e. how does the classifier make mistakes keeping the same feature extraction method based on the output of different segmentation methods. In this section we aim to show some tables with this information. Also, we present a final table showing the classification results of different segmentation and feature extraction combinations at one place. Let us first summarise how the mistakes are made by the linear classifier for each feature extraction method by changing the preceding segmentation process. For a total of 5 feature extraction methods, Tables 7.9-7.13 are drawn. Finally, Table 7.14 shows which combination of segmentation method with texture extraction method yields the best classification on vegetation data.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees		**	*
	Grass			
	Leaves	*	*	
Histogram	Trees		**	*
	Grass			
	Leaves	*	*	
Region Growing	Trees		*	*
	Grass			
	Leaves	*	*	
Split & Merge	Trees		*	*
	Grass			
	Leaves	*	*	

Table 7.9 Linear classifier mistakes for autocorrelation features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees		*	**
	Grass			*
	Leaves	*	*	
Histogram	Trees		*	**
	Grass			*
	Leaves	*	*	
Region Growing	Trees		*	**
	Grass	*		**
	Leaves	*	*	
Split & Merge	Trees		*	*
	Grass			*
	Leaves	*	*	

Table 7.10 Linear classifier mistakes for co-occurrence features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees		*	*
	Grass	*		
	Leaves	*		
Histogram	Trees			
	Grass	*		
	Leaves	*	*	
Region Growing	Trees		*	*
	Grass	*		
	Leaves	**	*	
Split & Merge	Trees			
	Grass	*		
	Leaves	*	*	

Table 7.11 Linear classifier mistakes for frequency features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees			**
	Grass			**
	Leaves	*		
Histogram	Trees			**
	Grass			**
	Leaves	*		
Region Growing	Trees			**
	Grass			**
	Leaves	*		
Split & Merge	Trees			**
	Grass			**
	Leaves			

Table 7.12 Linear classifier mistakes edge for Law's features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees		***	*
	Grass	**		
	Leaves	***	**	
Histogram	Trees		**	*
	Grass	**		
	Leaves	**	**	
Region Growing	Trees		**	***
	Grass	**		**
	Leaves	*		
Split & Merge	Trees		*	**
	Grass	**		***
	Leaves	*	*	

Table 7.13 Linear classifier mistakes for primitive length features.

In Tables 7.9-7.13, we find that for each feature extraction method, segmentation methods have small differences. Especially, FCM and histogram thresholding based segmentation performances are similar. Similarly, region growing, and split and merge performances are

similar. For autocorrelation features, most segmentation methods used show mistakes being made when trees are classed as grass. This mistake is made much more with FCM and histogram thresholding segmentation rather than with the other two techniques. Grass is less likely to be mistaken as anything else, but trees and leaves often get mistaken as grass. Leaves are often mistaken as grass or trees. In the case of co-occurrence features, we find that region growing segmentation performance is different from other techniques. The difference is present as grass gets misclassified as trees with region growing, and many more samples of grass are mistaken as leaves compared to other methods. On the whole, other classes get misclassified as leaves quite often. The other patterns are similar to autocorrelation features. In the case of edge frequency features, we find that region growing and FCM performances are similar, and histogram thresholding and split and merge performances are similar. The main difference between these two groups is that in the first group trees get misclassified as grass or leaves whereas in the second group this does not happen. For this feature set, we find that similar to co-occurrence features, grass samples are misclassified as others. This did not happen with FCM based segmentation. In the case of Law's features, all segmentation method results are nearly similar. There seems to be a major bias in errors in favour of classifying everything as leaves. Apart from this, the only other mistake is classifying leaves as trees. Finally, with primitive length features, we find that FCM and histogram based segmentation results are similar. Trees and grass are often mistaken as each other, and similarly trees and leaves are mistaken as each other. The main difference with the second group of segmentation techniques including region growing and split and merge is that grass is not mistaken as leaves as in the first group.

The overall performance of the classification scheme is shown in Table 7.14 where mean and standard deviations have been computed across rows and columns. This is also shown as a plot in Figure 7.21.

Feature Segmentation	Autocorrelation	Co-occurrence	Edge frequency	Law's	Primitive length	μ	σ
FCM	71.1%	57.3%	72.5%	62.6%	36.7%	60.0%	14.5%
Histogram	64.3%	62.8%	76.5%	59.8%	46.1%	61.9%	10.9%
Region Grow	66.8%	54.3%	69.5%	61.6%	31.3%	56.7%	15.3%
Split & Merge	68.4%	69.8%	79.4%	67.6%	41.1%	65.3%	14.3%
μ	67.6%	61.1%	74.5%	62.9%	38.8%	-	-
σ	2.8%	6.8%	4.3%	3.3%	6.3%	-	-

Table 7.14 The different classifier performance depending on which data set is used for vegetation analysis.

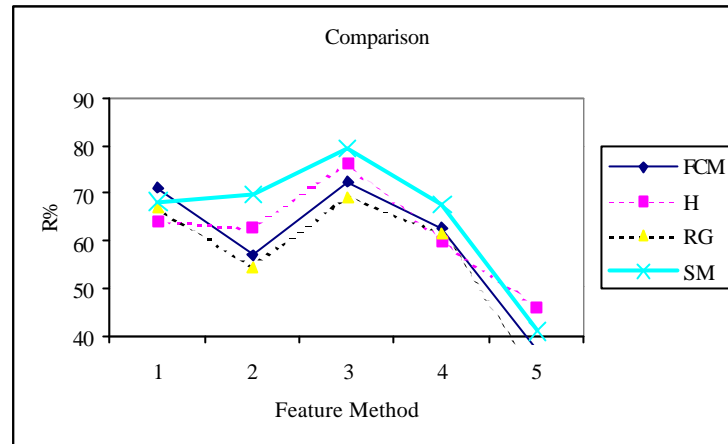


Figure 7.21 A graphical comparison of recognition accuracy obtained using different segmentation method and texture method combinations for vegetation data analysis using linear classifier.

The following conclusions can be drawn from Table 7.14 and Figure 7.21.

1. Edge frequency based texture extraction is the best for separating vegetation classes.
2. On the whole, split and merge segmentation technique is the best performer.
3. There is considerable variability in the following feature sets as the segmentation method is changed: co-occurrence, edge frequency, primitive length. The other feature extraction methods do not exhibit major performance changes.
4. There is considerable variability for all of the segmentation methods as the feature sets are changed. The segmentation methods can be ranked in order of decreasing variability of performance as follows: region growing, split and merge, FCM and histogram based segmentation. Similarly the feature extraction methods can be ranked in decreasing order of variability as follows: Co-occurrence, primitive length, edge frequency, Law's and autocorrelation. If we are to consider the least variable texture method as being the best, then autocorrelation is the best feature extraction method in our analysis.

7.3 Nearest neighbour classification

We discuss the results first for FCM clustering, followed by, in order, histogram thresholding, region growing and split and merge segmentation methods. The results have been produced using leave-one-out validation procedure.

7.3.1 Fuzzy c-means clustering segmentation

For this analysis, all images have been analysed using FCM clustering segmentation. Results for each of the five texture extraction methods using the k NN classifier are detailed below. Detailed confusion matrices are available in Appendix F.

Autocorrelation features

The best performance of 72.9% correct recognition is achieved using model-1 ($k=5$) which is slightly better than the linear analysis result of 71.1% correct. The confusion matrix shows that tree, samples have been misclassified as grass and leaves, grass samples have been mistaken as trees and leaves have been primarily mistaken as trees. The individual recognition accuracy is:

trees (73.9%), grass (80.6%), and leaves (47.0%). Hence, compared to the linear classification we are better at recognising trees but much poorer at recognising grass and leaves.

Co-occurrence features

The best performance of 56.5% correct is obtained for model1 ($k=7$). The mistakes are made when any given class sample is confused as something else. The error rate is favour of the two other classes is not much different. The individual accuracy is: trees (66.0%), grass (47.7%), and leaves (49.0%). Compared to the linear classification, the recognition performance of 56.5% is slightly poorer than 57.3% obtained with LDA. We find that the performance on recognising trees has increased by more than 25% but the performance on recognising leaves and grass has dropped by nearly 20%.

Edge frequency features

The best performance is achieved with model1 ($k=7$) of 72.0% correct recognition. This compares favourably with the linear classifier performance of 72.5% correct recognition. The confusion matrix shows that mistakes are made when significant amounts of grass samples are mistaken as trees and leaves samples are also mistaken as trees. The individual class recognition accuracy stands at: trees (85.7%), grass (64.2%), and leaves (56.3%). Compared to linear analysis, the nearest neighbour is significantly better at recognising trees but much worse on grass and leaves.

Law's features

The best recognition performance is achieved by model1 ($k=5$) of 74.6% correct which is significantly better than the linear classifier performance of 62.6%. The mistakes are made when tree samples are confused as grass, and when leaves are confused as grass of trees. The different classes are recognised with the following accuracy: trees (79.6%), grass (83.5%), and leaves (53.4%). Hence, compared to the linear classifier we get much better accuracy on recognising tree and grass samples but we are poor off at recognising the leaves data.

Primitive length features

The best performance of 57.6% correct is achieved using a single nearest neighbour. This is significantly better than the linear classification success of 36.7%. The mistakes are more or less equally distributed across different classes. The individual class accuracy are: trees (59.4%), grass (54.1%), and leaves (58.7%). Compared to linear analysis, we are better off at recognising tree samples, and significantly better at recognising leaves samples but slightly poorer at identifying grass samples.

Summary

On the whole we find that the k nearest neighbour classification improves the overall results but not significantly for the first three methods. The mistakes are redistributed across different class combinations. In almost all of the methods, the nearest neighbour classifier improves the recognition of tree samples. In Table 7.15 we present the summary of the nearest neighbour classifier performance using FCM segmentation and the five feature extraction methods.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	72.9%	Grass (80.6%)
Co-occurrence	50.7%	Trees (57.3%)
Edge frequency	72.0%	Trees (83.2%)
Law's	74.6%	Grass (83.5%)
Primitive length	57.6%	Trees (59.4%)

Table 7.15 Summary of nearest neighbour classifier performance with FCM.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. This information present in confusion matrices is not easily interpreted. In Table 7.16 we show a better presentation of these mistakes as we did for linear classifier. We restate the procedure for this. For each method of feature extraction, if the class in row M gets misclassified as object in row N , by more then 10% then we put one asterisk in that matrix position. We put two asterisks for more than 25% misclassification, and three asterisks for more than 50% misclassification.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		*	
	<i>Grass</i>	*		
	<i>Leaves</i>	**	*	
<i>Co-occurrence</i>	<i>Trees</i>		*	*
	<i>Grass</i>	**		*
	<i>Leaves</i>	**	*	
<i>Edge frequency</i>	<i>Trees</i>			
	<i>Grass</i>	**		
	<i>Leaves</i>	**		
<i>Law's</i>	<i>Trees</i>		*	
	<i>Grass</i>	*		
	<i>Leaves</i>	**	*	
<i>Primitive length</i>	<i>Trees</i>		**	*
	<i>Grass</i>	**		*
	<i>Leaves</i>	**	*	

Table 7.16 Description of mistakes made by the nearest neighbour classifier with FCM.

7.3.2 Histogram thresholding based segmentation

For this analysis, all images have been analysed using histogram thresholding. Results for each of the five texture extraction methods using the k nearest neighbour classifier are detailed below. Detailed confusion matrices are available in Appendix H.

Autocorrelation features

The best performance using nearest neighbour classifier is achieved with mode11 ($k=5$). This model shows the recognition accuracy of 70.0% correct. This is better than the linear classifier performance of 64.3% correct on the same data. The individual class recognition accuracy are:

trees (63.6%), grass (85.1%), and leaves (56.5%). Thus the nearest neighbour classifier is much better at recognising trees, slightly better at recognising leaves, and slightly poorer at recognising grass samples. For the nearest neighbour classifier, most of the mistakes are made when tree samples are confused as grass or leaves, and when leaves are confused a tree samples.

Co-occurrence features

Using model-1 ($k=7$), the best recognition performance of 57.3% correct is achieved. This is worse than that of the linear classifier at 62.6% correct. The mistakes are made when tree samples are, with nearly equal likelihood, labelled as grass and leaves, and when grass and leaves are confused as trees. In terms of the individual class accuracy we get: trees (60.5%), grass (60.1%), and leaves (45.1%). Hence, compared to the linear method we are much better at recognising trees but significantly worse off at recognising grass or leaves.

Edge frequency features

The best performance is achieved using model-1 ($k=7$) with a recognition rate of 70.0%. This performance is not as good as the one achieved by the linear classifier of 76.5% correct. In particular, the mistakes are made when tree samples are misclassified as leaves, grass and leaves samples are confused as trees. The individual accuracy of recognising each class are: trees (77.4%), grass (58.1%), and leaves (72.8%). This performance shows that the nearest neighbour classifier is better than the linear classifier at recognising leaves but poorer in identifying the other two classes.

Law's features

The nearest neighbour yields the best recognition performance of 59.1% correct for model-1 ($k=7$). This is not much different than the linear classifier performance of 59.8% correct. In terms of individual class recognition, we get the following accuracy: trees (63.7%), grass (71.7%), and leaves (34.7%). If we compare these performances with the linear classifier, we find that the nearest neighbour model is superior on recognising trees and grass but significantly poorer on recognising leaves.

Primitive length features

We find that the best performance is achieved with a single nearest neighbour. This result of 55.6% correct recognition is superior than the linear classification result of 46.1% correct. The model makes errors when classifying grass and leaves samples. The individual class recognition is: trees (65.2%), grass (53.9%), and leaves (41.3%). Compared to the linear classifier, all classes can now be recognised with more accuracy, especially leaves.

Summary

On the whole we find that the k nearest neighbour classification improves the overall results except for edge frequency measures where we see a drop of nearly 6% and with Law's features where no significant difference is present. Otherwise, the results tend to improve by nearly 5% and for primitive length method the results get better by nearly 10%. The mistakes are redistributed across different class combinations. In Table 7.17 we present the summary of the nearest neighbour classifier performance using Histogram thresholding segmentation and the five feature extraction methods.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	70.0%	Grass (85.1%)
Co-occurrence	57.3%	Trees (60.5%)
Edge frequency	70.0%	Trees (77.4%)
Law's	59.1%	Grass (71.7%)
Primitive length	46.1%	Trees(65.2%)

Table 7.17 Summary of nearest neighbour classifier performance with Histogram Thresholding.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. In Table 7.18 we show a better presentation of these mistakes as we did for linear classifier.

It appears that trees are often mistaken as grass and grass is mistaken as trees. The number of mistakes is just over 10% in such cases. Also leaves have a larger likelihood of being misclassified as trees as opposed to vice-versa. There is also significant confusion between distinguishing leaves from grass. On the whole, all feature sets except the last one yield similar errors when using the nearest neighbour classifier.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		*	*
	<i>Grass</i>			
	<i>Leaves</i>	**	*	
<i>Co-occurrence</i>	<i>Trees</i>		*	*
	<i>Grass</i>	**		*
	<i>Leaves</i>	**	*	
<i>Edge frequency</i>	<i>Trees</i>			*
	<i>Grass</i>	*		*
	<i>Leaves</i>	*		
<i>Law's</i>	<i>Trees</i>		*	*
	<i>Grass</i>	*		*
	<i>Leaves</i>	**	*	
<i>Primitive length</i>	<i>Trees</i>		*	*
	<i>Grass</i>	**		*
	<i>Leaves</i>	**	**	

Table 7.18 Description of mistakes made by the nearest neighbour classifier with Histogram Thresholding.

7.3.3 Region growing based segmentation

For this analysis, all images have been analysed using region growing. Results for each of the five texture extraction methods using the *k*NN classifier are detailed below. Detailed confusion matrices are available in Appendix J.

Autocorrelation features

The best performance with these features is obtained using model-1 ($k=3$). The recognition rate of 58.2% is much less than the 66.8% correct classification obtained with the linear model. The majority of the mistakes are made when tree samples are confused as leaves, and vice-versa. The individual recognition accuracy for classes are: trees (67.2%), grass (73.9%), and leaves (38.0%). Compared to the linear classifier, all classes have been recognised with lower accuracy, with the drop in performance for grass and leaves being significant.

Co-occurrence features

The best recognition is achieved using model-1 ($k=3$). The recognition rate of 52.3% is not too different from the linear classifier performance of 54.3%. The individual classes are recognised with the following accuracy: trees (64.5%), grass (22%) and leaves (48.0%). Compared to the linear classifier we find that trees are easier to recognise but grass and leaves are less recognisable. The drop in recognition accuracy for grass is around 30% and for leaves around 12%. The nearest neighbour classifier makes majority of mistakes when tree samples are confused as leaves, grass samples are confused as tree and leaves, and leaves are confused as trees.

Edge frequency features

These features yield the best performance of 65.0% correct recognition when using model-1 ($k=7$). This performance is slightly inferior to the linear classifier performance of 69.5%. The majority of the mistakes are made when tree samples are confused as leaves, and vice-versa. The worst recognition is seen for grass. More than half of the grass patterns get misclassified as trees and nearly a quarter get misclassified as leaves. The individual recognition rates for the classes are: trees (80.5%), grass (24.6%), and leaves (64.9%). Compared to the linear classifier we are slightly better at recognising trees and leaves, but poorer by up to 50% at recognising grass samples.

Law's features

These features give a best performance of 64.0% correct recognition when using model-1 ($k=7$). We find that this compares well with the linear classification rate of 61.6%. The individual recognition rates for each class are: trees (72.2%), grass (59.4%), and leaves (55.2%). Compared to the linear classifier, we are roughly 20% better at recognising trees, slightly poorer at recognising grass and roughly 20% inferior at recognising leaves. For the nearest neighbour classifier, the majority of the mistakes are made when the tree samples are confused as leaves and vice-versa, and majority of grass samples is confused as leaves.

Primitive length features

The best performance of 56.7% correct recognition is achieved using model-1 ($k=7$). This is much better than the linear classifier performance of 31.3% correct. The individual classes get recognised with the following accuracy: trees (76.7%), grass (7.2%), and leaves (55.2%). The performance for grass recognition is abysmal. Compared to the linear classifier, we find that the nearest neighbour method is much better at recognising trees by nearly 65% more, grass recognition is worse off by 7% and leaves are worse off by nearly 15%. The majority of

classification mistakes are made when trees are misclassified as leaves and vice-versa, and when leaves are confused as leaves or trees.

Summary

On the whole we find that the k nearest neighbour classification is inferior to the linear method except for a couple of feature sets. There is however a considerable improvement with the primitive length method. On the whole, the k nearest neighbour classifier makes different kinds of mistakes than the linear classifier. In Table 7.19 we present the summary of the nearest neighbour classifier performance-using region growing segmentation and the five feature extraction methods. On the whole the best performer is the edge frequency method closely followed by the Law's method. Trees appear to be the easiest to recognise in most of these feature sets.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	58.2%	Grass (73.9%)
Co-occurrence	52.3%	Trees (64.5%)
Edge frequency	65.0%	Trees (80.5%)
Law's	64.0%	Trees (72.2%)
Primitive length	56.7%	Trees (76.7%)

Table 7.19 Summary of nearest neighbour classifier performance with Region Growing.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. In Table 7.20 we show a better presentation of these mistakes.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		*	*
	<i>Grass</i>	*		*
	<i>Leaves</i>	***	*	
<i>Co-occurrence</i>	<i>Trees</i>			**
	<i>Grass</i>	**		**
	<i>Leaves</i>	**	*	
<i>Edge frequency</i>	<i>Trees</i>			*
	<i>Grass</i>	***		*
	<i>Leaves</i>	**		
<i>Law's</i>	<i>Trees</i>			*
	<i>Grass</i>			*
	<i>Leaves</i>	**	*	
<i>Primitive length</i>	<i>Trees</i>			**
	<i>Grass</i>	***		**
	<i>Leaves</i>	**	*	

Table 7.20 Description of mistakes made by the nearest neighbour classifier with Region Growing.

The Table 7.20 shows that trees are more likely to be mistaken as leaves rather than grass. Also grass is quite often mistaken as trees except for Law's feature set. It also gets mistaken as leaves

with most feature sets. Leaves are, in considerable quantity, mistaken as trees but to a smaller degree as grass.

7.3.4 Split and merge segmentation

For this analysis, all images have been analysed using split and merge segmentation. Results for each of the five texture extraction methods using the k NN classifier are detailed below. Detailed confusion matrices are available in Appendix L.

Autocorrelation features

The best with this feature set is obtained with model-1 ($k=5$). The recognition rate of 64.2% compares favourably with the linear classifier performance of 68.4% correct. Mistakes are made when tree samples are confused as grass or leaves, and when leaves are confused as grass and trees. The individual recognition rates are: trees (51.8%), grass (87.6%) and leaves (43.9%). Compared to the linear classifier, the nearest neighbour method is poorer at recognising trees by roughly 5%, grass by 4%, and leaves by 5%.

Co-occurrence features

The best recognition performance of 66.2% is obtained with model-1 ($k=5$). This is inferior to the linear classifier performance by roughly 3%. The individual classes are recognised with the following accuracy: trees (69.0%), grass (77.1%), and leaves (46.0%). Roughly an equal number of samples of a given class are misclassified as the two other classes. Compared to the linear classifier, we find that trees are recognised roughly 3% better, grass recognition is about 5% poorer and leaves 10% poorer.

Edge frequency features

The best recognition performance is obtained for model-1 ($k=5$). The recognition accuracy of 74.0% correct is much less than the linear classifier success of 79.4%. Most of the misclassifications can be attributed to tree samples being confused as leaves and grass samples confused as trees. The average recognition rates for the classes are: trees (82.2%), grass (66.7%), and leaves (75%). Compared to the linear classifier, the tree samples are recognised poorer by 9%, grass poorer by 7%, and leaves better recognised by 2%.

Law's features

The best recognition performance is achieved using model-1 ($k=7$). The recognition rate of 73.5% is better than the linear classifier performance of 67.6%. Classification mistakes are made when tree samples are confused as grass or trees, grass samples are confused as leaves, and leaves are confused as trees or grass. The individual class recognition accuracy are: trees (75.3%), grass (84.4%), and leaves (53.6%). Compared to the linear classifier, we are better at recognising trees by 10%, grass by 24% but poorer at recognising leaves by 28%.

Primitive length features

The best results are obtained for the single nearest neighbour model with 55.6% recognition success. This performance is much better compared to the linear classifier recognition of 41.1%. The individual class recognition accuracy are: trees (59.1%), grass (60.6%), and leaves (42.3%).

This shows that compared to the linear classifier we are better at recognising trees by 4%, grass by 43% but poorer at recognising leaves by 17%.

Summary

On the whole we find that the k NN classification is inferior to the linear method except for couple of feature sets. There is however a considerable improvement with the Law's and primitive length method. On the whole, the k NN classifier makes different kinds of mistakes than the linear classifier. In Table 7.21 we present the summary of the nearest neighbour classifier performance using split and merge segmentation and the five feature extraction methods. On the whole the best performer is the edge frequency method closely followed by Law's method. Grass appears to be the easiest to recognise in most of these feature sets.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	64.2%	Grass (87.6%)
Co-occurrence	66.2%	Grass (77.1%)
Edge frequency	74.0%	Trees (82.2%),
Law's	73.5%	Grass (84.4%)
Primitive length	55.6%	Grass (84.4%)

Table 7.21 Summary of nearest neighbour classifier performance with Split and Merge.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. In Table 7.22 we show a better presentation of these mistakes.

<i>Feature method</i>	<i>Class</i>	<i>Trees</i>	<i>Grass</i>	<i>Leaves</i>
<i>Autocorrelation</i>	<i>Trees</i>		*	**
	<i>Grass</i>			
	<i>Leaves</i>	**	*	
<i>Co-occurrence</i>	<i>Trees</i>		*	*
	<i>Grass</i>	*		*
	<i>Leaves</i>	**	**	
<i>Edge frequency</i>	<i>Trees</i>		*	
	<i>Grass</i>	**		
	<i>Leaves</i>		*	
<i>Law's</i>	<i>Trees</i>			*
	<i>Grass</i>			*
	<i>Leaves</i>	*	**	
<i>Primitive length</i>	<i>Trees</i>		*	*
	<i>Grass</i>	*		*
	<i>Leaves</i>	*	**	

Table 7.22 Description of mistakes made by the nearest neighbour classifier with Split and Merge.

Table 7.22 shows that trees get confused as both grass and leaves with all feature sets. Grass is more likely to be confused as leaves except for the edge frequency method where a considerable amount of grass is labelled as trees. Leaves have an equal likelihood of being labelled by mistake as grass or trees. For autocorrelation method, errors are biased in favour of grass, for

co-occurrence they are equally balanced, for edge frequency they are biased in favour of grass, for Law's they are biased in favour of leaves and for primitive length method they are balanced.

7.3.5 Summarising nearest neighbour classification results on vegetation data

In the above discussion we have presented the results of the nearest neighbour classifier. The summaries have been presented by grouping different feature extraction methods under a segmentation scheme. We can actually summarise the classifier mistakes in a reverse manner, i.e. how does the classifier make mistakes keeping the same feature extraction method based on the output of different segmentation methods. In this section we aim to show some tables with this information. Also, we present a final table showing the classification results of different segmentation and feature extraction combinations at one place.

We summarise how the nearest neighbour classifier makes the mistakes for each feature extraction method by changing the preceding segmentation process. For a total of 5 feature extraction methods, Tables 7.23-7.27 are drawn. Finally, Table 7.28 shows which combination of segmentation and texture extraction method yields the best classification on vegetation data.

The results in Tables 7.23 to 7.27 appear more uniform across different segmentation method for a given texture method than they did for different feature extraction methods for a given segmentation method. We can draw the following conclusions.

- i) For autocorrelation feature extraction, similar mistakes are made for histogram based segmentation and split and merge. Using FCM, misclassification as leaves is small. Most segmentation models yield errors in favour of grass, i.e. the two other classes get misclassified as grass. FCM is similar to histogram based segmentation except for the fact that no mistakes in favour of leaves are made. Also region growing is similar to histogram based segmentation except for a number of grass samples mistaken as leaves and trees.
- ii) For co-occurrence features, all segmentation methods make almost identical mistakes except for the number of mistakes and that region growing does not yield mistakes of trees confused as grass.
- iii) For edge frequency measures, histogram thresholding and region growing segmentation yield similar errors. In both cases, other classes get misclassified as trees or leaves, but nothing is mistaken as grass. FCM is slightly superior in that hardly any samples get mistaken as grass or leaves. Split and merge is different from all others: here grass gets confused as trees, and trees and leaves get confused as grass.
- iv) For Law's features, we find two separate groups. In group 1, we can put FCM and histogram thresholding with the minor difference that the histogram based segmentation makes mistakes in confusing tree and grass samples as leaves. In the second group we can put split and merge and region growing methods that make identical mistakes.
- v) For primitive length features, all segmentation methods yield similar errors. Most noticeable difference is the large number of mistakes made by the region growing method in confusing grass samples as trees.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees	*	*	
	Grass	**	*	
	Leaves			
Histogram	Trees		*	*
	Grass	**	*	
	Leaves			
Region Growing	Trees	*	*	*
	Grass	*		*
	Leaves	***	*	
Split & Merge	Trees		*	**
	Grass			
	Leaves	**	*	

Table 7.23 *k*NN classifier mistakes for autocorrelation features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees		*	*
	Grass	**	*	*
	Leaves	**	*	
Histogram	Trees		*	*
	Grass	**	*	*
	Leaves	**	*	
Region Growing	Trees			**
	Grass	**		**
	Leaves	**	*	
Split & Merge	Trees		*	*
	Grass	*		*
	Leaves	**	**	

Table 7.24 *k*NN classifier mistakes for co-occurrence features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees			
	Grass	**		
	Leaves	**		
Histogram	Trees			*
	Grass	*		*
	Leaves	*		
Region Growing	Trees			*
	Grass	***		*
	Leaves	**		
Split & Merge	Trees		*	
	Grass	**		
	Leaves		*	

Table 7.25 *k*NN classifier mistakes for edge frequency features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees		*	
	Grass	*	*	
	Leaves	**	*	
Histogram	Trees		*	*
	Grass	*	*	*
	Leaves	**	*	
Region Growing	Trees			*
	Grass			*
	Leaves	**	*	
Split & Merge	Trees			*
	Grass			*
	Leaves	*	**	

Table 7.26 *k*NN classifier mistakes for Law's features.

Segmentation method	Class	Trees	Grass	Leaves
FCM	Trees		**	*
	Grass	**	*	*
	Leaves	**	*	
Histogram	Trees		*	*
	Grass	**		*
	Leaves	**	**	
Region Growing	Trees			**
	Grass	***		**
	Leaves	**	*	
Split & Merge	Trees		*	*
	Grass	*		*
	Leaves	*	**	

Table 7.27 *k*NN classifier mistakes for primitive length features.

The overall performance of the classification scheme is shown in Table 7.28 and Figure 7.22.

Feature Segmentation	Autocorrelation	Co-occurrence	Edge frequency	Law's	Primitive length	μ	σ
FCM	72.9%	56.5%	72.0%	74.6%	57.6%	66.7%	8.8%
Histogram	70.0%	57.3%	70.0%	59.1%	46.1%	60.5%	10.0%
Region Grow	58.2%	52.2%	65.0%	64.0%	56.7%	59.2%	5.3%
Split & Merge	64.2%	66.2%	74.0%	73.5%	55.6%	66.7%	7.6%
μ	66.3%	58.1%	70.2%	67.8%	54.0%	-	-
σ	6.5%	5.8%	3.8%	7.5%	5.3%	-	-

Table 7.28 The different nearest neighbour classifier performance depending on which data set is used for vegetation analysis.

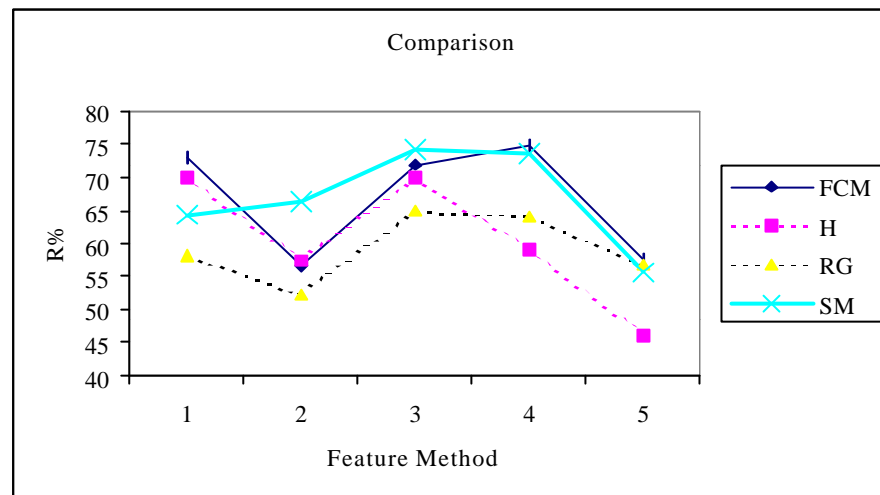


Figure 7.22 A graphical comparison of recognition accuracy obtained using different segmentation method and texture method combinations for vegetation data analysis.

On the basis of Table 7.28 and Figure 7.22, the following salient observations can be made:

- FCM and split and merge performances are very similar for all feature extraction methods. FCM is the best on 3 feature extraction methods and split and merge is the best on the remaining two methods.
- Region growing is the worst performer on three feature sets and histogram thresholding on two others.
- There is considerable variability in both rows and columns adding weight the argument that the choice of appropriate combination of segmentation and feature extraction methods is crucial. For example the best performance stands at 74.5% correct (FCM segmentation followed by Law's features), and the worst performance stands at 46.1% (Histogram thresholding followed by primitive length features) - a significant difference of nearly 28%. For segmentation methods, in order of decreasing variability, the methods are ranked as: histogram based segmentation, FCM, split and merge, and region growing. Feature extraction methods can be ranked in order of decreasing variability as follows: Law's, autocorrelation, co-occurrence, primitive length and edge frequency. If we are looking for the least variable feature extraction method, then edge frequency is the best.

Chapter 8

Natural object data analysis

In the previous chapter we discussed the analysis of vegetation data and obtained very good results on this non trivial problem. The other class of data that we did not discuss was natural objects. These natural objects include bricks, sky, clouds, pebbles and road. It was suggested in our experimental design that such object regions can be separated in the original data from vegetation on the basis of colour quite easily. In this chapter we show the results obtained using leave-one-out classification using both the linear classifier and the nearest neighbour classifier. We lay out our results in a similar fashion to the previous chapter. The results are first discussed for the linear classifier and then for the nearest neighbour method. The results are first presented for each segmentation method and then later summarised for each feature extraction method. For each segmentation method, we get a different number of regions and thus samples.

8.1 Linear classification

The linear classification scheme is important to determine how well the natural data distributions can be separated linearly. It is not always the case that they would perform inferior to approaches such as nearest neighbour as we found in the last chapter. In this chapter we describe the experiments with linear classifier first for FCM segmentation followed by, in order, histogram thresholding, region growing and split and merge. For each of the linear analysis we plot the PCA plot and the canonical discriminant function plot showing how well the two principal components differentiate the different classes and also how well decision boundaries between the two classes can be placed.

8.1.1 Fuzzy c-means clustering segmentation

We present the following results of feature sets that were extracted on images that were segmented using FCM clustering. The results for these are available in Appendix M.

Autocorrelation features

In Figure 8.1 the PCA plot and the discriminant plot for all groups is shown.

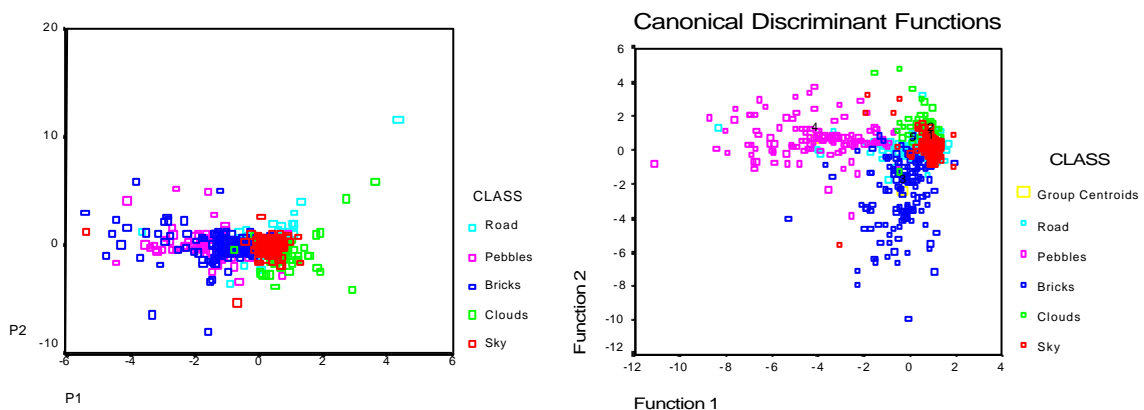


Figure 8.1 PCA plot and canonical discriminant function plot for the linear analysis using FCM and *autocorrelation* features for the discrimination of natural object data.

The following conclusions can be drawn immediately from the above plots. First, sky, road and clouds form compact clusters. Pebbles and bricks appear more variable. Second, there appears to be a certain degree of overlap between the first three classes mentioned. The test recognition rate of 69.1% is achieved. The majority of the mistakes are made when sky is confused as clouds, clouds are confused as sky, bricks are confused as road, pebbles are confused as road and when road is confused as sky. The individual recognition rates for the five classes are: sky (81.6%), clouds (64.0%), bricks (60.6%), pebbles (81.0%) and road (51.3%). Hence, it is easiest to recognise sky and pebbles and most difficult to recognise roads.

Co-occurrence features

The PCA and discriminant function plots for this data are shown in Figure 8.2.

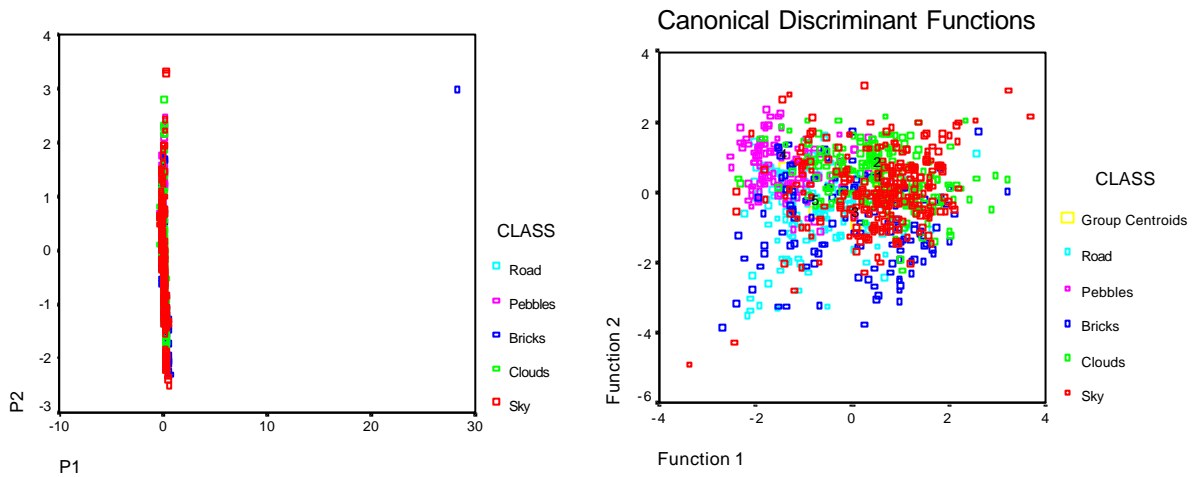


Figure 8.2 PCA plot and canonical discriminant function plot for the linear analysis using FCM and *co-occurrence* features for the discrimination of natural object data.

There is much more overlap across different classes in these plots. The principal components on their own give little information for classification purposes. In the discriminant plot, most categories appear overlapped except for pebbles that are isolated than the rest. The overall leave-one-out recognition of 51.1% is obtained for the analysis. The individual classes are recognised with the following accuracy: sky (39.1%), clouds (55.3%), bricks (33.9%), pebbles (78.4%), and road (60.9%). Hence, for most classes the accuracy is quite poor. Only pebbles and road have a reasonable result. We find that errors are not biased in favour of one class but uniformly distributed across different classes in almost equal numbers.

Edge frequency features

The PCA plot and the canonical discriminant function plots are shown in Figure 8.3. The plots show that the different classes appear as separate clusters for most part. In particular, pebbles cluster is disjoint from the rest. The recognition rate of 71.0% correct confirms that the data is fairly linearly separable. The individual recognition accuracy for classes are: sky (84%), clouds (60.9%), bricks (53.3%), pebbles (93.4%) and road (60.5%). Detailed inspection of the confusion matrix shows that the majority of mistakes can be attributed to sky samples confused as clouds and vice-versa, bricks samples confused as sky and road, and road confused as sky. It appears reasonable that roads could appear similar to sky as they have similar texture. Also sky

and cloud confusion is not out of the ordinary as in cases when clouds are not dense, they may appear similar to sky. The only unexpected result is bricks confused as sky and road.

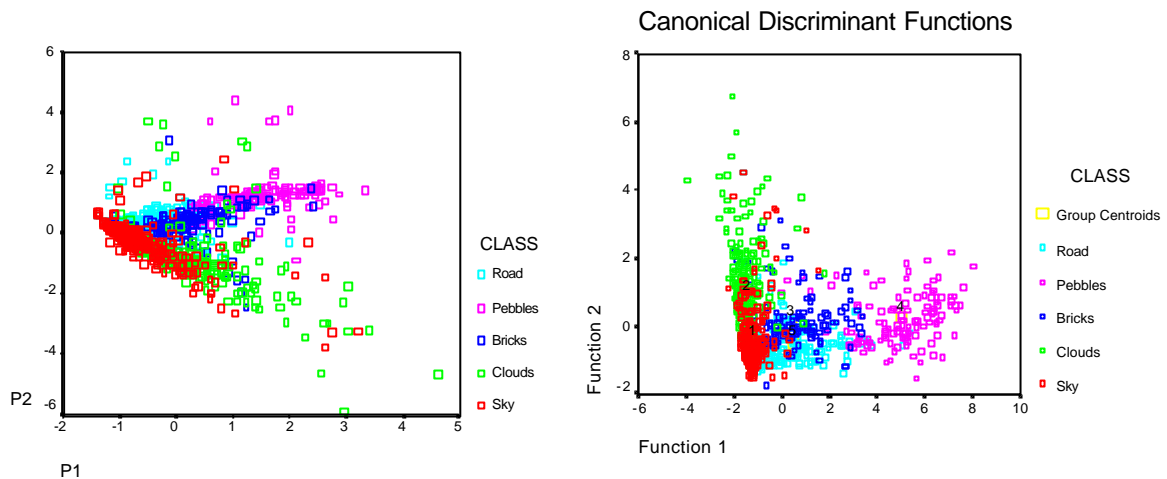


Figure 8.3 PCA plot and canonical discriminant function plot for the linear analysis using FCM and *edge frequency* features for the discrimination of natural object data.

Law's features

The PCA and canonical discriminant function plots are shown in Figure 8.4.

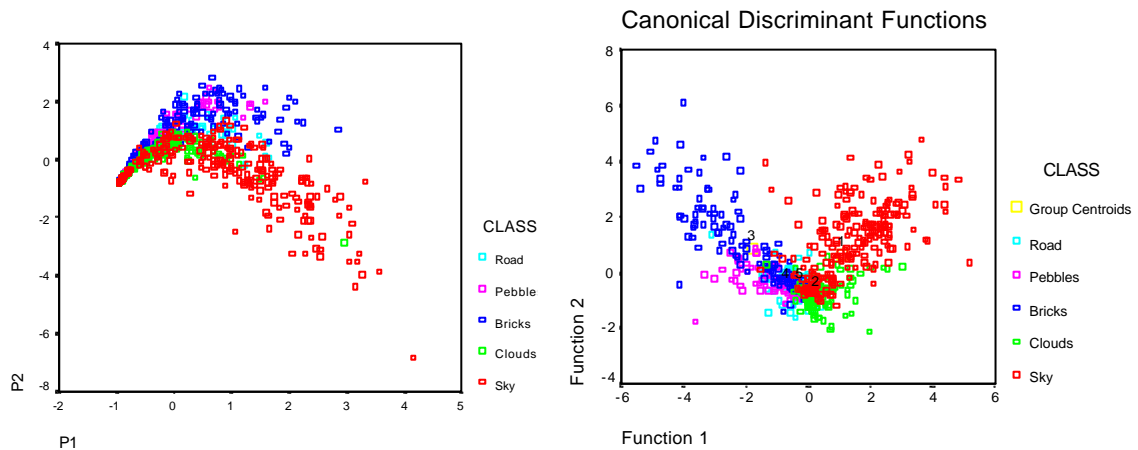


Figure 8.4 PCA plot and canonical discriminant function plot for the linear analysis using FCM and *Law's features* for the discrimination of natural object data.

Figure 8.4 shows a boomerang shaped data distribution. At the two ends of this structure lie sky and bricks data distributions. The other classes overlap in the middle. We obtain a linear classification accuracy of 58.1% correct. The individual class recognition is as follows: sky (51.2%), clouds (89.5%), bricks (50.4%), pebbles (48.8%) and road (34.9%). Clouds have been recognised with the highest accuracy and road recognition is the worst. The mistakes are made when sky is confused as clouds, bricks are confused as clouds, pebbles or road, pebbles are confused as clouds, and road is confused as clouds or pebbles.

Primitive length features

Figure 8.5 shows the PCA plot. The analysis fails to yield a discriminant function plot. Unfortunately, not much information can be gained from the PCA plot alone.

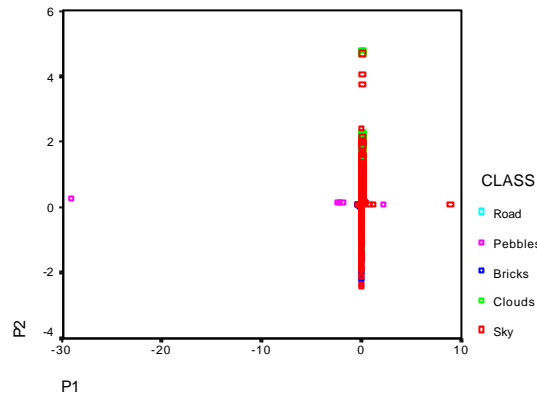


Figure 8.5 PCA plot using FCM and *primitive length* features for discrimination of natural object data.

The confusion matrix shows a recognition accuracy of 38.7% correct with the individual class recognition accuracy as follows: sky (17.4%), clouds (73.4%), bricks (57.7%), pebbles (25.6%) and road (16.4%). Clouds have been recognised with the highest accuracy. This result is supported by other work within our laboratory with cloud analysis where it was found that in distinguishing different varieties of clouds, the primitive length method was the best. A large number of mistakes are however made. The most significant ones include large numbers of sky samples confused as clouds and bricks, large number of pebble samples confused as sky, and large number of road samples confused as bricks.

Summary

In Table 8.1 we summarise the overall results and highlight which class was best recognised.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	69.1%	Sky (81.6%)
Co-occurrence	51.1%	Pebbles (78.4%)
Edge frequency	71.0%	Pebbles (93.4%)
Law's	58.1%	Clouds (89.5%)
Primitive length	38.7%	Clouds (73.4%)

Table 8.1 Summary of linear classifier performance with FCM.

The best results are obtained for the edge frequency measures. Pebbles and clouds appear as one of the easiest classes to distinguish from others. Co-occurrence method, widely considered as an important texture analysis method, is slightly disappointing.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. This information present in confusion matrices is not easily interpreted. In Table 8.2 we show a better presentation of these mistakes. For each method of feature extraction, if the class in row M gets misclassified as class in row N by more than 10% then we put one asterisk in that matrix position. We put two asterisks for more than 25% misclassification, and three asterisks for more than 50% misclassification.

Table 8.2 shows that co-occurrence and primitive length method make most of the mistakes. Primitive length method is the worst performer so it can be considered separately. On all other

methods we find that mistakes are likely to be made when clouds get mistaken as sky and vice-versa and when bricks are confused as road. Each method also has its distinguishing characteristics. For example, for autocorrelation and edge frequency methods, hardly any mistakes are made confusing other class data as bricks or pebbles. For both of these methods the mistakes are biased in favour of sky class. Similarly for co-occurrence nothing except sky samples get confused as bricks. Mistakes are biased in favour of pebbles. For Law's method nothing gets confused as sky or bricks and mistakes are biased in favour of clouds. Primitive length makes a number of mistakes where other classes are less likely to be mistaken as clouds. Also as we noted before, it distinguishes clouds very well.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					*
	<i>Road</i>	**				
<i>Co-occurrence</i>	<i>Sky</i>		**	*	*	*
	<i>Clouds</i>	*			*	
	<i>Bricks</i>	*	*		*	**
	<i>Pebbles</i>					
	<i>Road</i>				*	
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					
	<i>Road</i>	*				
<i>Law's</i>	<i>Sky</i>		**			
	<i>Clouds</i>					
	<i>Bricks</i>		*		*	*
	<i>Pebbles</i>		**			
	<i>Road</i>		**		*	
<i>Primitive length</i>	<i>Sky</i>		**	**	*	*
	<i>Clouds</i>	*				
	<i>Bricks</i>	*			*	*
	<i>Pebbles</i>	***		*		
	<i>Road</i>	*		***	*	

Table 8.2 Description of mistakes made by the linear classifier with FCM.

8.1.2 Histogram thresholding segmentation

We present the following results of feature sets that were extracted on images that were segmented using histogram thresholding. The results for these are available in Appendix O.

Autocorrelation features

In Figure 8.6 the PCA plot and the discriminant plot for all groups is shown.

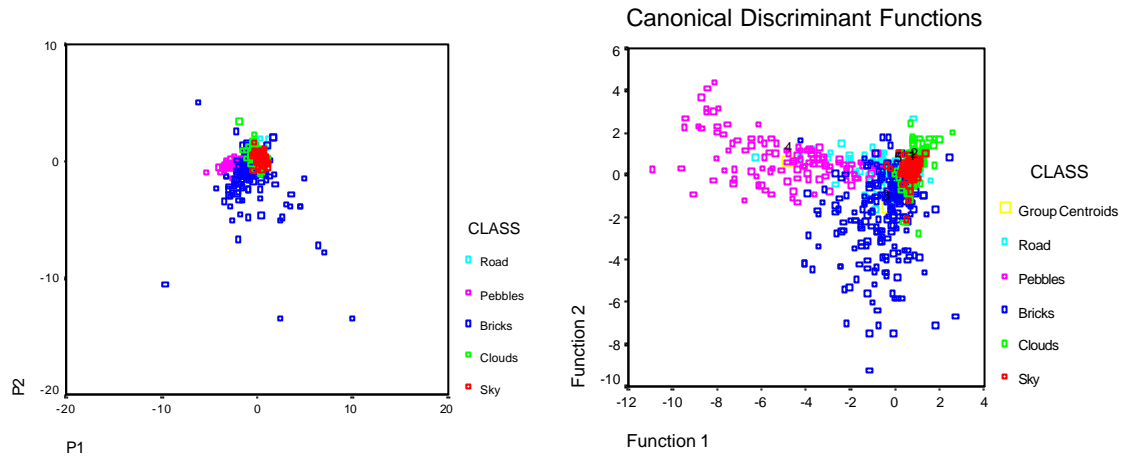


Figure 8.6 PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding and *autocorrelation* features for the discrimination of natural object data.

The following observations can be made from the above figure. First, sky and clouds appear as a compact cluster. Pebbles cluster has high variability but it is disjoint. Second, there are a number of outliers in the data. Most of them are present in pebbles and bricks data. Also, road, cloud and sky appear too close and similar to each other. If we were to have a third layer of classification which could separate these three from pebbles and bricks, even better recognition could be obtained we hope. We achieve a recognition performance of 63.0% correct with the following class accuracy: sky (82.8%), clouds (52.8%), bricks (51.5%), pebbles (80.7%) and road (42.3%). These results are consistent with the plots shown above. The misclassifications can be attributed to sky being confused as clouds and vice-versa, bricks confused as sky and road, pebbles confused as road, and road confused as sky.

Co-occurrence features

The PCA and canonical discriminant function plots are shown in Figure 8.7. The PCA plot does not show much discriminatory information. In the other plot, sky appears as one tight cluster at the bottom and as another spread out cluster overlapping other classes. All classes appear mixed with each other and as such this method can not be expected to yield the best results for our analysis.

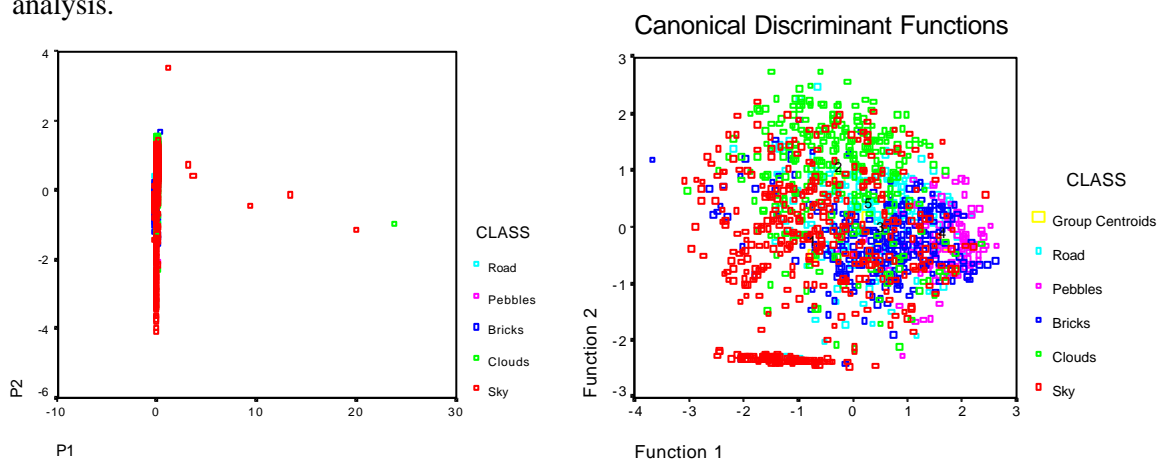


Figure 8.7 PCA plot and canonical discriminant function plot for the linear analysis using Histogram Thresholding segmentation and *co-occurrence* features for the discrimination of natural object data.

We obtain a result of 53.6% correct recognition for linear analysis. The individual classes can be recognised with the following accuracy: sky (49.3%), clouds (61.7%), bricks (52.9%), pebbles (81.4%) and road (34.2%). Here pebbles are recognised the best and road the worst. In terms of mistakes they are not biased in favour of any single class. Almost all classes are confused as something else with a roughly equal number. It appears that the results can be improved with such data by outlier elimination and using a non-linear classification method.

Edge frequency features

The PCA and canonical discriminant plots are shown in Figure 8.8.

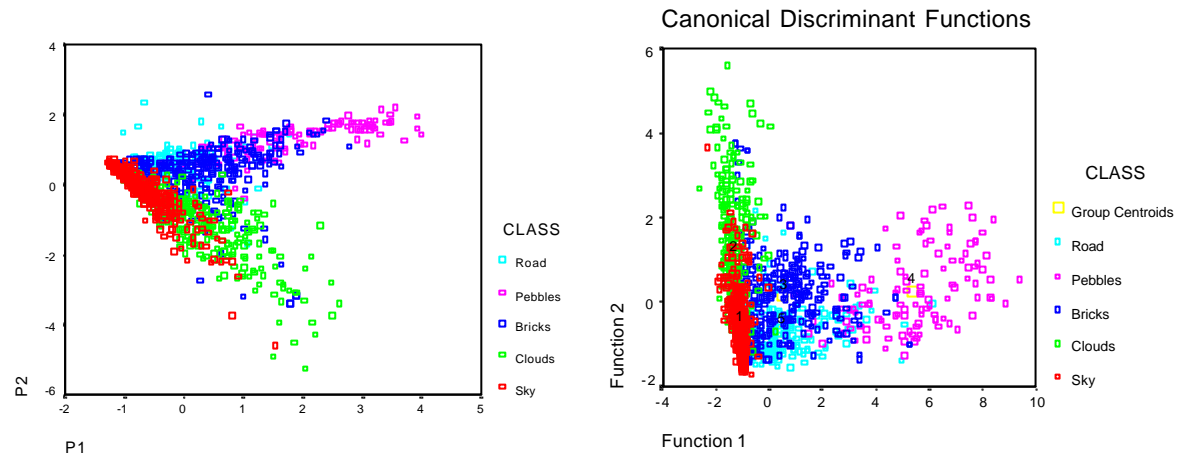


Figure 8.8 PCA plot and canonical discriminant function plots for the linear analysis using Histogram Thresholding segmentation and edge frequency features for the discrimination of natural object data.

In these plots we find the best separation across different groups. Sky and cloud samples appear as compact groups but there is a region of overlap between these two classes. Pebbles appear as a scattered but disjoint group. Road samples are the worst affected by the overlap. The recognition rate of 69.7% is reasonably good for this data. The individual class accuracy stands at: sky (87.4%), clouds (63.4%), bricks (48.5%), pebbles (90.4%) and road (59.2%). The mistakes are made when some sky samples are confused as clouds and vice versa, bricks are confused as sky, clouds and road, and road is confused as sky.

Law's features

The PCA and discriminant function plots are shown in Figure 8.9.

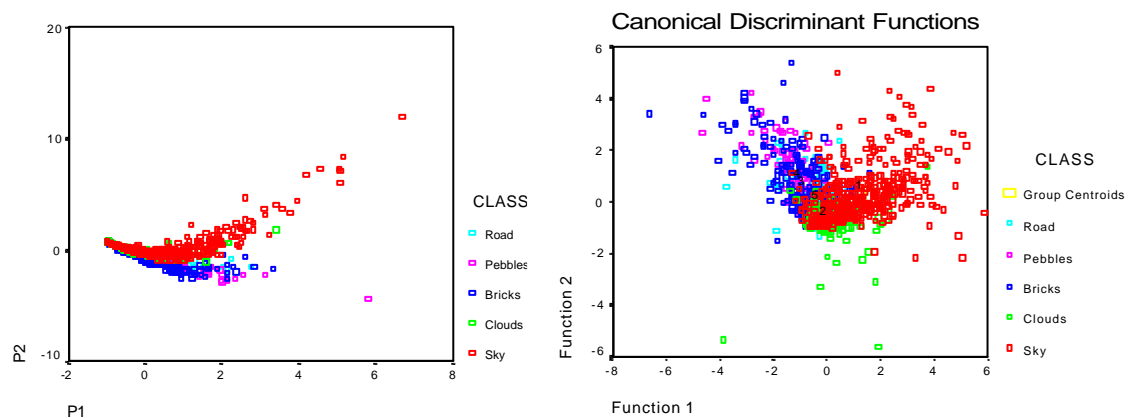


Figure 8.9 PCA plot and canonical discriminant function plots for the linear analysis using Histogram Thresholding and Law's features for the discrimination of natural object data.

Sky and bricks appear as disjoint groups but tend to overlap the distributions of other classes. The appendix shows result of 53.4% correct. The individual class accuracy is found as sky (59.2%), clouds (88.8%), bricks (30.3%), pebbles (36.0%) and road (28.6%). Most of the mistakes are made when sky samples are confused as clouds, bricks are confused as clouds or road, pebbles are confused as clouds or road and road is confused as clouds. The mistakes appear to be biased in favour of clouds.

Primitive length features

In Figure 8.10 we show the PCA plot of this method. The analysis fails to give the discriminant function plot.

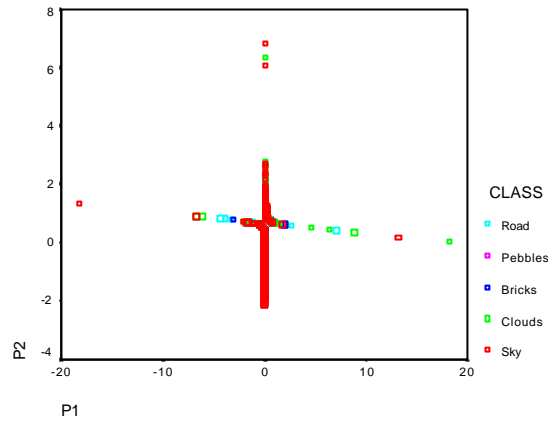


Figure 8.10 PCA plot using Histogram Thresholding and *primitive length* features for the discrimination of natural object data.

We obtain a recognition accuracy of 31.9% correct. The individual classes are recognised with the following accuracy: sky (6.9%), clouds (64.7%), bricks (10.9%), pebbles (49.1%), and road (53.6%). Majority of the mistakes are made when sky is confused as clouds or road. Also cloud samples get confused as pebbles or road, bricks get confused as pebbles or road, pebbles get confused as clouds and road gets confused as sky and clouds. The performance on recognising clouds is quite good but the method exhibits a very poor performance of recognising sky and bricks.

Summary

In Table 8.3 we summarise the overall results and highlight which class was best recognised.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	63.0%	Sky (82.8%)
Co-occurrence	53.6%	Pebbles (81.4%)
Edge frequency	69.7%	Pebbles (90.4%)
Law's	53.4%	Clouds (88.8%)
Primitive length	31.9%	Clouds (64.7%)

Table 8.3 Summary of linear classifier performance for Histogram Thresholding.

The best results are obtained for the edge frequency measures. Pebbles and clouds appear as one of the easiest classes to distinguish from the others.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. Table 8.4 shows these mistakes in a more meaningful manner.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					*
	<i>Road</i>	**	*			
<i>Co-occurrence</i>	<i>Sky</i>		*	*		
	<i>Clouds</i>	*		*		*
	<i>Bricks</i>				*	*
	<i>Pebbles</i>			*		
	<i>Road</i>	*	*	*	*	
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*			*
	<i>Pebbles</i>					
	<i>Road</i>	**				
<i>Law's</i>	<i>Sky</i>		**			
	<i>Clouds</i>					
	<i>Bricks</i>		**		*	*
	<i>Pebbles</i>		**	*		*
	<i>Road</i>		**		*	
<i>Primitive length</i>	<i>Sky</i>		**			***
	<i>Clouds</i>				*	*
	<i>Bricks</i>	*	*		*	**
	<i>Pebbles</i>		**			*
	<i>Road</i>	*	*			

Table 8.4 Description of mistakes made by the linear classifier with Histogram Thresholding.

Different feature extraction methods are biased in making mistakes in favour of particular classes. For example, for autocorrelation and edge frequency methods, most of the mistakes are made as samples of other classes are mistaken as sky. Co-occurrence model is biased in favour of bricks, and Law's and primitive length in methods favour of clouds. Both autocorrelation and edge frequency methods do not make many mistakes confusing samples of other classes for bricks or pebbles. Similarly Law's method does not by mistake confuse anything as sky and primitive length does not by mistake confuse anything as bricks. On the whole, for almost all methods, sky and cloud samples are confused as each other. Bricks are confused as road, and road as sky or cloud, or both.

8.1.3 Region growing segmentation

We present the following results of feature sets that were extracted on images that were segmented using region growing. The results for these are available in Appendix Q.

Autocorrelation features

In Figure 8.11 the PCA plot and the discriminant plot for all groups is shown. The sky and cloud clusters appear fairly compact. These overlap the road cluster. The bricks and pebbles samples are scattered but easily distinguishable from the other categories. The road cluster group appears to have several outliers that are surrounded by bricks and pebbles samples. The overall recognition rate of 64.3% is obtained. The individual classes are recognised with the following accuracy: sky (78.1%), clouds (58.5%), bricks (62.2%), pebbles (75.4%) and road (30.5%). The poor performance on recognising roads is as expected. Sky and pebbles are best recognised. The majority of the mistakes are made when sky samples are confused as cloud and vice-versa. Also brick and pebbles samples are confused as road. Road is confused as sky and cloud.

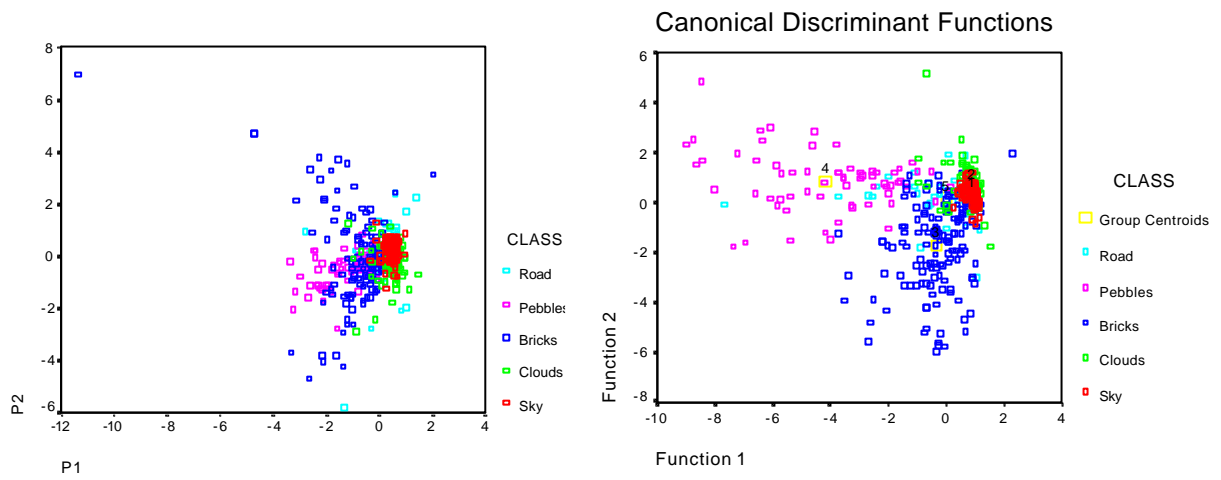


Figure 8.11 PCA plot and canonical discriminant function plots for the linear analysis using Region Growing and *autocorrelation* features for the discrimination of natural object data.

Co-occurrence features

The PCA plot and the discriminant function plot for this feature set is shown in Figure 8.12.

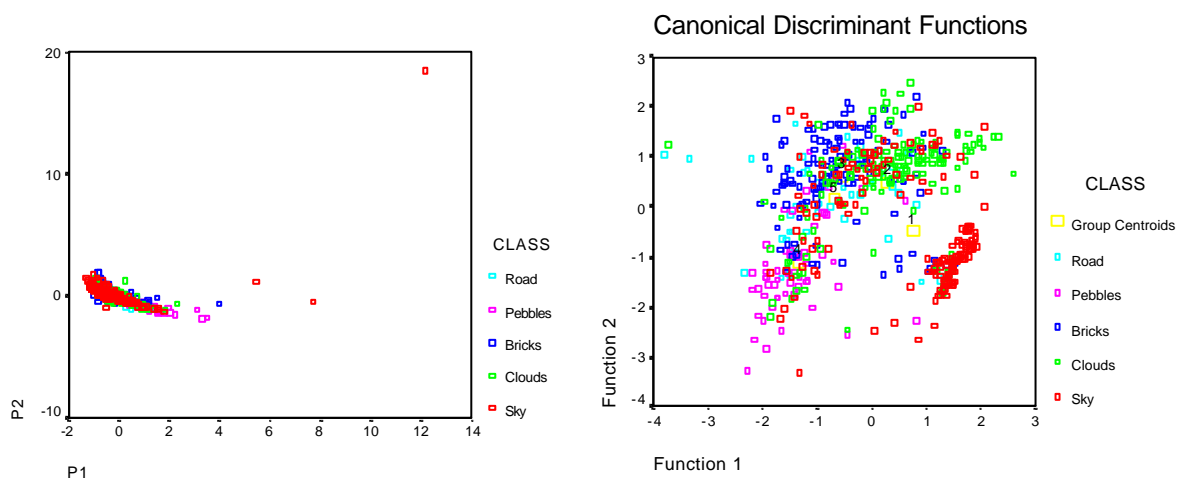


Figure 8.12 PCA plot and canonical discriminant function plots for the linear analysis using Region Growing segmentation and *co-occurrence* features for the discrimination of natural object data.

The PCA plot shows almost a complete overlap and little discriminatory information in principal components alone. The discriminant function plot shows sky distributed as two

separate clusters: one cluster is very compact and the other is fairly scattered. There is a considerable overlap across all classes. Pebbles appear to overlap with road, bricks with clouds and road, and sky with clouds. The confusion matrix shows an overall recognition accuracy of 55.2% correct. The individual classes are recognised with the following accuracy: sky (57.8%), clouds (58.0%), bricks (46.6%), pebbles (76.3%), and road (33.3%). This poor performance is explained by the following misclassifications: sky samples confused as cloud and vice-versa, sky and clouds confused as bricks, bricks confused as sky, cloud and roads, pebbles confused as road, and road confused as bricks and pebbles.

Edge frequency features

The PCA and canonical discriminant function plot is shown in Figure 8.13. This plot appears to be one of the best that we have seen so far. Each class appears as a separate cluster except for road samples that are overlapped by brick and sky samples. The linear classifier can be expected to give a decent results on this data. The confusion matrix shows an overall result of 68.6% correct recognition. The individual classes are recognised with the following accuracy: sky (89.3%), clouds (65.9%), bricks (46.9%), pebbles (92.3%) and road (37.3%). The poor result for roads and bricks is to be expected considering the large overlap across their samples. The majority of the mistakes are made when sky samples are confused as clouds and vice-versa, bricks are confused as sky or road, and when road is confused as other classes, in particular sky.

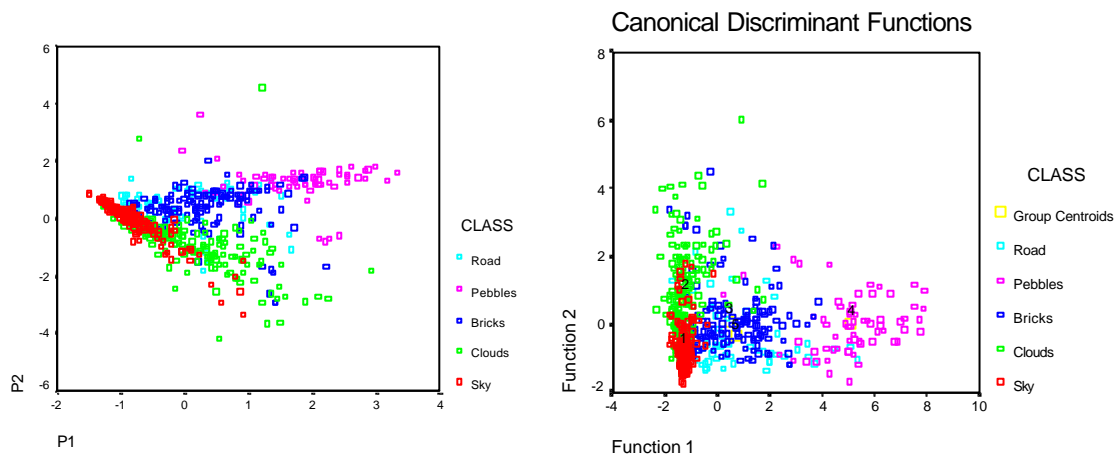


Figure 8.13 PCA plot and canonical discriminant function plots for the linear analysis using Region Growing and edge frequency features for the discrimination of natural object data.

Law's features

The PCA plot and the discriminant function plot for this data set are shown in Figure 8.14. These plots show the classic boomerang shape. At one end of it are the brick samples and at the other end sky samples. This looks very similar to the plot obtained for Law's features using FCM segmentation. There is a considerable overlap between pebbles, brick and road samples. At the other end, these three categories further overlap with clouds. Sky samples appear distinctly separate from a small cluster overlapped by clouds. The linear classifier yields a recognition performance of 67.5% correct. The individual classes are recognised with the following accuracy: sky (71.1%), clouds (93.2%), bricks (50.3%), pebbles (56.9%) and road (32.2%). The majority of the mistakes are made when sky samples are confused as clouds, bricks are confused as clouds, pebbles and road, pebbles are confused as clouds or road, and road is confused as clouds or pebbles.

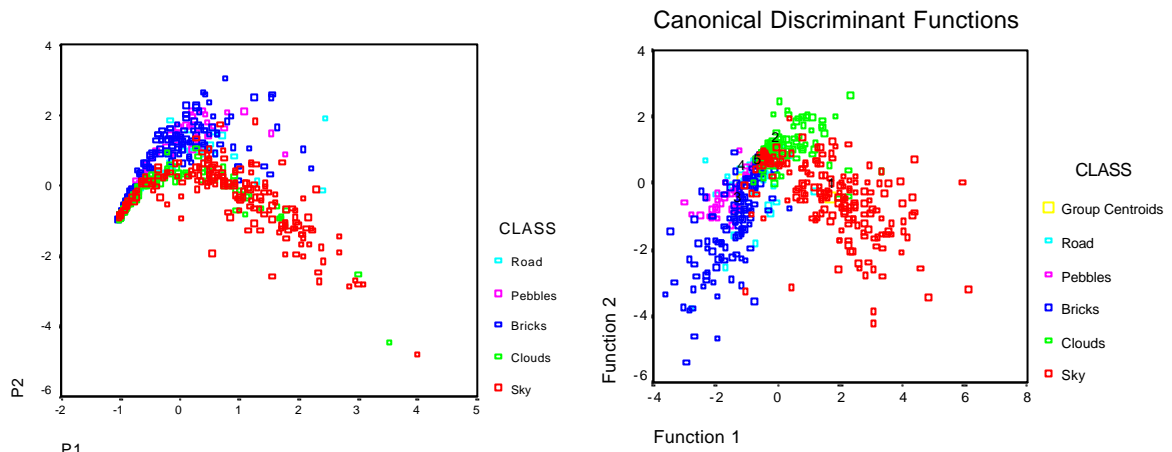


Figure 8.14 PCA plot and canonical discriminant function plots for the linear analysis using Region Growing segmentation and *Law's* features for the discrimination of natural object data.

Primitive length features

The PCA plot for this analysis is shown in Figure 8.15. The analysis fails to yield a discriminant function plot. The plot contains little information for classifying the five categories of data.

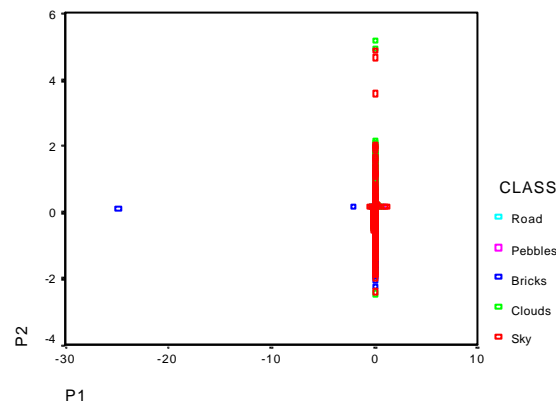


Figure 8.15 PCA plot for the linear analysis using Region Growing segmentation and *primitive length* features for the discrimination of natural object data.

The linear classifier yields a best recognition performance of 43.8% correct. The individual class recognition performances can be shown as: sky (8.0%), clouds (75.0%), bricks (54.5%), pebbles (49.2%) and road (32.2%). Sky has been confused in equal numbers as almost everything else. Clouds are primarily confused as pebbles, bricks are confused as pebbles and road, pebbles are confused as bricks and road is confused as a bit of everything else.

Summary

In Table 8.5 we summarise the overall results and highlight which class was best recognised.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	64.3%	Sky (78.1%)
Co-occurrence	55.2%	Pebbles (76.3%)
Edge frequency	68.6%	Pebbles (92.3%)
Law's	67.5%	Clouds (93.2%)
Primitive length	43.8%	Clouds (75.0%)

Table 8.5 Summary of linear classifier performance with Region Growing.

The best results are obtained for the edge frequency measures. These are closely followed by the Law's feature and autocorrelation performances. Pebbles and clouds appear as one of the easiest classes to distinguish from the others.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>					**
	<i>Pebbles</i>					*
	<i>Road</i>	**	*			
<i>Co-occurrence</i>	<i>Sky</i>		*	*		
	<i>Clouds</i>	*		*	*	
	<i>Bricks</i>	*	*		*	*
	<i>Pebbles</i>					*
	<i>Road</i>		*	*	*	
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					
	<i>Road</i>	*	*	*	*	
<i>Law's</i>	<i>Sky</i>		*			
	<i>Clouds</i>					
	<i>Bricks</i>		*		*	*
	<i>Pebbles</i>		**			
	<i>Road</i>		**		*	
<i>Primitive length</i>	<i>Sky</i>		*	**	*	*
	<i>Clouds</i>				*	
	<i>Bricks</i>				*	*
	<i>Pebbles</i>	*	*	*		
	<i>Road</i>	*		**		

Table 8.6 Description of mistakes made by the linear classifier.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. Table 8.6 shows these mistakes in a more meaningful manner. The performance of each feature set is different than others. For autocorrelation feature set, mistakes are biased in favour of sky and road. For co-occurrence feature set they are more balanced across different classes. In the case of edge frequency, the misclassification are biased toward sky, for Law's feature set they are biased toward clouds, and they again appear fairly balanced across different classes for the primitive length feature set. Also some feature sets never make mistakes in favour of certain classes. For example, for autocorrelation features, hardly any samples are by mistake assigned to bricks and pebbles classes. Similarly in the case of Law's features, hardly by mistake anything is assigned to sky.

8.1.4 Split and merge segmentation

We present the following results of feature sets that were extracted on images that were segmented using region split and merge. The results for these are available in Appendix S. These results have been obtained using leave-one-out cross validation.

Autocorrelation features

The PCA and canonical discriminant plots are shown in Figure 8.16.

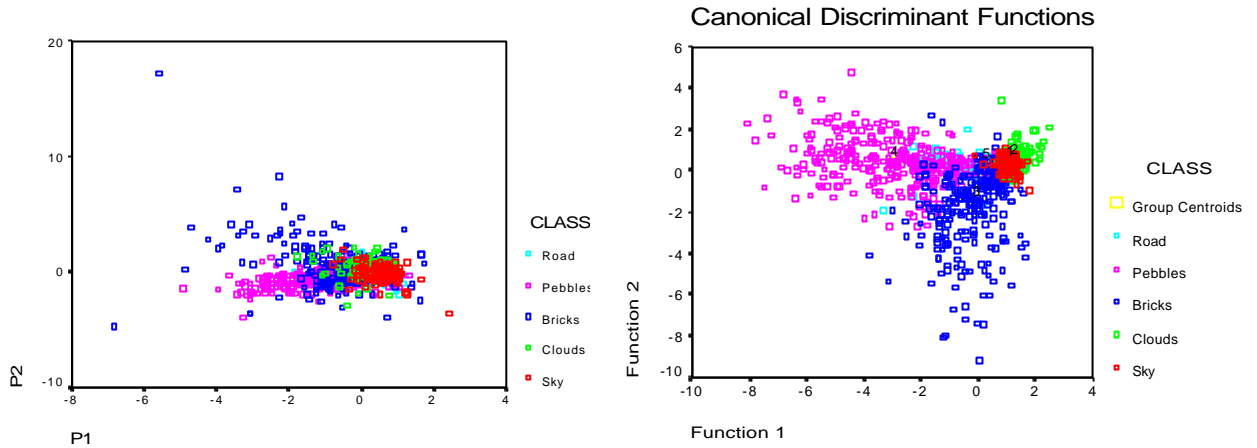


Figure 8.16 PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge and *autocorrelation* features for the discrimination of natural object data.

The above plots show fairly good clusters for all groups except for road. Sky and cloud samples form a very tight cluster. Also bricks and pebbles form distinct clusters. There is a significant overlap though between sky and clouds, sky and bricks and road samples with others. The overall recognition accuracy of 64.7% is achieved with the linear classifier. The individual classes are recognised with the following accuracy: sky (81.3%), clouds (58.7%), bricks (52.6%), pebbles (72.6%), and road (48.4%). The majority of the mistakes are made when sky samples are confused as clouds and vice-versa, brick samples are confused as sky, clouds and road, pebbles are confused as road, and road is confused as sky.

Co-occurrence features

The PCA and canonical discriminant plots are shown in Figure 8.17.

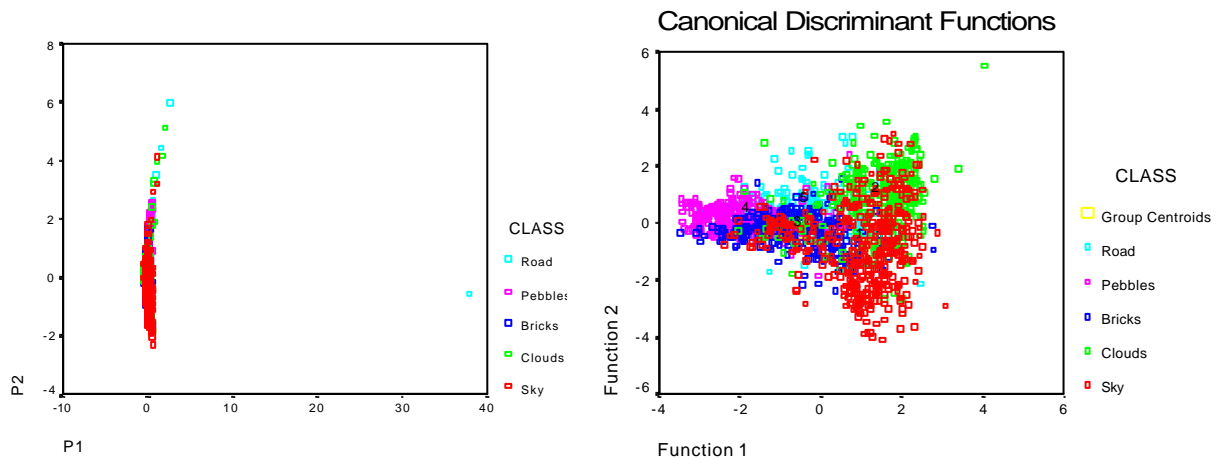


Figure 8.17 PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge and *co-occurrence* features for the discrimination of natural object data.

The above plot shows a much better separation across data samples than what we have seen with co-occurrence matrices in the past. Most of the classes, except for sky, form a very tight cluster but there is significant overlap across the following class combinations: bricks and sky, bricks and road, sky and clouds. An overall recognition rate of 64.7% is obtained. The individual classes are recognised with the following accuracy: sky (81.3%), clouds (58.7%), bricks (52.6%), pebbles (72.6%), and road (48.4%). The majority of the misclassifications can be attributed to sky samples being confused as cloud and vice-versa, bricks confused as confused as sky and road, pebbles confused as road, and road confused as sky.

Edge frequency features

The PCA and canonical discriminant plot is shown in Figure 8.18.

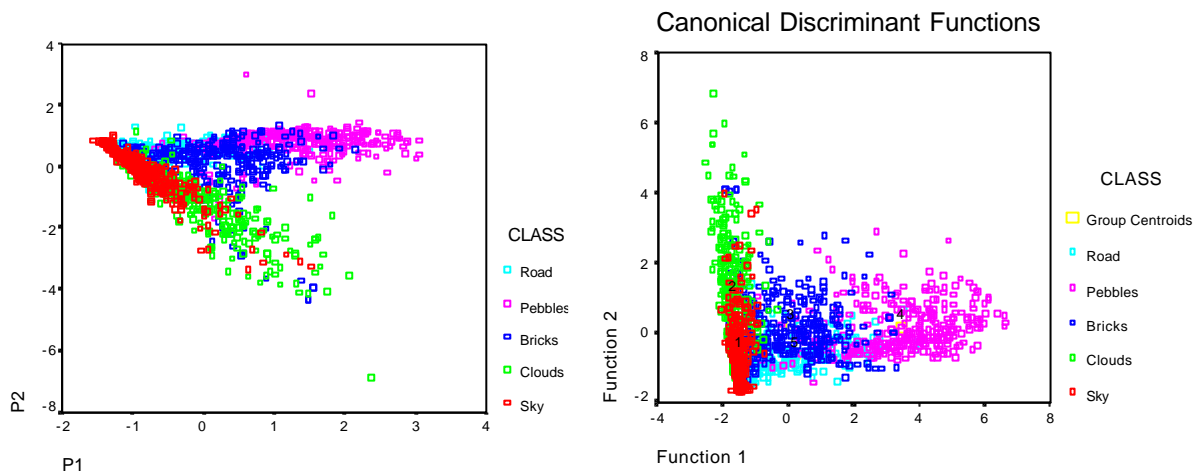


Figure 8.18 PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge segmentation and *edge frequency* features for the discrimination of natural object data.

In these plots it appears that most classes form separate clusters. Bricks are most overlapping on pebbles and road. Sky also overlaps across cloud. We find an overall recognition rate of 69.6%. The individual classes are recognised with the following accuracy: sky (87.8%), clouds (61.9%), bricks (54.3%), pebbles (81.6%), and road (51.6%). The majority of the misclassifications are made when sky samples are confused as clouds and vice-versa, brick samples are confused as sky, clouds and road, pebbles are confused as road, and road samples are confused as sky.

Law's features

The PCA and discriminant function plot are shown in Figure 8.19. In these plots we find that there is significant overlap across different classes. The best recognition performance of 55.8% is obtained with the linear classifier. The individual classes are recognised with the following accuracy: sky (50.1%), clouds (77.5%), bricks (23.3%), pebbles (78.7%) and road (44.4%). We find that most of the mistakes are made when sky samples are confused as clouds and vice-versa, bricks are confused as pebbles or road, pebbles are confused as bricks, and road is confused as clouds and pebbles.

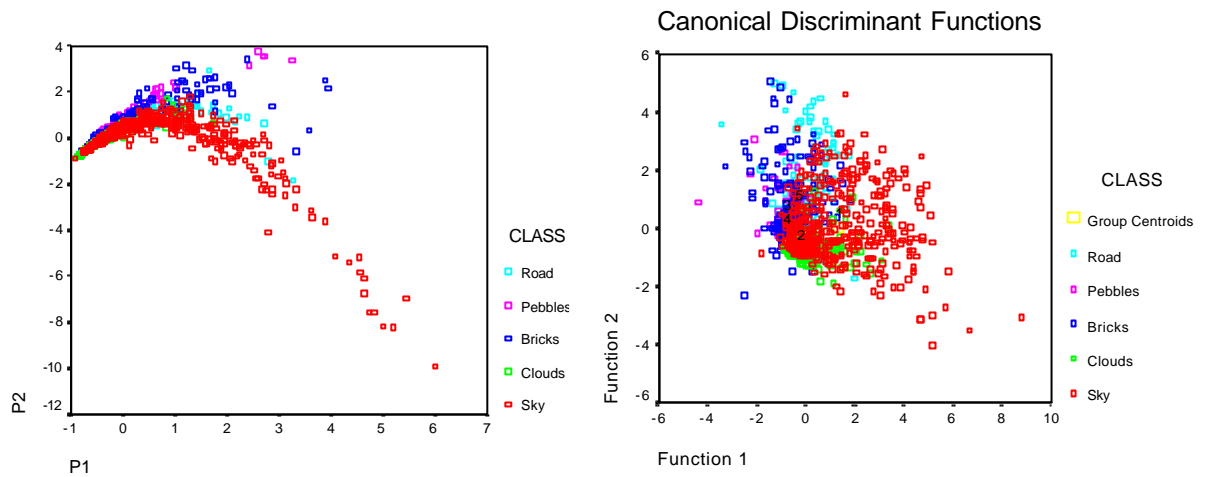


Figure 8.19 PCA plot and canonical discriminant function plots for the linear analysis using Split and Merge and *Law's* features for the discrimination of natural object data.

Primitive length features

The PCA plot is shown in Figure 8.20. The analysis does not yield a discriminant function plot.

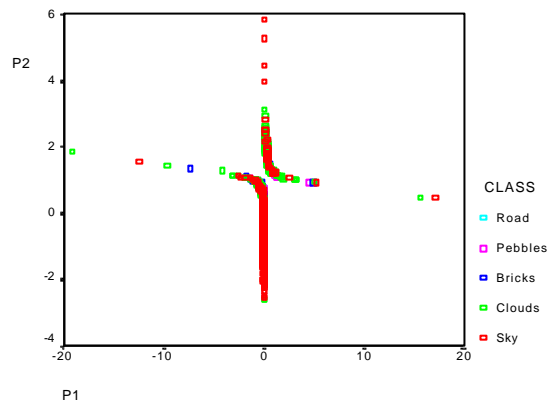


Figure 8.20 PCA plot the linear analysis using Split and Merge and *primitive length* features for the discrimination of natural object data.

The plot shows that there are two symmetrical bands. The linear classifier gives a leave-one-out classification of 44.0% correct. The individual classes are recognised with the following accuracy: sky (56.5%), clouds (67.3%), bricks (24.2%), pebbles (50.0%), and road (5.6%). It appears as before this method is well suited to recognising clouds. However, the method is abysmal at recognising road. The mistakes are evenly spread out for different classes.

Summary

In Table 8.7 we summarise the overall results and highlight which class was best recognised.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	64.7%	Sky (81.3%)
Co-occurrence	60.9%	Clouds (72.0%)
Edge frequency	69.6%	Sky (87.8%)
Law's	55.8%	Pebbles (78.7%)
Primitive length	44.0%	Clouds (67.3%)

Table 8.7 Summary of linear classifier performance with Split and Merge.

The best results are obtained for edge frequency features. These are followed by autocorrelation performance. For different features we find that different classes are the easiest to recognise.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*			*
	<i>Pebbles</i>					*
	<i>Road</i>	**				
<i>Co-occurrence</i>	<i>Sky</i>		*	*		
	<i>Clouds</i>	*				
	<i>Bricks</i>	*			*	*
	<i>Pebbles</i>			*		
	<i>Road</i>			*	*	
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*			*
	<i>Pebbles</i>					*
	<i>Road</i>	**			*	
<i>Law's</i>	<i>Sky</i>		**			*
	<i>Clouds</i>	*			*	
	<i>Bricks</i>				***	*
	<i>Pebbles</i>			*		
	<i>Road</i>		*		**	
<i>Primitive length</i>	<i>Sky</i>		*	*	*	
	<i>Clouds</i>	*			*	
	<i>Bricks</i>	**	*		*	
	<i>Pebbles</i>		**	*		
	<i>Road</i>	***	*	*	*	

Table 8.8 Description of mistakes made by the linear classifier with Split and Merge.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. Table 8.8 shows these mistakes in a more meaningful manner. The pattern of mistakes appears similar for autocorrelation and edge frequency features. Other feature sets show different patterns of mistakes. For autocorrelation and edge frequency method there are hardly any mistakes in favour of classes such as bricks and pebbles. Mistakes appear to be biased in favour of sky class. For co-occurrence features, the mistakes appear to be biased in favour of bricks. In the case of Law's feature set, mistakes are biased in favour of pebbles. In the case of primitive length method, there are several mistakes made as we have found earlier. Most of the samples are attributed to classes other than road that have also got a very poor recognition of only 5.6%.

8.1.5 Summarising linear classification results on natural object data

In the previous sections we grouped results under different segmentation methods that were used to generate the different regions in images. We can summarise the classifier mistakes in the

reverse manner, i.e. how does the classifier make mistakes keeping the same feature extraction method based on the output of different segmentation methods. In this section we aim to show some tables with this information. Also, we present a final table showing the classification results of different segmentation and feature extraction combinations at one place. Let us first summarise how the mistakes are made by the linear classifier for each feature extraction method by changing the preceding segmentation process. For a total of 5 feature extraction methods, Tables 8.9-8.13 are drawn. Finally, Table 8.14 shows which combination of segmentation method with texture extraction method yields the best classification on natural object data.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					*
	<i>Road</i>	**				
<i>Histogram</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					*
	<i>Road</i>	**	*			
<i>Region Growing</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>					**
	<i>Pebbles</i>					*
	<i>Road</i>	**	*			
<i>Split & Merge</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*			*
	<i>Pebbles</i>					*
	<i>Road</i>	**				

Table 8.9 Mistakes made by the linear classifier for *autocorrelation* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		**	*	*	*
	<i>Clouds</i>	*			*	
	<i>Bricks</i>	*	*		*	**
	<i>Pebbles</i>					
	<i>Road</i>				*	
<i>Histogram</i>	<i>Sky</i>		*	*		
	<i>Clouds</i>	*		*		*
	<i>Bricks</i>			*	*	*
	<i>Pebbles</i>			*		
	<i>Road</i>	*	*	*	*	
<i>Region Growing</i>	<i>Sky</i>		*	*		
	<i>Clouds</i>	*		*	*	
	<i>Bricks</i>	*	*		*	*
	<i>Pebbles</i>				*	*
	<i>Road</i>		*	*	*	
<i>Split & Merge</i>	<i>Sky</i>		*	*		
	<i>Clouds</i>	*				
	<i>Bricks</i>	*			*	*
	<i>Pebbles</i>			*		
	<i>Road</i>			*	*	

Table 8.10 Mistakes made by the linear classifier for *co-occurrence* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					
	<i>Road</i>	*				
<i>Histogram</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*			*
	<i>Pebbles</i>					
	<i>Road</i>	**				
<i>Region Growing</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				*
	<i>Pebbles</i>					
	<i>Road</i>	*	*	*	*	
<i>Split & Merge</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*			*
	<i>Pebbles</i>					*
	<i>Road</i>	**			*	

Table 8.11 Mistakes made by the linear classifier for *edge frequency* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		**			
	<i>Clouds</i>		*		*	*
	<i>Bricks</i>		**			
	<i>Pebbles</i>		**		*	
	<i>Road</i>		**			
<i>Histogram</i>	<i>Sky</i>		**			
	<i>Clouds</i>		**		*	*
	<i>Bricks</i>		**	*	*	*
	<i>Pebbles</i>		**		*	
	<i>Road</i>		**		*	
<i>Region Growing</i>	<i>Sky</i>		*			
	<i>Clouds</i>		*		*	*
	<i>Bricks</i>		**		*	
	<i>Pebbles</i>		**		*	
	<i>Road</i>		**		*	
<i>Split & Merge</i>	<i>Sky</i>		**			*
	<i>Clouds</i>	*			*	*
	<i>Bricks</i>			*	***	*
	<i>Pebbles</i>				*	
	<i>Road</i>		*		**	

Table 8.12 Mistakes made by the linear classifier for *Law's* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		**	**	*	*
	<i>Clouds</i>	*				
	<i>Bricks</i>	*			*	*
	<i>Pebbles</i>	***		*		
	<i>Road</i>	*		***	*	
<i>Histogram</i>	<i>Sky</i>		**			***
	<i>Clouds</i>				*	*
	<i>Bricks</i>	*	*		*	**
	<i>Pebbles</i>		**			*
	<i>Road</i>	*	*			
<i>Region Growing</i>	<i>Sky</i>		*	**	*	*
	<i>Clouds</i>				*	
	<i>Bricks</i>				*	*
	<i>Pebbles</i>	*	*	*		
	<i>Road</i>	*		**		
<i>Split & Merge</i>	<i>Sky</i>		*	*	*	
	<i>Clouds</i>	*			*	
	<i>Bricks</i>	**	*		*	
	<i>Pebbles</i>		**	*		
	<i>Road</i>	***	*	*	*	

Table 8.13 Mistakes made by the linear classifier for *primitive length* features.

We can draw the following conclusions from Tables 8.9 to 8.13.

- i) For the autocorrelation feature extraction, the mistakes across the use of different segmentation methods is very similar. The mistakes are biased in favour of sky, clouds and road classes. No matter which segmentation method is used, during the classification of the feature set, hardly any mistakes are made attributing samples wrongly to bricks or pebbles class.
- ii) For co-occurrence features, different preceding segmentation methods yield different results. The mistakes are spread across different classes in a uniform manner.
- iii) For edge frequency features, the results are very similar to the autocorrelation features except for those obtained with region growing segmentation.
- iv) For Law's feature set, mistakes are highly biased in the favour of clouds. There are hardly any mistakes made in favour of sky. The first three segmentation methods, FCM, histogram based segmentation, and region growing, are similar in their performances. Split and merge behaves quite differently from these.
- v) For primitive length, just as with co-occurrence matrices, there does not appear to be consistency in mistakes across different segmentation methods.

The overall performance of the classification scheme is shown in Table 8.14. This is also shown as a plot in Figure 8.21. From Table 8.14, the following conclusions can be drawn.

- i) The best average recognition rate performance across the four segmentation methods is that of edge frequency with the average recognition performance of 69.7% correct. The single best performance is of edge frequency with FCM of 71% correct.
- ii) The best average performance across the rows (for all segmentation methods using the different feature extraction methods), is that of region growing with an average 59.9% correct.

- iii) In terms of variability in performance, texture algorithms can be ranked as follows in order of decreasing variability: Law's, primitive length, co-occurrence, autocorrelation and edge frequency. If we are to consider the best texture method that has the least variability, then edge frequency can be considered the best. Similarly the segmentation methods can be ranked in order of decreasing variability as follows: Histogram thresholding, FCM, Region growing and Split and merge.

Feature \ Segmentation	Autocorrelation	Co-occurrence	Edge frequency	Law's	Primitive length	μ	σ
FCM	69.1%	51.1%	71.0%	58.1%	38.7%	57.6%	13.3%
Histogram	63.0%	53.6%	69.7%	53.4%	31.9%	54.3%	14.3%
Region Grow	64.3%	55.2%	68.6%	67.5%	43.8%	59.9%	10.4%
Split & Merge	64.7%	60.9%	69.6%	55.8%	44.0%	59.0%	9.8%
μ	65.3%	55.2%	69.7%	58.7%	39.6%	-	-
σ	2.6%	4.2%	1.0%	6.2%	5.7%	-	-

Table 8.14 The different linear classifier performance depending on which data set is used for natural object data analysis.

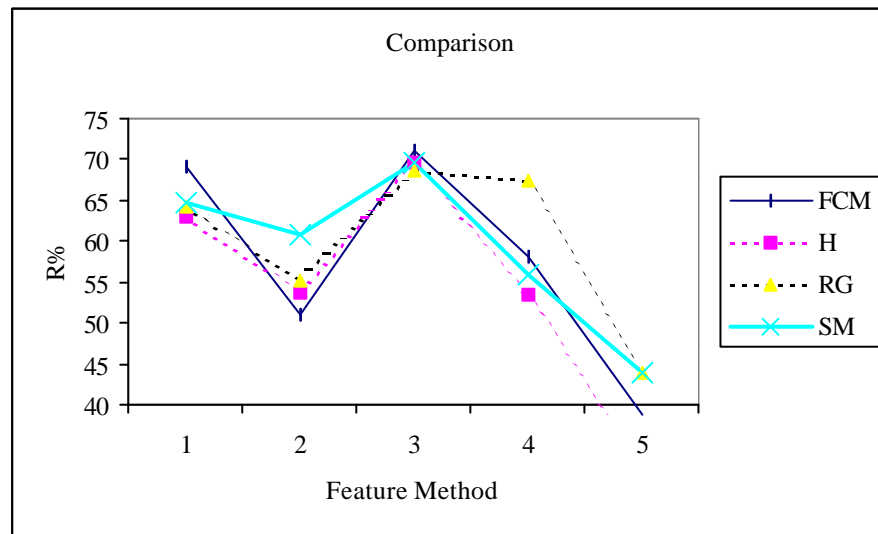


Figure 8.21 A graphical comparison of linear classifier recognition accuracy obtained using different segmentation method and texture method combinations for natural data analysis.

From Figure 8.21 it can be seen that all segmentation methods have similar performances for edge frequency features. On the whole, the two best segmentation algorithms appear to be region growing and split and merge.

8.2 Nearest neighbour classification

We now show the results obtained using the nearest neighbour classifier. As with our earlier discussion, we group results on a per segmentation method basis and at the end collate results within each feature extraction category and provide a summary. The results have been generated using the leave-one-out procedure.

8.2.1 Fuzzy c-means clustering segmentation

In this section we discuss experimental results obtained for FCM segmentation.

Autocorrelation features

The results for these are shown in Appendix N. The best results are obtained using model-1 ($k=5$). The recognition rate of 69.4% is slightly better than the linear classifier performance of 69.1%. We also find that the individual classes are recognised with the following accuracy: sky (71.7%), clouds (76.5%), bricks (61.3%), pebbles (81.8%), and road (51.9%). Compared to the linear classifier, we are better at recognising clouds by 12%, and poorer at recognising the sky by 10%. On other classes, the performances are more or less the same.

Co-occurrence features

The best performance is achieved using the model-1 classifier ($k=3$). The recognition accuracy of 52.2% is slightly better than the linear classifier result of 51.1%. The following accuracy is achieved on recognising individual classes: sky (55.5%), clouds (46.7%), bricks (42.1%), pebbles (48.6%), and road (50.7%). Majority of the mistakes are made when sky and cloud are confused as the other, and road is confused as sky. The other mistakes are balanced over all classes. Compared to the linear classifier, the k NN classifier is better at recognising sky by nearly 15%, and bricks by 8%. It is however worse at recognising other classes such as clouds by 9%, pebbles by 30% and road by 10%.

Edge frequency features

The k NN classifier achieves the best performance with model-1 ($k=5$). The recognition rate of 69.2% is slightly inferior to the linear classifier performance of 71.0% correct. In terms of individual classes, they are recognised with the following accuracy: sky (72.3%), clouds (77.7%), bricks (47.4%), pebbles (90.1%), and road (52.6%). The misclassifications can be attributed to confusion between sky and cloud samples, bricks confused as clouds, pebbles or road, and road confused as sky or bricks. Compared to the linear classifier, we find that the nearest neighbour classifier is better at recognising clouds by nearly 16%. It is however worse off at recognising sky by 12%, bricks by 6%, pebbles by 3%, and road by 8%.

Law's features

The best performance of the nearest neighbour classifier is achieved with model-1 ($k=7$). The recognition rate of 59.8% correct is a slight improvement on the 58.1% correct classification obtained with the linear classifier. The individual classes can be recognised with the following accuracy: sky (63.8%), clouds (79.7%), bricks (49.6%), pebbles (59.5%), and road (29.6%). The majority of the mistakes are made when sky and cloud samples are confused as the other, bricks are confused as pebbles and road, pebbles are confused as road and road is confused as everything else except bricks. Compared to the linear classifier, k NN is better at recognising sky by 13% and pebbles by 11%. It is however poor at recognising others including clouds by 10%, bricks by 1% and road by 5%.

Primitive length features

The best performance is achieved using model-1 ($k=3$). The classification success of 49.8% correct is superior to the linear classifier performance of 38.7% correct. The individual classes

are recognised with the following accuracy: sky (43.3%), clouds (61.9%), bricks (35.7%), pebbles (49.5%) and road (55.9%). The majority of the mistakes are biased in favour of sky and bricks. The k NN classifier is better than the linear classifier in recognising sky by 26%, pebbles by 24%, and road by 40%, but poor at recognising clouds by 11% and bricks by 22%.

Summary

In Table 8.15 we summarise the overall results and highlight which class was best recognised.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	69.4%	Pebbles (81.8%)
Co-occurrence	52.2%	Sky (55.5%)
Edge frequency	69.2%	Pebbles (90.1%)
Law's	59.8%	Clouds (79.7%)
Primitive length	49.8%	Clouds (61.9%)

Table 8.15 Summary of nearest neighbour classifier performance for FCM.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>				**	
	<i>Pebbles</i>					
	<i>Road</i>	**				
<i>Co-occurrence</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*		*	*
	<i>Pebbles</i>	*				
	<i>Road</i>	*	*			
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>		*		*	*
	<i>Pebbles</i>					
	<i>Road</i>	*		*		
<i>Law's</i>	<i>Sky</i>		**			
	<i>Clouds</i>	*				*
	<i>Bricks</i>				**	*
	<i>Pebbles</i>					*
	<i>Road</i>	*	**		*	
<i>Primitive length</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	**				
	<i>Bricks</i>	**			*	*
	<i>Pebbles</i>	*		*		*
	<i>Road</i>	*		*	*	

Table 8.16 Description of mistakes made by the nearest neighbour classifier with FCM.

The best results are obtained for the autocorrelation features. These are followed by the edge frequency performance. For different features we find that different classes are the easiest to recognise but pebbles and clouds appear to be most distinguishable.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. Table 8.16 shows these mistakes in a more meaningful manner. The performance of each feature set is found to be different than others.

The following conclusions can be drawn from the above table. Most of the feature sets make different kinds of mistakes. Autocorrelation feature set makes the least number of mistakes which confirms it as being the best for the overall analysis. Co-occurrence model is biased heavily in favour of sky and clouds when making misclassifications. Both of these feature sets and Law's method do not make any errors misclassifying patterns as of type bricks.

8.2.2 Histogram thresholding segmentation

In this section we discuss experimental results obtained on all feature sets based on regions generated by the histogram thresholding method. The results for these are shown in Appendix P. *Autocorrelation features*

The best results are obtained for model-1 ($k=5$). The recognition rate of 62.5% is similar to the 63.0% correct recognition achieved by the linear classifier. The individual classes are recognised with the following accuracy: sky (78.3%), clouds (51.8%), bricks (53.3%), pebbles (86.8%), and road (42.8%). The worst mistakes are made when sky samples are confused as cloud and vice-versa, bricks are confused as pebbles or road, pebbles are confused as road or when road is confused as sky. Compared to the linear classifier, the k NN classifier is better at recognising pebbles by roughly 6% and worse off at recognising sky by 4%. Other classes have very similar recognition accuracy across the two classifiers.

Co-occurrence features

The best recognition rate is obtained using model-1 ($k=7$). The recognition success of 53.7% is the same as that achieved using the linear classifier of 53.6%. The distribution of mistakes is however different across the two classifier. The k NN individual class accuracy are: sky (53.6%), clouds (57.7%), bricks (56.3%), pebbles (46.9%), and road (47.8%). The mistakes are evenly distributed across different classes. The worst ones are made when sky and cloud are confused. Compared to the linear classifier, sky is recognised better by 4%, bricks by 4%, and road by 13%. However, the k NN classifier is poor at recognising clouds by 4% and pebbles by 34%.

Edge frequency features

The best performance is obtained with model-1 ($k=3$). The recognition rate of 65.8% is slightly inferior to the linear classifier performance of 69.7% correct. The individual classes are recognised with the following accuracy: sky (77.0%), clouds (65.0%), bricks (43.4%), pebbles (86.8%), and road (61.7%). The majority of the misclassifications can be attributed to sky and cloud samples confused as the other, bricks confused as road, and road confused as bricks and sky. Compared to the linear classifier, the k NN classifier is better at recognising road by roughly 2%. It is poor on everything else: we are worse off on sky by 10%, clouds by 2%, bricks by 5%, and pebbles by 3%.

Law's features

k NN model-1 ($k=7$) achieves a best recognition performance of 52.3% correct. This is not much different than the linear classifier performance of 53.4% correct recognition. The individual classes are recognised with the following accuracy: sky (71.8%), clouds (46.2%), bricks (60.2%), pebbles (25.4%) and road (24.4%). Sky and cloud samples are often confused. Also, bricks are likely to be confused as all other categories, pebbles primarily as bricks or road, and road as all other categories except pebbles. Compared to the linear classifier, this classifier is better at recognising sky samples by a margin of 13%, bricks by a margin of 30%. It is however worse off on the other categories with an inferior performance margin of 42% for clouds, pebbles by 11% and road by 4%.

Primitive length features

The best recognition performance of 45.3% correct is achieved using model-1 ($k=5$). This is much better than the linear classification result of 31.9% correct. The individual classes are recognised with the following accuracy: sky (45.6%), clouds (48.5%), bricks (44.1%), pebbles (34.2%), and road (47.4%). The mistakes are evenly spread across different categories. Compared to the linear classifier we find that the k NN classifier is superior at recognising sky by 39% and bricks by 33%. It is however worse off on the other categories by the following margins: for cloud 16%, for pebbles 15% and for road 6%.

Summary

In Table 8.17 we summarise the overall results and highlight which class was best recognised.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	62.5%	Pebbles (86.8%)
Co-occurrence	53.7%	Clouds (57.7%)
Edge frequency	65.8%	Pebbles (86.8%)
Law's	52.3%	Sky (71.8%)
Primitive length	45.3%	Clouds (48.5%)

Table 8.17 Summary of nearest neighbour classifier performance for Histogram Thresholding.

In Table 8.17 the best results are obtained for the edge frequency features. For different features we find that different classes are the easiest to recognise but pebbles and clouds appear to be most distinguishable. The difference in performances can be categorised under three groups. The first group consisting of autocorrelation and edge frequency shows good performance above 60%. In the second group we find Law's and co-occurrence that have performances above 50%, and finally primitive length method that has performance of more than 40%.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. Table 8.18 shows these mistakes in a more meaningful manner. We find in this table that autocorrelation and edge frequency methods make similar mistakes. For the first three feature extraction methods, most of the mistakes are biased in favour of the first three classes of sky, clouds and bricks. The primitive length feature method shows the majority of the mistakes.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>				*	*
	<i>Pebbles</i>					
	<i>Road</i>	**	*			
<i>Co-occurrence</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				
	<i>Pebbles</i>	*	*	**		*
	<i>Road</i>	*	*	*		
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>		*		*	**
	<i>Pebbles</i>					
	<i>Road</i>	*		*		
<i>Law's</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**		*		*
	<i>Bricks</i>				*	*
	<i>Pebbles</i>		*	**		*
	<i>Road</i>	*	**	*		
<i>Primitive length</i>	<i>Sky</i>		*	*		*
	<i>Clouds</i>	**		*		
	<i>Bricks</i>	**	*		*	*
	<i>Pebbles</i>	*	*	*		*
	<i>Road</i>	*		*		

Table 8.18 Description of mistakes made by the linear classifier for Histogram segmentation.

8.2.3 Region growing segmentation

In this section we discuss experimental results obtained on all feature sets based on regions generated by the region growing method. The results for these are shown in Appendix R.

Autocorrelation features

The best performance of 66.8% correct for this feature set is achieved using model 1 k NN ($k=3$) which is better than the linear classifier performance of 64.3% correct. The individual classes are recognised with the following accuracy: sky (60.9%), clouds (63.0%), bricks (67.8%), pebbles (75.3%), and road (33.9%). The major mistakes are localised as follows: sky and cloud samples are confused as the other, bricks are confused as pebbles and road, pebbles are confused as the road and road samples are confused as sky. When compared to the linear classifier, we find that the k NN classifier has a better accuracy on recognising clouds by a margin of 5%, bricks by 6%. Pebbles and road are recognised with roughly the same accuracy but the k NN classifier is poorer on recognising sky by a margin of 18%.

Co-occurrence features

The best recognition performance of 51.7% is obtained using model-1 ($k=7$). This is inferior than the linear classifier recognition rate of 55.2%. The individual classes are recognised with the following accuracy: sky (66.7%), clouds (57.9%), bricks (45.0%), pebbles (37.2%), and road (11.7%). The classifier has particular problems when classifying road samples as they are mostly confused as sky or bricks. Bricks are also mostly confused as sky or cloud. When compared to the linear classifier we find that the nearest neighbour method is better at recognising sky by a margin of 8% but it is worse off on recognising pebbles by a margin of 39% and road by a margin of 22%. The recognition on clouds and bricks is nearly the same.

Edge frequency features

The best recognition performance of 67.1% is achieved using model-1 ($k=7$). The result is poorer than the linear classifier performance of 68.6%. The individual classes are recognised with the following accuracy: sky (82.8%), clouds (69.3%), bricks (53.8%), pebbles (86.1%), and road (22.0%). Most of the mistakes are made when sky and cloud samples as confused as the other and bricks are confused as cloud or pebbles or road, and when road is confused as bricks. Compared to the linear classifier we are better at recognising clouds by a margin of 4%, and bricks by 7%. It is poorer in recognition by a rough margin of 7% on sky, by 6% on pebbles and by 15% on roads.

Law's features

The overall best result is obtained with model-1 ($k=5$). The recognition rate of 62.1% compares poorly with the linear classifier's performance of 67.5%. The classes are recognised with the following accuracy: sky (68.4%), clouds (77.2%), bricks (67.8%), pebbles (30.7%), and road (16.9%). The mistakes are evenly spread across the first three classes. Compared to the linear classifier, kNN is better at recognising bricks by 17%. Otherwise it performs inferior on all other classes with the following margins: sky by 3%, clouds by 16%, pebbles by 27% and road by 15%.

Primitive length features

The best recognition rates are obtained for model-1 ($k=5$). The overall performance of 48.1% is much better than 43.8% correct performance obtained using the linear classifier. The individual classes are recognised with the following accuracy: sky (39.5%), clouds (67.1%), bricks (47.5%), pebbles (32.3%), and road (38.9%). The misclassifications can be attributed to sky and cloud samples confused as the other, bricks confused as all other classes, pebbles confused as sky and bricks and road confused as sky or bricks. Compared to the linear classifier, the kNN classifier is better at recognising sky by 31% and road by 6%. On other classes the performance is poorer by 8% for clouds, 7% for bricks and 2% for pebbles.

Summary

In Table 8.19 we summarise the overall results and highlight which class was best recognised. The best results for region growing based segmentation are achieved using edge frequency as shown in Table 8.19. The performance is not much different compared to autocorrelation method or Law's feature set performance. Co-occurrence and primitive length features continue to perform poorly on this problem.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	66.8%	Pebbles (75.3%)
Co-occurrence	51.7%	Sky (66.7%)
Edge frequency	67.1%	Pebbles (86.1%)
Law's	62.1%	Clouds (77.2%)
Primitive length	48.1%	Clouds (67.1%)

Table 8.19 Summary of nearest neighbour classifier performance for Region Growing.

Another manner in which the result can be summarised is to consider where the classifier makes most of the mistakes. Table 8.20 shows these mistakes in a more meaningful manner. We find in this table that autocorrelation and edge frequency methods make similar mistakes. Also, co-occurrence and Law's feature sets have similar pattern of mistakes. In general, most mistakes are made in favour of clouds as they contain an element of texture that is found in other categories.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>				*	*
	<i>Pebbles</i>					*
	<i>Road</i>	**	*	*		
<i>Co-occurrence</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>	*	*			
	<i>Pebbles</i>		*	*		*
	<i>Road</i>	**	*	**	*	
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>		*		*	*
	<i>Pebbles</i>					
	<i>Road</i>		*	*		
<i>Law's</i>	<i>Sky</i>		**			
	<i>Clouds</i>	*				
	<i>Bricks</i>		*		*	
	<i>Pebbles</i>		*	**		*
	<i>Road</i>		**	*	*	
<i>Primitive length</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	*				
	<i>Bricks</i>	*	*		*	
	<i>Pebbles</i>	**		*		
	<i>Road</i>	**		**		

Table 8.20 Description of mistakes made by the nearest neighbour classifier for Region Growing.

8.2.4 Split and merge segmentation

In this section we discuss experimental results obtained on all feature sets based on regions generated by the split and merge method. The results for these are shown in Appendix T.

Autocorrelation features

The best recognition performance of 65.3% correct is achieved using the nearest neighbour model-1 with $k=7$. This is slightly better than the linear classification performance of 64.7% correct. The individual classes are recognised with the following accuracy: sky (75.7%), clouds (60.0%), bricks (51.0%), pebbles (83.8%), and road (46.9%). The majority of the mistakes are made when sky and cloud samples are confused as the other, bricks are confused as pebbles or road, pebbles are confused as the road, and road is confused as sky or pebbles.

Co-occurrence features

The best results using the nearest neighbour classifier are achieved with model-1 ($k=7$). This performance of 55.6% correct is only much worse than the linear classification of 64.7%. The individual classes are recognised with the following accuracy: sky (52.5%), clouds (64.0%), bricks (57.3%), pebbles (64.8%), and road (33.0%). The mistakes are balanced across different classes. The comparison with the linear classifier shows that we are now better at recognising clouds and bricks by a rough margin of 5%, but worse off on other by rough margins of 29% for sky, 8% for pebbles and 15% for road.

Edge frequency features

For these features, the k NN model with $k=1, 3$ and 7 neighbours yields the same performance of 64.3% correct. For the sake of discussion, we use the single nearest neighbour model as the best. This result is slightly inferior to the linear classification accuracy of 69.6% correct. The different classes are recognised with the following accuracy: sky (71.0%), clouds (60.6%), bricks (50.3%), pebbles (81.9%) and road (51.6%). The majority of the mistakes are made when sky and cloud samples are confused as the other, bricks are confused as road or pebbles, pebbles are confused as road and when road is confused as bricks, pebble and sky. Compared to the linear classifier, the k NN classifier is worse off at recognising sky by a rough margin of 16%. The rest of the classes are recognised with similar accuracy.

Law's features

The best recognition performance of 50.4% is achieved using model-1 nearest neighbour with $k=7$. This is slightly inferior to the linear classifier performance of 55.8% correct. The classes involved are recognised with the following accuracy: sky (59.3%), clouds (57.7%), bricks (36.4%), pebbles (53.8%), and road (32.1%). The mistakes are quite evenly spread across different classes. Compared to the linear classifier, k NN is better at recognising sky by a rough margin of 9% and bricks by 13%, however, it is worse off at recognising clouds by 20%, pebbles by 25% and road by 12%.

Primitive length features

The best recognition performance of this feature set is obtained using model-1 ($k=7$). The recognition rate of 50.1% is significantly better than the linear classification rate of 44.0% correct. The classes are recognised with the following accuracy: sky (42.7%), clouds (61.3%),

bricks (40.0%), pebbles (69.0%) and road (33.9%). The mistakes are quite evenly spread out across different classes. Compared to the linear classifier, we get better performances on recognising bricks by 16%, pebbles by 19%, and road by 27%. However the performance is inferior for sky by 14%, and clouds by 6%.

Summary

In Table 8.21 we summarise the overall results and highlight which class was best recognised.

<i>Feature method</i>	<i>Recognition</i>	<i>Easiest to recognise</i>
Autocorrelation	65.3%	Pebbles (83.8%)
Co-occurrence	55.6%	Pebbles (64.8%)
Edge frequency	64.3%	Pebbles (81.9%)
Law's	50.4%	Sky (59.3%)
Primitive length	50.1%	Pebbles (69.0%)

Table 8.21 Summary of the nearest neighbour classifier performance for Split and Merge.

<i>Feature method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>Autocorrelation</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>				*	*
	<i>Pebbles</i>					
	<i>Road</i>	**			*	*
<i>Co-occurrence</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	*			*	
	<i>Bricks</i>	*				
	<i>Pebbles</i>			*		
	<i>Road</i>	*	*	*	*	
<i>Edge frequency</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>				*	*
	<i>Pebbles</i>					
	<i>Road</i>	*		*	**	
<i>Law's</i>	<i>Sky</i>		*			*
	<i>Clouds</i>	*			*	
	<i>Bricks</i>				**	*
	<i>Pebbles</i>		*	*		
	<i>Road</i>	*	*	*	*	
<i>Primitive length</i>	<i>Sky</i>		*	*		*
	<i>Clouds</i>	*				
	<i>Bricks</i>	**	*		*	*
	<i>Pebbles</i>					
	<i>Road</i>	**		*	*	

Table 8.22 Description of mistakes made by the nearest neighbour classifier for Split and Merge.

The best results are obtained for autocorrelation feature set closely followed by the edge frequency method. Again we find that these two methods are one level up compared to the others. Also, pebbles tend to be the easiest to distinguish.

In Table 8.22 it does not appear that there is any particular pattern of mistakes that are made for the five different feature extraction methods. For autocorrelation, mistakes are hardly made assigning data to bricks incorrectly and for co-occurrence no mistakes are made in favour of road. There is however some similarity between the pattern of misclassifications for edge frequency and autocorrelation.

8.2.5 Summarising nearest neighbour classification results on natural object data

In the previous sections we grouped results under different segmentation methods that were used to generate the different regions in images. We can summarise the classifier mistakes in the reverse manner, i.e. how does the classifier make mistakes keeping the same feature extraction method based on the output of different segmentation methods. In this section we aim to show some tables with this information. Also, we present a final table showing the classification results of different segmentation and feature extraction combinations at one place. Let us first summarise how the mistakes are made by the nearest neighbour classifier for each of the feature extraction methods by changing the preceding segmentation process. For a total of 5 feature extraction methods, Tables 8.23-8.27 are drawn. Finally, Table 8.28 shows which combination of segmentation and texture extraction methods yields the best classification on natural object data.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>				**	
	<i>Pebbles</i>					
	<i>Road</i>	**				
<i>Histogram</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**			*	*
	<i>Bricks</i>					
	<i>Pebbles</i>		*			
	<i>Road</i>	**	*			
<i>Region Growing</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**			*	*
	<i>Bricks</i>					*
	<i>Pebbles</i>		*	*		*
	<i>Road</i>	**	*	*		
<i>Split & Merge</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**			*	*
	<i>Bricks</i>					
	<i>Pebbles</i>				*	*
	<i>Road</i>	**			*	*

Table 8.23 Mistakes made by the nearest neighbour classifier for *autocorrelation* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>	*	*		*	*
	<i>Pebbles</i>	*				
	<i>Road</i>	*	*			
<i>Histogram</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	**				
	<i>Bricks</i>	*				
	<i>Pebbles</i>	*	*	**		*
	<i>Road</i>	*	*	*		
<i>Region Growing</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>	*	*			
	<i>Pebbles</i>		*	*		*
	<i>Road</i>	**	*	**	*	
<i>Split & Merge</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	*			*	
	<i>Bricks</i>	*				
	<i>Pebbles</i>			*		
	<i>Road</i>	*	*	*	*	

Table 8.24 Mistakes made by the nearest neighbour classifier for *co-occurrence* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>		*		*	*
	<i>Pebbles</i>					
	<i>Road</i>	*		*		
<i>Histogram</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>		*		*	**
	<i>Pebbles</i>					
	<i>Road</i>	*		*		
<i>Region Growing</i>	<i>Sky</i>		*			
	<i>Clouds</i>	*				
	<i>Bricks</i>		*		*	*
	<i>Pebbles</i>					
	<i>Road</i>		*	*		
<i>Split & Merge</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**				
	<i>Bricks</i>				*	*
	<i>Pebbles</i>					
	<i>Road</i>	*		*	**	

Table 8.25 Mistakes made by the nearest neighbour classifier for *edge frequency* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		**			
	<i>Clouds</i>	*				*
	<i>Bricks</i>				**	*
	<i>Pebbles</i>					*
	<i>Road</i>	*	**		*	
<i>Histogram</i>	<i>Sky</i>		*			
	<i>Clouds</i>	**		*		*
	<i>Bricks</i>			*	*	*
	<i>Pebbles</i>		*	**		*
	<i>Road</i>	*	**	*		
<i>Region Growing</i>	<i>Sky</i>		**			
	<i>Clouds</i>	*				
	<i>Bricks</i>		*		*	
	<i>Pebbles</i>		*	**		*
	<i>Road</i>		**	*	*	
<i>Split & Merge</i>	<i>Sky</i>		*			*
	<i>Clouds</i>	*			*	
	<i>Bricks</i>				**	*
	<i>Pebbles</i>		*	*		
	<i>Road</i>	*	*	*	*	

Table 8.26 Mistakes made by the nearest neighbour classifier for *Law's* features.

<i>Segmentation method</i>	<i>Class</i>	<i>Sky</i>	<i>Clouds</i>	<i>Bricks</i>	<i>Pebbles</i>	<i>Road</i>
<i>FCM</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	**				
	<i>Bricks</i>	**			*	*
	<i>Pebbles</i>	*		*		*
	<i>Road</i>	*		*	*	
<i>Histogram</i>	<i>Sky</i>		*	*		*
	<i>Clouds</i>	**		*		
	<i>Bricks</i>	**	*		*	*
	<i>Pebbles</i>	*	*	*		*
	<i>Road</i>	*		*		
<i>Region Growing</i>	<i>Sky</i>		**	*		
	<i>Clouds</i>	*				
	<i>Bricks</i>	*	*		*	
	<i>Pebbles</i>	**		*		
	<i>Road</i>	**		**		
<i>Split & Merge</i>	<i>Sky</i>		*	*		*
	<i>Clouds</i>	*				
	<i>Bricks</i>	**	*		*	*
	<i>Pebbles</i>					
	<i>Road</i>	**		*	*	

Table 8.27 Mistakes made by the nearest neighbour classifier for *primitive length* features.

The following conclusions can be drawn from Tables 8.23 to 8.27.

- i) The number of mistakes made by the autocorrelation method is smaller in comparison with other methods. The FCM and histogram segmentation based mistake patterns are similar, and the two region growing based methods are similar. The mistakes are biased in favour of sky class.
- ii) Co-occurrence features yield many more mistakes, and for them the first three segmentation methods show similar performances. The mistakes are biased in favour of sky and clouds. However these two classes are least likely to be confused as others.
- iii) For edge frequency measures, the mistakes are almost identical across the first three segmentation methods, and even the fourth segmentation method does not give too many dissimilar mistakes. Bricks are most likely to be confused as something else. Road is the second most confused class.
- iv) For Law's feature set, all segmentation methods totally different patterns of mistakes. Sky, pebbles and road have a high risk of being misclassified.
- v) For primitive length, the majority of the mistakes are made in favour of class sky. Bricks, pebbles, and road are most likely to be misclassified. All methods yield similar patterns of mistakes except for region growing.

The overall performance of the classification scheme is shown in Table 8.28. This is also shown as a plot in Figure 8.22.

Feature Segmentation	Autocorrelation	Co-occurrence	Edge frequency	Law's	Primitive length	μ	σ
FCM	69.4%	52.2%	69.2%	59.8%	49.8%	62.6%	9.2%
Histogram	62.5%	53.7%	65.8%	52.3%	45.3%	58.5%	8.2%
Region Grow	66.8%	51.7%	67.1%	62.1%	48.1%	61.9%	8.8%
Split & Merge	65.3%	55.6%	64.3%	50.4%	50.1%	58.9%	7.3%
μ	66.0%	53.3%	66.6%	56.2%	48.3%	-	-
σ	2.9%	1.7%	2.1%	5.6%	2.2%	-	-

Table 8.28 The different nearest neighbour classifier performance depending on which data set is used for natural object data analysis.

The following conclusions can be drawn from Table 8.28.

- i) The best overall performance is that of autocorrelation features using FCM at 69.4% correct recognition.
- ii) We can rank the feature extraction methods on the basis of their variability across different segmentation methods. In decreasing order of variability these are Law's, autocorrelation, primitive length, edge frequency, and co-occurrence matrices. If we are to consider a texture method that is robust to the changes in segmentation process, we can safely say that co-occurrence matrices and edge frequency are the best. Similarly we can rank the segmentation methods in decreasing order of variability for the use of different texture methods as follows: FCM, region growing, histogram thresholding and split and merge.

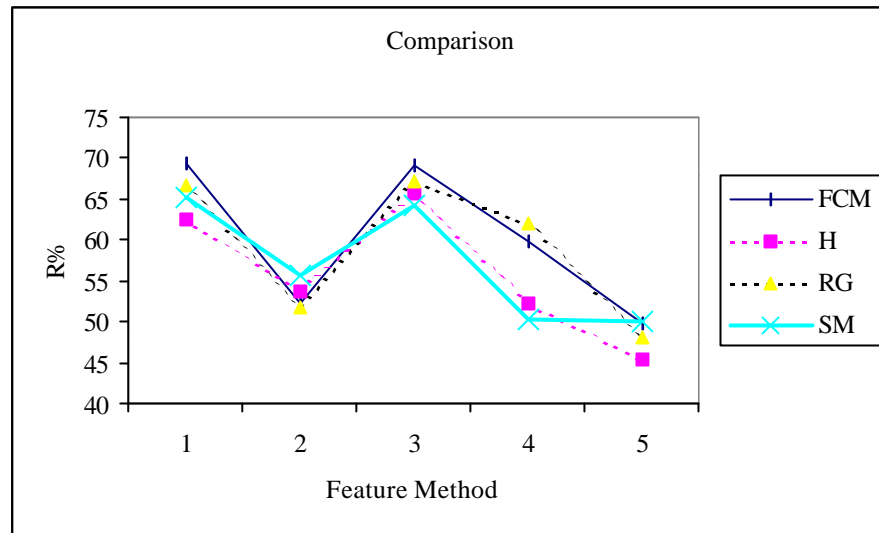


Figure 8.22 A graphical comparison of recognition accuracy obtained using different segmentation method and texture method combinations for natural data analysis.

It is not easy to discriminate which segmentation method is the overall best performer. We illustrate briefly how we have determined the overall best winner. We do this by ranking each method as follows in Table 8.29.

f_1	f_2	f_3	f_4	f_5
FCM	SM	FCM	RG	FCM
RG	H	RG	FCM	SM
SM	FCM	H	H	RG
H	RG	SM	SM	H

Table 8.29 Ranking the different segmentation programs based on different feature sets f_1 (autocorrelation) to f_5 (primitive length).

We can average out the rankings to get these final scores: FCM (2.7), H (3.2), RG (2.4) and SM (2.8). Hence, on the basis of the lower the score, the better the method, we can rank them in order of importance as region growing as follows: FCM, split and merge and histogram thresholding.

8.3 Conclusions

Some very brief conclusions are in order after writing this chapter. More detailed conclusions and discussion follows in the next chapter. For a complex recognition task as ours, it would be unrealistic to assume too much out of feature extraction methods or classifiers. For a five class problem, the best that can be achieved by chance is only 20% and therefore the best recognition performance of nearly 70% is impressive. However, this does not mean that further improvements are not possible. We discuss this in our next and final chapter. In this whole analysis we have used original data without any processing for improving recognition rates. This was done to make the study comparative. Further improvements can be made by better feature selection, data preprocessing and through the use of other non-linear classifiers. We find from this chapter that the nearest neighbour classifier did not make much of an improvement as expected because of the nature of the data. However, as we discuss in the next chapter,

significant improvements based on principal component data are possible. Another important observation we have made is the presence of significant difference across different segmentation and feature extraction method combinations. This difference adds weight to the fundamental hypothesis of this thesis that optimisation of these processes on their own in an image processing pipeline will not necessarily produce good results, i.e. it is the combination, of these processes that is extremely crucial.

Chapter 9

Conclusions

Writing this thesis has been a journey. The primary goal of the thesis was to explore the effect of using different image segmentation and texture extraction algorithms on natural scene recognition. Also in this study we showed comparative results across different texture extraction algorithms on two popular texture databases. As a part of this exploration there have been challenges and successes, but above all the satisfaction of having some answers that were not available at the beginning of the thesis. Of course the number of choices available for different methodologies possible for evaluating images are enormous and it has not been possible to test all of them. Also, even though the database for natural images that we have developed is large, yet limited in size, we must be cautious in generalising our results. However, the study has managed to address some important issues at a scale that the results have meaning and even though we have not tried many more algorithms for benchmarking, as would be ideal, we have used some of the most popular methods available generating useful conclusions.

In this chapter we present some key conclusions of this work. Detailed results on our chosen experimental design have been presented in the previous chapters. We first discuss the importance of the work and how well this study addresses issues of relevance to image understanding and scene analysis research. We next discuss how further improvements can be made on our work. In other words, how we can achieve better recognition accuracy on our data. Next, we quote some of our key results from our previous chapters on benchmarking texture databases and PANN scene analysis benchmark. Some key conclusions from our analysis are drawn. Finally, we summarise some salient observations from this work.

9.1 Scene analysis in perspective

The analysis of images is important from a variety of perspectives. In some cases we are primarily interested in recognising its components. In others we aim to derive a better understanding of the relationship between these components. Our study has been devoted to the first objective. The recognition of various objects in natural scene images is a complex problem. The complexity of the problem derives from the intrinsic nature of real images. In such images, the image environment is dynamically changing and as such the role of light in terms of its reflectance and absorption with different natural objects plays an important role. Texture of such objects, especially when dealing with grey level images alone, depends on the brightness of pixels which in turn is determined by the surface properties and lighting conditions. Furthermore, for natural object recognition, the number of different characteristic descriptors for objects is limited. As such, based on texture alone, the identification of such objects is quite difficult. As image processing hardware has been considerably expensive in the past, the role of colour has also been quite limited restricting most of the past studies in this area to grey level image analysis. Parker[161] states that although only 256 grey levels are possible in a typical grey image, there are 65536 times this number of possible colours. With these limitations, it becomes very important that the choice of image segmentation and texture analysis algorithms is as close to optimal as possible in order to achieve reasonable results. As mentioned earlier,

what is reasonable is debatable. It depends on the purpose of the study. Imagine a situation where our sole purpose of analysis is to determine if the image contains vegetation. If we can recognise trees and grass with near complete accuracy, and get a less than 10% accuracy on everything else, then we have achieved reasonable results. On the other hand if our objective was to recognise all objects with a reasonable accuracy, we fail to meet this objective.

Scene analysis, though complex, is the basic building block of future autonomous systems. An autonomous system must be able to acquire images by itself, process them, and make some decisions on what its understanding of these images is. The first step is that of recognising the objects within the image and the second step is that of interpreting the scene as a whole. Better recognition of objects for such a computer-based system depends on the following factors: a) accurate region definition; b) accurate texture definition of these regions; c) intelligent data recognition strategies; d) use of *a priori* information on object relationships. The image understanding scheme can be visualised as a pipeline where each of these components is usually plugged in order. The system is fed an input stream of images that are analysed using low level image processing operators. The output efficiency of this pipeline is based on the overall recognition accuracy of the system based on some cross-validation scheme. It is, in most applications, important to get the best output from this pipeline. One of the schemes involves that each component of the pipeline is optimised on its own. In other words, we evaluate how good texture algorithms are on some benchmark images, and evaluate segmentation algorithms in isolation and choose the best ones. Unfortunately, such evaluations are quite difficult on real data. For example, evaluating segmentation algorithms on real natural images is quite difficult. It is unlikely that such a scheme of work will yield the best recognition results. In most studies, however, this is mostly what is done. Algorithms considered as the best from previous experience or on the basis of other studies are plugged in a scene analysis or object recognition system without an exhaustive testing of different combinations. Obviously the number of possible combinations can be quite large. Consider a total of N steps during the image processing. Each process p has a maximum of choices n_i . The total number of possible combinations is:
$$\prod_{i=1}^N n_i$$
.

Some of the important processes that need optimisation include image enhancement, image segmentation, feature extraction, and classification. In literature there are dozens of good options for each of these and it is only common sense that any study must be selective in terms of what to use. In some cases, such as for enhancement, quantitative methods have been developed to select the optimal algorithms[194]. Thus, they can be eliminated from the combinatorial exhaustive testing. However, the other three steps need exhaustive combinatorial testing because of the following reasons: a) different segmentation processes yield different regions that give different texture measures for the same feature extraction method; b) different texture feature sets are best suited for classification by different classifiers because of the nature of data distributions. By the same token, different classifiers will output different kinds of mistakes on the same data and hence show different recognition performance.

In this thesis we do not preach exhaustive testing of data sets for all image analysis applications but demonstrate the variability across results if different algorithm combinations are used for processing data. As such our results are quite important in demonstrating this variability. Only

when it has been accepted that such variability exists, and that is important when designing image understanding solution, we can then begin to address it. This area has not been explored in enough detail by other studies on scene analysis. Only when we begin to synthesise this problem in more detail, novel solutions of optimising the algorithmic pipeline can be found that are computationally feasible. These solutions will invariably lead to better quality image understanding and scene analysis systems.

9.2 Further improvement on recognition rates

The purpose of this study was primarily on comparing different image analysis algorithms on the basis of the final recognition accuracy on identifying natural objects. For an evenly based comparison, it has not been possible to incorporate optimal improvements possible on each feature set. For example, as we have mentioned earlier, it was decided not to remove outliers from the feature sets, as for different feature sets we get different outliers. The same applies to feature selection as different features derived from the same texture algorithm will be more discriminatory for different data sets. One of the disadvantages of this limitation has been that we can not claim that the results that we have generated are the best. Obviously several improvements are possible and we list some of them that are the most obvious. Further extension to this work will investigate these more thoroughly.

a) Improvements based on outlier removal

Outliers are those patterns that are the farthest away from the mean feature vector of a given class. These can be removed in several ways. Either we can take out those patterns that are further away from the mean by a certain standard deviation of a class distribution. We can also remove the outliers by ranking how far away these samples are from the mean and simply taking away a certain percentage of the most distant samples. Outlier removal not necessarily improves the performance. Only those outliers that are scattered into the region dominated by other classes are useful to remove.

b) Improvements based on using PCA data

Principal component analysis is an important methodology for reducing data dimensionality. This reduction does not mean that the complexity of the data classification problem is much reduced. The principal components are based on their ability to explain the overall variability in the data set. Since each principal component is a weighted average of different features, features that are the least variable, and in some ways redundant, are given much less weight than highly variable features. To some extent, PCA scores can be used as a better set of features than the original variables. We have thus also produces results on PCA scores to demonstrate that better results can be obtained using them, especially for the nearest neighbour classifier. We hope that by this process we can still compare the different algorithmic combinations on an even basis.

In Table 9.1 we first demonstrate the results obtained on the MeasTex data using PCA scores for nearest neighbour classification. So, how does this compare to the results obtained by the same classifier on the original data as discussed in Chapter 5. On autocorrelation, we had the best result of 79.5% correct and now we have 86.1% correct, an improvement of nearly 7%. On co-occurrence features, we had a previous best of 86.9% that has now improved to 93.5%, an improvement of 7% again. For edge frequency we had on original data recognition of 70.7%

that has now improved by 4% to 74.9%. On Law's features, there is hardly any change from 70.9% to 69.3%. On primitive length features we get a better performance as results improve from 54.1% correct to 55.9%. On combined features, there is hardly any change in performance from 83.3% correct to 83.6%. On the whole we can conclude that the use of PCA features improves the performance on most feature sets and has very little impact on the others.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	84.0%	85.9%	85.8%	86.1%
Co-occurrence	93.2%	93.4%	93.5%	93.4%
Edge frequency	71.7%	73.7%	74.9%	73.8%
Laws	61.6%	66.4%	68.2%	69.3%
Primitive length	52.3%	55.5%	55.5%	55.9%
Combined	82.8%	82.8%	83.6%	83.4%

Table 9.1 MeasTex PCA data classification using nearest neighbour classifier.

In Table 9.2 we show the results of the nearest neighbour classifier with PCA data of VisTex database.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	91.4%	90.4%	88.6%	80.4%
Co-occurrence	93.6%	86.4%	81.8%	82.9%
Edge frequency	73.6%	76.1%	74.3%	73.2%
Laws	47.5%	52.9%	53.2%	52.5%
Primitive length	56.1%	55.4%	54.3%	55.7%
Combined	84.6%	85.0%	82.9%	82.5%

Table 9.2 VisTex PCA data classification using nearest neighbour classifier.

Let us compare these to the original data classification rates: autocorrelation results improve from 85.7% to 91.4% (by nearly 6%), co-occurrence results improve from 80.7% to 93.6% (by nearly 13%), edge frequency results improve from 66.8% to 76.1% (by nearly 10%), Law's results are slightly worse from 56.1% to 53.2% (by nearly 3%), primitive length features improve from 42.4% to 56.1% (by nearly 14%) and combined features result improves from 61.3% to 85.0% (by nearly 24%). This is a significantly good performance of the k NN classifier on PCA data.

We can draw some important conclusions from the above. First, we find that on MeasTex data the maximum improvement is that of 7% on autocorrelation and co-occurrence feature sets. On other features sets there is not much difference in performance. To put the 7% increment in perspective when dealing with 944 samples, it means a total of 66 samples recognised correctly that were otherwise misclassified. Second, on VisTex data we find that improvements are of much larger magnitude. The most significant improvement is that of 24% on combined features. This implies 67 samples correctly recognised that would have been otherwise misclassified out of a total of 280 samples. Law's is the only feature set where the performance declines marginally. On all other feature sets, the performance increases considerably.

The performance improvements on the vegetation data using the four different segmentations methods are shown in the next four tables. In Table 9.3, we show the change in performance with PCA data classification. For autocorrelation method, the recognition rate improves from 72.9% to 74.6% (by nearly 2%), for co-occurrence features it improves from 56.5% to 59.5% (by 3%), for edge frequency features the performance improves from 72.0% to 74.8% (by nearly 3%), for Law's it deteriorates from 74.6% to 69.6% (by nearly 5%), and for primitive length features it improves from 57.6% to 60.5% (by nearly 3%). Hence, we find that except for Law's feature set, the performance improves on the whole.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	70.5%	73.1%	74.6%	73.9%
Co-occurrence	53.0%	56.1%	57.4%	59.5%
Edge frequency	70.0%	73.4%	74.8%	74.6%
Laws	63.2%	67.9%	68.1%	69.6%
Primitive length	60.3%	60.5%	58.7%	56.9%

Table 9.3 PCA data classification using nearest neighbour classifier on vegetation data recognition with FCM.

The results on Histogram Thresholding are shown in Table 9.4.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	67.0%	70.1%	70.6%	71.2%
Co-occurrence	56.7%	56.7%	59.8%	58.9%
Edge frequency	69.0%	72.8%	74.4%	76.1%
Laws	54.8%	57.5%	58.5%	59.1%
Primitive length	58.3%	55.5%	53.2%	51.4%

Table 9.4 PCA data classification using nearest neighbour classifier on vegetation data recognition with Histogram Thresholding.

The performance changes are as follows. For autocorrelation features the performance improves from 70.0% to 71.2% (by nearly 1%), for co-occurrence features it improves from 57.3% to 59.8% (by more than 2%), for edge frequency features it improves from 70.0% to 76.5% (by more than 6%), for Law's features it remains the same at 59.1%, and on primitive length features it improves from 55.6% to 58.3% (by nearly 3%).

In Table 9.5 we show the results for Region Growing based segmentation. In this case, the performance changes are as follows. For autocorrelation features, the performance improves from 58.2% to 64.2% (by 6%), co-occurrence performance improves from 52.3% to 55.5% (more than 3%), edge frequency performance improves from 65.0% to 73.9% (nearly 9%), for Law's it remains the same at 64.0%, and for primitive length it improves from 56.7% to 64.2% (nearly 8%).

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	61.1%	64.2%	64.2%	62.7%
Co-occurrence	55.5%	52.3%	49.1%	54.1%
Edge frequency	71.3%	72.6%	73.9%	73.6%
Laws	60.6%	61.6%	62.4%	64.0%
Primitive length	60.8%	61.9%	63.7%	64.2%

Table 9.5 PCA data classification using nearest neighbour classifier on vegetation data recognition with Region Growing.

In Table 9.6, we show the results for Split and Merge segmentation.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	63.4%	65.8%	68.0%	68.1%
Co-occurrence	62.4%	65.6%	68.0%	69.9%
Edge frequency	77.4%	80.6%	81.2%	82.0%
Laws	68.2%	71.8%	73.2%	73.5%
Primitive length	63.1%	60.9%	62.6%	63.4%

Table 9.6 PCA data classification using nearest neighbour classifier on vegetation data recognition with Split and Merge.

We find that once more better results are obtained using PCA data than the original data. The performances improve as follows. For autocorrelation features, it improves from 64.2% to 68.1% (by nearly 6%), for co-occurrence features it improves from 66.2% to 69.9% (by more than 3%), for edge frequency features it improves from 74.0% to 82.0% (by 8%), for Law's features it remains the same at 73.5% and for primitive length method it improves from 55.6% to 63.4% (by nearly 8%).

The following conclusions can be drawn from the above analysis. Region growing shows the largest difference in magnitude between the performances with the original data and the PCA data and FCM shows the least difference. However these percentages must be understood cautiously. Since the different feature sets have different number of samples for different segmentation methods, the percentages translates into different number of actual samples that are correctly recognised. For region growing and edge frequency combination, we have a total of 383 samples, so a 9% better performance translates into 35 samples correctly recognised that were misclassified with original data. This appears to be significant improvement. We next present the results on classifying natural object data using PCA scores. These are again classified as per the segmentation method.

In Table 9.7 we show the results obtained using FCM clustering segmentation. The following changes in performance are noticeable. The recognition rate for autocorrelation features improves from 69.4% to 71.3% correct (by nearly 2%), for co-occurrence features it improves from 52.2% to 57.5% (by nearly 5%), for edge frequency it improves from 69.2% to 73.8% (by nearly 4%). For Law's features it remains the same at 59.8% and for primitive length it changes from 49.8% to 53.5% (by nearly 4%).

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	66.6%	69.6%	71.3%	69.4%
Co-occurrence	52.8%	55.0%	56.9%	57.5%
Edge frequency	70.0%	71.8%	73.8%	72.8%
Laws	53.1%	56.7%	58.0%	59.8%
Primitive length	51.7%	52.5%	53.5%	52.7%

Table 9.7 PCA data classification using nearest neighbour classifier on natural object data recognition with FCM.

We show the results of Histogram Thresholding in Table 9.8.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	61.3%	63.0%	64.7%	65.2%
Co-occurrence	47.2%	50.1%	53.6%	54.1%
Edge frequency	67.6%	69.8%	69.6%	70.3%
Laws	44.8%	49.1%	51.6%	52.3%
Primitive length	49.1%	48.0%	50.1%	49.2%

Table 9.8 PCA data classification using nearest neighbour classifier on natural object data recognition with Histogram Thresholding.

The performance difference compared to the original data is as follows. Recognition rate improves for autocorrelation features from 62.5% to 65.2% (by nearly 3%), it improves for co-occurrence features from 53.7% to 54.1% (by less than 1%), for edge frequency features it improves from 65.8% to 70.3% (by nearly 5%), for Law's features it remains the same at 52.3% and for primitive length features it improves from 45.3% to 50.1% (nearly 5%).

We show the results for region growing segmentation in Table 9.9.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	67.1%	68.9%	69.1%	69.5%
Co-occurrence	55.5%	57.1%	56.9%	56.9%
Edge frequency	70.5%	72.5%	72.2%	71.6%
Laws	56.0%	57.8%	62.1%	60.8%
Primitive length	58.4%	57.1%	60.0%	59.7%

Table 9.9 PCA data classification using nearest neighbour classifier on natural object data recognition with Region Growing.

In this case the performance improvement is higher. The following improvements are noticeable: for autocorrelation the recognition improves from 66.8% to 69.5% (by nearly 3%), for co-occurrence features it remains the same at 57.1%, for edge frequency it improves from 67.1% to 72.5% (by nearly 5%), for Law's it remains the same at 62.1% correct, and finally for primitive length we see a significant improvement of nearly 12% from 48.1% correct to 60.0% correct.

Finally, for split and merge segmentation, the results are shown in Table 9.10.

Feature extraction	k=1	k=3	k=5	k=7
Auto-correlation	62.1%	64.4%	65.9%	66.2%
Co-occurrence	52.6%	56.3%	57.0%	59.5%
Edge frequency	69.5%	72.7%	73.5%	74.2%
Laws	47.3%	49.7%	51.4%	52.9%
Primitive length	59.2%	59.7%	61.7%	60.4%

Table 9.10 PCA data classification using nearest neighbour classifier on natural object data recognition with Split and Merge.

The performances are different this time as follows. For autocorrelation features, the performance improves from 65.3% to 66.2% (nearly 3%), for co-occurrence matrices it improves from 55.6% to 59.5% (by nearly 3%), for edge frequency it improves from 64.3% to 74.2% (by nearly 10%), for Law's feature set they improve from 50.4% to 52.9%, and finally for primitive length features they improve from 50.1% to 61.7% (by nearly 12%).

The following conclusions can be drawn from the above. First, we find that larger improvements are made with region growing and split and merge methods of segmentation. Second, we find that the largest improvement of 12% for primitive length using split and merge and the second largest of 10% for edge frequency features for split and merge. These translated to the number of samples now correctly identified but previously misclassified amounts to 185 and 154 samples respectively.

c) Improvements based on feature selection

One of the reasons for using a scheme of feature selection is to remove redundant or unnecessary information, thereby improving the recognition rates. There are two ways in which the results can be improved. The first involves the selection of d most discriminatory variables out of p possible. This is termed as feature selection. The other involves the transformation of p measurements to a lower dimensional space. This is termed as feature extraction [219]. This may involve a linear or non-linear transformation of the original variables. As seen earlier, we have used PCA for improving our results. However, in the future, further improvements are possible using feature selection. For a total of d variables, and selection of subsets of size p , the total number of possible subsets is equal to

$$\frac{d!}{(p-d)!d!}$$

Obviously, the number of subsets to be evaluated based on testing all subsets exhaustively is an enormous task. As such, a number of strategies have been proposed in the literature for selecting the best combination of features without testing all subsets. The selection of features can be based on two criteria. First, we can design a classifier based on a reduced feature set and choose the feature set for which the classifier performs the best on a separate test/validation set. Second, instead of the above, we can estimate the overlap between distributions from which the data are drawn and favour those feature sets for which this overlap is minimal. This process is independent of the classifier used. It has the advantage of computational ease of

implementation, however, the separation measures can be crude and unreliable. A range of probability based distances are used in various studies. These distances include Bhattacharya distance, Chernoff distance, Patrick Fisher distance and Divergence. In addition to these methods, search strategies can also be used for effective subset selection. Some of these procedures include branch and bound procedure, best individual N method, Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), and Floating Search methods.

d) Improvements based on multistage classification

As we mentioned in our experimental design, one way of improving results is to develop a multistage classification strategy. In such a strategy, multiple classifiers can be trained to separate out those classes first that are easily separable. It has been shown that such a strategy can yield better performances. Our analysis has shown that different classifiers are superior at recognising different classes involved in our data. Also as we mentioned earlier, if two sets of data distributions, both linearly separable themselves, overlap each other, the resultant data is no longer linearly separable. This problem is better resolved with multistage classification.

e) Improvements based on non-linear classifiers

Neural networks have been widely recognised as an important class of non-linear classifiers. One of the main reasons for not including them in this study has been the limited amount of time we had for the research. Also since we are dealing with very large number of data sets, optimising neural networks would normally take a very long time. Hence, nearest neighbour classifiers, often shown to perform as good as neural networks, were selected for our analysis. However, there is scope for further improvement should non-linear classifiers be applied on our data.

f) Improvements based on using colour information

One of assumptions made in this study was that colour information is too expensive to process and hence we have excluded that from analysis. This will not true be for long as image processing hardware becomes much cheaper and faster. Colour has much of a role to play either in the form of colour features for different image regions, or for acting as a classifier triggering mechanism for a multistage classifier strategy. It is only common sense that as humans we use most colour information for object recognition and that computers should have the same advantage.

9.3 Key results

In Table 9.11 we present the best results obtained on the four sets of data considered in this thesis. The results are compared across original and PCA data.

<i>Data</i>	<i>Segmentation</i>	<i>Texture</i>	<i>Classifier</i>	<i>Data Type</i>	<i>Result</i>
MeasTex	-	Co-occurrence	k NN	PCA	93.5%
VisTex	-	Combined	Linear	Original	94.6%
PANN Vegetation	Split & Merge	Edge frequency	k NN	PCA	82.0%
PANN Natural Object	Split & Merge	Edge frequency	k NN	PCA	74.2%

Table 9.11 The best results on different data sets.

We can draw the following conclusions from the results presented in the thesis.

- As discussed earlier, we found that PCA data yields better results.
- In general the use of nearest neighbour classifier improves the quality of results. The recognition accuracy for the four data sets in the above table is impressive considering the complexity of the task.
- Split and merge algorithm shows the best performance.
- There is a considerable variability in recognition performances depending on the choice of segmentation method used. These are summed up on vegetation analysis as follows from Table 7.28: autocorrelation (6.5%), co-occurrence (5.8%), edge frequency (3.8%), Law's (7.5%) and primitive length (5.3%). For natural object data analysis, these are: autocorrelation (2.9%), co-occurrence (1.7%), edge frequency (2.1%), Law's (5.6%) and primitive length (2.2%). So in both cases Law's feature set is the most affected, and edge frequency one of the least affected.
- In terms of computational time taken for image segmentation, split and merge takes the longest and histogram thresholding takes the least.
- In terms of the total number of regions generated for PANN benchmark, we have in order: split and merge (2485 samples), histogram thresholding (2045 samples), FCM (1811 samples) and region growing (1013 samples).
- Primitive length features yield the worst results in general but are one of the best on recognising clouds.
- Texture algorithms get different ranks in order of how well they perform on synthetic data as compared to their real data performance.
- The combined feature set has some advantage in classification and must be explored in further studies.
- We have successfully demonstrated that there is a large range of performances depending on the choice of segmentation and texture methods used in combination. Based on a detailed study on the pattern of mistakes made, and inspection of cases where mistakes are made, image analysis can be further improved.

9.4 Summary

In this study we have compared a range of texture analysis and image segmentation algorithms for understanding the differences in their performances on the same data. We have found that there is considerable variability in the performances of different texture analysis algorithms. Also we find that these algorithms perform much better on synthetic textures as compared to real data. The definition of regions in natural images is an important factor in the quality of features extracted from them. Regions that are poorly segmented yield texture measures contaminated by pixels that come from two or more regions. We have found that different combination of segmentation and texture extraction methods yield different recognition accuracy and the variability in these results is significant. Only by optimising for this fact, high quality scene analysis systems can be developed.

References

1. R. Adams and L. Bischof, Seeded region growing, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, 1994.
2. J.K. Aggarwal and S. Shah, Object recognition and performance bounds, Lecture Notes in Computer Science: Image Analysis and Processing, A. Bimbo (eds.), Springer, pp. 343-360, 1997.
3. D.W. Aha and R.L. Bankert, A comparative evaluation of sequential feature selection algorithms, in *Learning from data: AI and statistics*, D. Fisher and H.J. Lenz (Eds.), Springer, pp. 199-206, 1996.
4. N. Ahuja, A. Rosenfeld and R.M. Haralick, Neighbour gray levels as features in pixel classification, *Pattern Recognition*, vol. 12, pp. 251-260, 1980.
5. N. Ahuja and A. Rosenfeld, Mosaic models for textures, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 1, pp. 1-11, 1981.
6. A. Al-Janobi, Performance evaluation of cross-diagonal texture matrix method of texture analysis, *Pattern Recognition*, vol. 34, pp. 171-180, 2001.
7. M. Asada and Y. Shirai, Building a world model for a mobile robot using dynamic semantic constraints, Proc. 11th International Joint Conference on Artificial Intelligence, pp. 1629-1634, vol. II, Detroit, 1989.
8. M.F. Augustejin, Performance evaluation of texture measures for ground cover identification in satellite images by means of a neural-network classifier, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, pp. 616-625, 1995.
9. R. Azencott, J.P. Wang and L. Younes, Texture classification using windowed Fourier filters, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 149-153, 1997.
10. R. Bajcsy and A.K. Joshi, A partially ordered world model and natural outdoor scenes, in *Computer Vision Systems*, A.R. Hanson and E.M. Riseman (eds.), pp. 263-270, Academic Press, New York, 1978.
11. A. Baraldi and F. Parmigianni, An investigation of the textural characteristics associated with gray level co-occurrence matrix statistical parameters, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 2, pp. 293-302, 1995.
12. S. Basu, Image segmentation by semantic method, *Pattern Recognition*, pp. 497-511, 1987.
13. J. Batlle, A. Casals, J. Freixenet and J. Marti, A review on strategies for recognising natural objects in colour images of outdoor scenes, *Image and Vision Computing*, vol. 18, pp. 515-530, 2000.
14. J.M. Beaulieu and M. Goldberg, Hierarchy in picture segmentation: a stepwise optimisation approach, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 150-163, 1989.
15. D.C. Becalick, Natural scene classification using a weightless neural network, *PhD Thesis*, Department of Electrical and Electronic Engineering, Imperial College, London, 1996.
16. M. Betke and N.C. Makris, Information-conserving object recognition, *Technical Report* no. CS-TR-3799, Computer Vision Laboratory, University of Maryland, 1997.
17. B. Bhanu and O.D. Faugeras, Segmentation of images having unimodal distributions, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 4, no. 4, pp. 408-419, 1982.

18. B. Bhanu and B.A. Parvin, Segmentation of natural scenes, *Pattern Recognition*, vol. 20, no. 5, pp. 487-496, 1987.
19. B. Bhanu, S. Lee and J. Ming, Adaptive image segmentation using a genetic algorithm, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 12, pp. 1543-1567, 1995.
20. R.L. Blankert, Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network, *Journal of Applied Meteorology*, vol. 33, pp. 909-918, 1994.
21. C.M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, 1995.
22. A.C. Bovik, M. Clark and W. S. Geisler, Multichannel texture analysis using localised spatial filters, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 12, no. 1, 1990.
23. J.P. Braquelaire and L. Brun, Image segmentation with topological maps and inter-pixel representation, *Journal of Visual Communication and Image Representation*, vol. 9, no. 1, pp. 62-79, 1998.
24. C.R. Brice and C.L. Fennema, Scene analysis using regions, *Artificial Intelligence*, vol. 1, pp. 205-226, 1970.
25. P. Brodatz, *Textures: a photographic album for artists and designers*, Dover publications, New York, 1966.
26. R.A. Brooks, Model based 3D interpretation of 2D images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 140-150, 1983.
27. J.M.H. Buf, M. Kardan and M. Spann, Texture feature performance for image segmentation, *Pattern Recognition*, vol. 23, no. 3/4, pp. 291-309, 1990.
28. M. Campani, M. Capello, G. Piccioli, E. Reggi, Visual routines for outdoor navigation, Proc. of Intelligent Vehicles Symposium, pp. 107-112, Tokyo, Japan, 1993.
29. N.W. Campbell, W.P.J. Mackeown, B.T. Thomas and T. Troscianko, Automatic interpretation of outdoor scenes, British Machine Vision Conference, Birmingham, UK, 1995.
30. N.W. Campbell, W.P.J. Mackeown, B.T. Thomas and T. Troscianko, The automatic classification of outdoor images, Proc. International Conference on Engineering Applications of Neural Networks, Systems Engineering Association, pp. 339-342, 1996.
31. N.W. Campbell, B.T. Thomas and T. Troscianko, Automatic segmentation and classification of outdoor images using neural networks, *International Journal of Neural Systems*, vol. 8, no. 1, pp. 137-144, 1997a.
32. N.W. Campbell, W.P.J. Mackeown, B.T. Thomas, and T. Troscianko, Interpreting image databases by region classification. *Pattern Recognition*, vol. 30, no. 4, pp. 555-563, 1997b.
33. F.H.Y. Chan, F. K. Lam and H. Zhu, Adaptive thresholding by variational method, *IEEE Transactions on Image Processing*, vol. 2, no. 3, pp. 168-174, 1998.
34. Y.L. Chang and X. Li, Adaptive image region growing, *IEEE Transactions on Image Processing*, vol. 3, no. 6, pp. 868-873, 1994.
35. B. B. Chaudhuri, P. Kundu and N. Sarkar, Detection of gradation of oriented texture, *Pattern Recognition Letters*, vol. 14, pp. 147-153, 1993.
36. Y.Q. Chen, M.S. Nixon and D.W. Thomas, Statistical geometric features for texture classification, *Pattern Recognition*, vol. 28, no. 4, pp.537- 552, 1995.
37. Y.Q. Chen, Novel techniques for image texture classification, *PhD Thesis*, Department of Electronics and Computer Science, University of Southampton, 1995.

38. C.C. Chen and C.C. Chen, Filtering methods for texture discrimination, *Pattern Recognition Letters*, vol. 20, pp. 783-790, 1999.
39. C.C. Chen and D.C. Chen, Multi-resolutional Gabor filter in texture analysis, *Pattern Recognition Letters*, vol. 17, pp. 1069-1076, 1996.
40. P.C. Chen and T. Pavlidis, Segmentation by texture using correlation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 1, pp. 64-68, 1981.
41. M. Cheriet, J. N. Said and C. Y. Suen, A recursive thresholding technique for image segmentation, *IEEE Transactions on Image Processing*, vol. 7, no. 6, pp. 918-920, 1998.
42. K. Cho and P. Meer, Image segmentation from consensus information, *Computer Vision and Image Understanding*, vol. 68, no. 1, pp. 72-89, 1997.
43. K. Cho, P. Meer and J. Cabrera, Performance assessment through bootstrap, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1185-1198, 1997.
44. W.S. Chou, Classifying image pixels into shaped, smooth, and textured points, *Pattern Recognition*, vol. 32, pp. 1697-1706, 1999.
45. M.L. Comer and E.J. Delp, Segmentation of textured images using a multi-resolution Gaussian autoregressive model, *IEEE Transactions on Image Processing*, vol. 8, no. 3, pp. 408-420, 1999.
46. R.W. Connors and C.A. Harlow, A theoretical comparison of texture algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 3, pp. 204-222, 1980.
47. A. Dailianas, Comparison of automatic video segmentation algorithms, in *SPIE Photonics West*, vol. 2615, pp. 2-16, Philadelphia, 1995.
48. K.J. Dana, B. van Ginneken, S.K. Nayar and J.J. Koenderink, Reflectance of texture of real world surfaces, *ACM Transactions on Graphics*, vol. 18, no. 1, pp. 1-34, 1999.
49. L.S. Davis, S.A. Johns and J.K. Aggarwal, Texture analysis using generalised co-occurrence matrices, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 3, pp. 251-259, 1979.
50. L.S. Davis, M. Clearman and J.K. Aggarwal, An empirical evaluation of generalised co-occurrence matrices, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 2, pp. 214-221, 1981.
51. L.S. Davis, Image texture analysis techniques - a survey, *Digital Image Processing*, Simon and R. M. Haralick (eds.), pp. 189-201, 1981.
52. P.A. Devijver, A note on ties in voting with the k-NN rule, *Pattern Recognition*, vol. 10, pp. 297-298, 1978.
53. P.A. Devijver and J. Kittler, *Pattern recognition: a statistical approach*, Prentice Hall, London, 1982.
54. R.J. Douglass, Interpreting three dimensional scenes: a model building approach, *Computer Graphics, Vision and Image Processing*, vol. 17, pp. 91-113, 1981.
55. B.A. Draper, R.T. Collins, J. Brolio, A.R. Hanson and E.M. Riseman, The scheme system, *International Journal of Computer Vision*, vol. 2, pp. 209-250, 1989.
56. R.C. Dubes and A.K. Jain, Clustering techniques: the user's dilemma, *Pattern Recognition*, vol. 8, pp. 247-260, 1976.
57. R.C. Dubes, How many clusters are best?-an experiment, *Pattern Recognition*, vol. 20, no. 6, pp. 645-663, 1987.

58. W. Efenberger and V Graefe, Distance invariant object recognition in natural scenes, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1433-1439, Osaka, Japan, 1996.
59. J.O. Eklundh, H. Yamamoto and A. Rosenfeld, A relaxation method for multispectral pixel classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 1, pp. 72-75, 1980.
60. A. Farago, T. Linder and G. Lugosi, Fast nearest neighbour search in dissimilarity spaces, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 957-962, 1993.
61. O.D. Faugeras and W.K. Pratt, Decorrelation methods of texture feature extraction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 4, pp. 323-333, 1980.
62. S. Fioravanti, R. Fioravanti, F.G. DeNatale, R. Marik, M. Mirmehdi, J. Kittler, and M. Petrou, *European Transactions on Telecommunications*, vol. 6, no. 3, pp. 287--300, 1995.
63. G.L. Foresti, Outdoor scene classification by a neural tree-based approach, *Pattern Analysis and Applications*, vol. 2, pp. 129-142, 1999.
64. H. Frigui and R. Krishnapuram, A robust competitive clustering algorithm with applications in computer vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 450-465, 1999.
65. K.S. Fu and J.K. Mui, A survey on image segmentation, *Pattern Recognition*, vol. 13, pp. 3-16, 1981.
66. K. Fukunaga and T.E. Flick, Classification error for a very large number of classes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 779-788, 1984.
67. K. Fukunaga and J.M. Mantock, Nonparametric data reduction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 115-118, 1984.
68. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd edition, Academic Press, 1990.
69. D. Gabor, Theory of communication, *Journal of Institute of Electrical Engineers*, vol. 93, pp. 429-459, 1946.
70. A. Gagalowicz, A new method for texture fields synthesis: some applications of the study of human vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 5, pp. 520-533, 1981.
71. P. Gamba, R. Lodola and A. Mecocci, Scene interpretation by fusion of segment and region information, *Image and Vision Computing*, vol. 15, pp. 499-509, 1997.
72. J.P. Gambotto, A new approach to combining region growing and edge detection, *Pattern Recognition Letters*, vol. 14, pp. 869-875, 1993.
73. P. Garcia, M. Petrou and S. Kamata, The use of boolean model for texture analysis of grey images, *Computer Vision and Image Understanding*, vol. 74, no. 3, pp. 227-235, 1999.
74. R.C. Gonzalez and R.E. Woods, *Digital image processing*, Addison Wesley World Student Series, 1994.
75. E. Gose, R. Johnsonbaugh and S. Jost, *Pattern recognition and image analysis*, Prentice Hall, New Jersey, 1996.
76. H. Greenspan, Non-parameteric texture learning, in *Early Visual Learning*, S.K. Nayar and T. Poggio (Eds.), pp. 299-328, Oxford University Press, New York, 1996.

77. J.F. Haddon and J.F. Boyce, Image segmentation by unifying region and boundary information, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 929-948, 1990.
78. J.F. Haddon and J.F. Boyce, Co-occurrence matrices for image analysis, *Electronics and Communication Engineering Journal*, pp. 71-83, 1993.
79. J.F. Haddon and J.F. Boyce, Texture classification of segmented regions of FLIR images using neural networks, Proc. 1st International Conference on Image Processing, 1994.
80. J.F. Haddon, M. Schneebeli and O. Buser, Automatic segmentation and classification using a co-occurrence based approach, Proceedings of Digital Image Technologies II, Techniques and Civil Engineering Applications, 1997.
81. J.F. Haddon and J.F. Boyce, Integrating spatio-temporal information in image sequence analysis for the enforcement of consistency of interpretation, *Digital Signal Processing*, special issue on image analysis and information fusion, 1998.
82. J.F. Haddon, Adaptive scene analysis, Proc. Workshop on Advanced Concepts for Intelligent Vision Systems (ACIVS'99), Baden-Baden, pp. 68-73, 1999.
83. G.M. Haley and B.S. Manjunath, Rotation-invariant texture classification using a complete space-frequency model, *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 255-269, 1999.
84. M.W. Hansen and W.E. Higgins, Relaxation methods for supervised image segmentation, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 19, no. 9, pp. 949-961, 1997.
85. A.R. Hanson and E.M. Riseman, Visions: a computer system for interpreting scenes, in *Computer Vision Systems*, A.R. Hanson and E.M. Wiseman (eds.), pp. 303-333, Academic Press, New York, 1978.
86. R.M. Haralick, K. Shanmugam and I. Dinstein, Textural features for image classification, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 6, pp 610-621, 1973.
87. R.M. Haralick, Statistical and structural approaches to texture, *Proceedings of IEEE*, vol. 67, pp. 786-804, 1979.
88. R.M. Haralick, Image segmentation survey, in *Fundamentals in Computer Vision*, O. D. Faugeras (ed.), pp. 209-224, Cambridge University Press, Cambridge, 1983.
89. R.M. Haralick and L.G. Shapiro, Survey- image segmentation techniques, *Computer Vision Graphics and Image Processing*, vol. 29, pp. 100-132, 1985.
90. R.M. Haralick and L.G. Shapiro, *Computer and robot vision*, vol. 1, Addison Wesley, 1993.
91. R.M. Haralick, Propagating Covariance In Computer Vision," in *Advances in Image Understanding*, Azriel Rosenfeld, Bowyer and Ahuja, (eds.), IEEE Computer Society Press, pp. 142-157, Washington, 1996 and in *Studies in Pattern Recognition*, H. Freeman (ed.), pp. 171-182, World Scientific, Singapore, 1996.
92. D. Harwood, T. Ojala, M. Pietikäinen, S. Kelman and L.S. Davis, Texture classification by center-symmetric auto-correlation using Kullback discrimination of distributions, *Pattern Recognition Letters*, vol. 16, no. 11, pp. 1-10, 1995.
93. D.C. He and L. Wang, Texture features based on texture spectrum, *Pattern Recognition*, no. 5, pp. 391-399, 1991.
94. D.C. He and L. Wang, Unsupervised textural classification of images using the texture spectrum, *Pattern Recognition*, vol. 25, no. 3, pp. 247-255, 1992a.

95. D.C. He and L. Wang, Detecting texture edges from images, *Pattern Recognition*, vol. 25, no. 6, pp. 595-600, 1992b.
96. H. He and Y.Q. Chen, Unsupervised texture segmentation using resonance algorithm for natural scenes, *Pattern Recognition Letters*, vol. 21, pp. 741-757, 2000.
97. M. Hild and Y. Shirai, Interpretation of natural scenes using multi-parameter default models and qualitative constraints, Proc. 4th International Conference on Computer Vision, pp. 497-501, Berlin, Germany, 1993.
98. S. Hirata, Y. Shirai and M. Asada, Scene interpretation using 3D information extracted from monocular colour images, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1603-1610, Raleigh, NC, 1992.
99. T.H. Hong, C.R. Dyer and A. Rosenfeld, Texture primitive extraction using an edge based approach, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 10, no. 10, pp. 659-675, 1980.
100. S.A. Hojjatoleslami and J. Kittler, Region growing: a new approach, *CVSSP Technical Report TR-6/95*, University of Surrey, Department of Electronic and Electrical Engineering, 1995.
101. A. Hoover, G. Jean-Baptiste, X. Jiang, P.J. Flynn, H. Bunke, D.B. Goldof, K. Bowyer, D.W. Eggert, A. Fitzgibbon and R.B. Fisher, An experimental comparison of range image segmentation algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 673-689, 1996.
102. A. Hoover, G. Jean-Baptiste, D.B. Goldof and K. Bowyer, A methodology for evaluating range image segmentation techniques, Proc. 2nd IEEE Workshop on Applications for Computer Vision, 1994.
103. A.R.R. Hummel and S. Zucker, Scene labelling by relaxation operations, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, pp. 420-433, 1976.
104. A. Ide, M. Tateda, H. Naruse, A. Nobiki and T. Yabuta, Automatic recognition and stereo correspondence of target objects in natural scenes, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1597-1602, Raleigh, NC, 1992.
105. L. Itti, C. Koch and E. Niebur, A model of saliency-based visual attention for rapid analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254-1259, 1998.
106. L. Itti and C. Koch, Learning to detect salient objects in natural scenes using visual attention, Proc. Image Understanding Workshop, 1999.
107. B. Jähne and H. Haußecker, *Computer vision and applications: a guide for students and practitioners*, Academic Press, New York, 2000.
108. A.K. Jain and R.C. Dubes, Algorithms for clustering data, Prentice Hall, Englewood Cliffs, N.J., 1988.
109. A.K. Jain, R.P. Duin and J. Mao, Statistical pattern recognition: an overview, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4-37, 2000.
110. R. Jain, R. Kasturi and B.G. Schunck, *Machine vision*, McGraw-Hill International editions, Singapore, 1995.
111. M.E. Jernigan and F. D'astous, Entropy-based texture analysis in the spatial frequency domain, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 2, pp. 237-243, 1984.

112. B. Julesz, Textons, the elements of texture perception and their interactions, *Nature*, vol. 290, pp. 91-97, 1981.
113. M. Kallergi, G.M. Carney and J. Gaviria, Evaluating the performance of detection algorithms in digital mammography, *Medical Physics*, vol. 26, no.2, pp. 267-275, 1999.
114. T. Kanungo, M.Y. Jaisimha, J. Palmer and R.M. Haralick, A methodology for quantitative performance evaluation of detection algorithms, *IEEE Transactions on Image Processing*, vol. 4, no. 12, pp. 1667-1674, 1995.
115. L.M. Kaplan, Extended fractal analysis for texture classification and segmentation, *IEEE Transactions on Image Processing*, vol. 8, no. 11, pp. 1572-1585, 1999.
116. K. Karu, A. Jain and R.M. Bolle, Is there any texture in the Image? *Pattern Recognition*, vol. 29, pp. 1437-1446, 1996.
117. I.Y. Kim and H.S. Yang, Efficient image labelling based on markov random field and error backpropagation network, *Pattern Recognition*, vol. 26, no. 2, pp. 1695-1707, 1995.
118. L. Kitchen and A. Rosenfeld, Scene analysis using region-based constraint filtering, *Pattern Recognition*, vol. 17, no. 2, pp. 189-203, 1984.
119. V. Kovalev and M. Petrou, Multidimensional co-occurrence matrices for object recognition and matching, *Graphical Models and Image Processing*, vol. 58, no. 3, pp. 187-197, 1996.
120. P. Kruizinga and N. Petkov, Nonlinear operator for oriented texture, *IEEE Transactions on Image Processing*, vol. 8, no. 10, pp. 1395-1407, 1999.
121. K.S. Kumar and U.B. Desai, Joint segmentation and image interpretation, Proc. 3rd IEEE International Conference on Image Processing, vol. 1, pp. 853-856, 1996.
122. P. Kundu and B. B. Chaudhuri, Fuzzy geometric feature-based texture classification, *Pattern Recognition Letters*, vol. 14, pp. 825-832, 1993.
123. T. Kurita, An efficient agglomerative clustering algorithm using a heap, *Pattern Recognition*, vol. 24, no. 3, pp. 205-209, 1991.
124. M.D. Lavine, A knowledge based computer vision system, in *Computer Vision Systems*, A.R. Hanson and E.M. Riseman (eds.), pp. 335-352, Academic Press, New York, 1978.
125. M.D. Lavine and S.I. Shaheen, A modular computer vision system for picture segmentation and interpretation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 5, pp. 540-556, 1981.
126. M.D. Lavine and A.M. Nazif, An experimental rule based system for testing low level segmentation techniques, *Multicomputers and Image Processing Algorithms and Programs*, Academic Press, pp. 149-160, 1982.
127. K.I. Laws, Textured image segmentation, USCPI Rep. 940, Image Processing Institute, University of Southern California, Los Angeles, 1980a.
128. W.K. Leow and R. Miikkulainen, Visual schemas in neural networks for object recognition, *Connection Science*, vol. 9, pp.161-200, 1997.
129. B. Lerner, H. Guterman, M. Aladjem, I. Dinstein, A comparative study of neural network based feature extraction paradigms, *Pattern Recognition Letters*, vol. 20, pp. 7-14, 1999.
130. L. Li, J. Gong and W. Chen, Gray-level image thresholding based on Fisher linear projection of two-dimensional histogram, *Pattern Recognition*, vol. 30, no. 5, pp. 743-749, 1997.
131. X. Liu, T. Kanungo and R.M. Haralick, Statistical validation of computer vision software, Proc. DARPA Image Understanding Workshop, vol. II, pp. 1533-1540, Palm Springs, CA, 1996.

132. S.W. Lu and H. Xu, Textured image segmentation using autoregressive model and artificial neural network, *Pattern Recognition*, vol. 28, no. 12, pp. 1807-1817, 1995.
133. W. Mackeown, A labelled image database and its application to outdoor scene analysis, *PhD Thesis*, University of Bristol, UK, 1994.
134. W. Mackeown, P. Greenway, B.T. Thomas and W. Wright, Contextual image labelling with a neural network, *Proc. IEE Vision, Image and Speech Processing*, vol. 1, no. 2, pp. 151-156, 1994a.
135. W. Mackeown, P. Greenway, B.T. Thomas and W. Wright, Road recognition with a neural network, *Engineering Applications of Artificial Intelligence*, vol. 7, no. 2, pp. 169-176, 1994b.
136. V. Manian, R. Vásquez and P. Katiyar, Texture classification using logical operators, *IEEE Transactions on Image Processing*, vol. 9, no. 10, pp. 1693-1703, 2000.
137. J. Mantas, Methodologies in pattern recognition and image analysis - a brief survey, *Pattern Recognition*, vol. 20, no. 1, pp. 1-6, 1987.
138. Meastex database: <http://www.cssip.elec.uq.edu.au/~guy/meastex/meastex.html>
139. A. Mecocci, R. Lodola and U. Salvatore, Outdoor scene interpretation for blind people navigation, *Proc. 5th International Conference on Image Processing and its Applications*, pp. 256-260, Edinburgh, UK, 1995.
140. G.G. Medioni and Y. Yasumoto, A note on using fractal dimension for segmentation, *Proc. IEEE Computer Vision Workshop*, pp. 25-30, 1984.
141. A. Mehnert and P. Jackway, An improved seeded region growing algorithm, *Pattern Recognition Letters*, vol. 18, pp. 1065-1071, 1997.
142. R. Michalski, A. Rosenfeld, Z. Duric, M. Maloof and Q. Zhang, Learning patterns in images, in Michalski, R.S., Bratko, I. and Kubat, M. (Eds.), *Machine Learning and Data Mining: Methods and Applications*, pp. 241-268, John Wiley & Sons, London, 1998.
143. O.R. Mitchell, C.R. Myers and W. Boyne, A maximum measure for image texture analysis, *IEEE Transactions on Computers*, vol. 2, pp. 408-414, 1977.
144. J.W. Modestino, R.W. Fries and A.L. Vickers, Texture discrimination based upon an assumed stochastic texture model, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 5, pp. 557-580, 1981.
145. V. Murino, C. Ottonello, and S. Pagnan, Noisy texture classification: a higher order statistics approach, *Pattern Recognition*, vol. 34, no. 4, pp. 383-393, 1998.
146. M. Nadler and E.P. Smith, *Pattern recognition engineering*, John Wiley, New York, 1993.
147. A.M. Nazif and M.D. Lavine, Low level image segmentation: an expert system, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 5, pp. 555-577, 1984.
148. L. Ng, M. Nixon and J. Carter, Feature sets for texture classification, in *Proc. of International Conference on Advances in Pattern Recognition*, S. Singh (Ed.), Springer, London, pp. 35-44, 1998.
149. M.K. Ng, A note on constrained k-means algorithm, *Pattern Recognition*, vol. 33, pp. 515-519, 2000.
150. P.P. Ohanian and R.C. Dubes, Performance evaluation for four classes of textural features, *Pattern Recognition*, vol. 25, no. 8, pp. 819-833, 1992.
151. R.B. Ohlander, Analysis of natural scenes, *PhD Thesis*, Carnegie Institute of Technology, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1975

152. Y. Ohta, Region oriented image analysis system by computer, *PhD Thesis*, Kyoto University, March 1985.
153. Y. Ohta, T. Kanade and T. Sakai, Colour information for region segmentation, *Computer Graphics, Vision and Image Processing*, vol. 13, pp. 222-241, 1980.
154. J.R. Ohm and P. Ma, Feature-based cluster segmentation of image sequences, *Proc. IEEE International Conference on Image Processing*, pp. 178-181, 1997.
155. T. Ojala, M. Pietikäinen and D. Harwood, A comparative study of texture measures with classification based on feature distributions, *Pattern Recognition*, vol. 29, pp. 51-59, 1996.
156. T. Ojala and M. Pietikäinen, Unsupervised texture segmentation using feature distributions, *Pattern Recognition*, vol. 32, pp. 477-486, 1999.
157. N. Otsu, A threshold selection method from grey level histograms, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, pp. 62-66, 1978.
158. N.R. Pal and S.K. Pal, A review on image segmentation techniques, *Pattern Recognition*, vol. 26, pp. 1277-1294, 1993.
159. F. Paetzold and U. Franke, Road recognition in urban environment, *Image and Vision Computing*, vol. 18, pp. 377-387, 2000.
160. N. Papamarkos, C. Strouthopoulos and I. Andreadis, Multithresholding of colour and gray-level images through a neural network technique, *Image and Vision Computing*, vol. 18, pp. 213-222, 2000.
161. J.R. Parker, *Practical computer vision using C*, John Wiley, New York, 1994.
162. J.A. Parikh, A comparative study of cloud classification techniques, *Remote Sensing of Environment*, vol. 6, pp. 67-81, 1977.
163. P. Parodi and G. Piccioli, A feature based recognition scheme for traffic scenes, *Proc. Intelligent Vehicles Symposium*, pp. 229-234, Tokyo, Japan, 1995.
164. J. Pauwels and G. Frederix, Finding salient regions in images, *Computer Vision and Image Understanding*, vol. 75, pp. 73-85, 1999.
165. T. Pavlidis, *Algorithms for graphics and image processing*, Springer, Berlin, 1982.
166. A. Pentland, Fractal-based description of natural scenes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 661-674, 1984.
167. W.A. Perkins, Area segmentation of images using edge points, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 2, no. 1, pp. 8-15, 1980.
168. J. Peng and B. Bhanu, Closed-loop object recognition using reinforcement learning, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 139-154, 1998.
169. P. Perner, An architecture for a CBR image segmentation system, *Engineering Applications of Artificial Intelligence*, vol. 12, pp. 749-759, 1999.
170. M. Petrou, *Image processing- the fundamentals*, John Wiley, Chichester, 1999.
171. R.W. Picard, T. Kabir and F. Liu, Real-time recognition with the entire Brodatz texture databases, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, New York, pp. 638-639, 1993.
172. R.W. Picard and T. Kabir, Finding similar patterns in large image database, *Proc. IEEE ICASSP conference*, Minneapolis, vol. 5, pp. 161-164, 1993.
173. O. Pichler, A. Teuner and B.J. Hosticka, A comparison of texture feature extraction using adaptive Gabor filter, pyramidal and tree structured wavelet transforms, *Pattern Recognition*, vol. 29, no. 5, pp. 733-742, 1996.

174. M. Pietikäinen,, Image texture analysis and segmentation, *PhD Thesis*, University of Oulu, 1982.
175. M. Pietikäinen, T. Ojala and Z. Xu, Rotation-invariant texture classification using feature distributions, *Pattern Recognition*, vol. 33, pp. 43-52, 2000.
176. I. Pitas, *Digital image processing algorithms and applications*, John Wiley, New York, 2000.
177. J.M. Prager, Extracting and labeling boundary segments in natural scenes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 1, pp. 16-27, 1980.
178. W.K. Pratt, *Digital image processing*, John Wiley, New York, 1991.
179. T. Randen and J.H. Husøy, Filtering for texture classification: a comparative study, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 291-310, 1999.
180. T.R. Reed and J.M.H. Buf, A review of recent texture segmentation and feature extraction techniques, *Computer Vision Graphics and Image Processing: Image Understanding*, vol. 57, no. 3, pp. 359-372, 1993.
181. U. Regenberger and V. Graefe, Visual recognition of obstacles on roads, in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 980-987, Munich, Germany, 1994.
182. F. Roli, G. Giacinto and G. Vernazza, Comparison and combination of statistical and neural network algorithms for remote-sensing image classification, *Neurocomputation in Remote Sensing Data Analysis*, Advances in Spatial Science Series, Springer, 1997.
183. A. Rosenfeld, Scene labelling by relaxation operations, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 6, pp. 420-433, 1976.
184. A. Rosenfeld, Survey: Image analysis and computer vision: 1999, *Computer Vision and Image Understanding*, vol. 78, pp. 222-302, 2000.
185. F.A. Sadjadi, Performance evaluations of correlations of digital images using different separability measures, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, no. 4, pp. 436-441, 1982.
186. P.K. Sahoo, S. Soltani, A.K.C. Wong and Y.C. Chen, A survey of thresholding techniques, *Computer Vision, Graphics and Image Processing*, vol. 41, pp. 233-260, 1988.
187. S. Santini and R. Jain, Similarity measures, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 871-883, 1999.
188. S. Singh, J.F. Haddon and M. Markou. Nearest Neighbour Strategies for Image Understanding, Proc. Workshop on Advanced Concepts for Intelligent Vision Systems (ACIVS'99), Baden-Baden, pp. 74-79, August, 1999.
189. S. Singh, A single nearest neighbour fuzzy approach for pattern recognition, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 13, no. 1, pp. 49-54, 1999.
190. S. Singh, M. Markou and J.F. Haddon. FLIR Image Segmentation and Natural Object Classification, Proc. 15th International Conference on Pattern Recognition, Barcelona, IEEE Press, vol. 1, pp. 681-684, 2000a.
191. S. Singh, M. Markou and J.F. Haddon. Natural object classification using artificial neural networks, Proc. IEEE International Joint Conference on Neural Networks, Como, Italy, IEEE Press, vol. 3, pp. 139-144, 2000b.

192. S. Singh, M. Markou and J.F. Haddon. Detection of new image objects in video sequences using neural networks, SPIE conference on Applications of Artificial Neural Networks in Image Processing, Electronic Imaging '2000, San Jose, pp. 204-213, 2000c.
193. S. Singh and M. Markou. Learning and Adaptation in Scene Analysis Tasks using Neural Networks, Proc. 6th International Conference on Control, Automation, Robotics and Vision, Singapore, December, 2000.
194. S. Singh, R. Al-Mansoori, Identification of regions of interest in digital mammograms, *Journal of Intelligent Systems*, vol. 10, no. 2, pp. 183-217, 2000.
195. S. Singh, M. Markou and J.F. Haddon, Nearest Neighbour Classifiers in Natural Scene Analysis, (in press, *Pattern Recognition*, 2001).
196. S. Singh and K.J. Bovis, Medical image segmentation in digital mammography, in *Advanced Algorithmic Approaches to Medical Image Segmentation*, J. Suri, K. Setarehdan and S. Singh (eds.), Springer, London, 2001.
197. S. Singh and M. Sharma, Texture experiments with Meastex and Vistex benchmarks, Proc. International Conference on Advances in Pattern Recognition, Lecture Notes in Computer Science no. 2013, S. Singh, N. Murshed and W. Kropatsch (eds.), Springer, 2001.
198. G. Smith and I. Burns, Measuring texture classification algorithms, *Pattern Recognition Letters*, vol. 18, pp. 1495-1501, 1997.
199. M. Sonka, V. Hlavac and R. Boyle, *Image processing, analysis and machine vision*, PWS press, 1998.
200. M. Spann and R. Wilson, A quad-tree approach to image segmentation which combines statistical and spatial information, *Pattern Recognition*, vol. 18, pp. 257-269, 1983.
201. J. Strand and T. Taxt, Local frequency features for texture classification, *Pattern Recognition*, vol. 27, no. 10, pp. 1397-1406, 1994.
202. T.M. Strat, *Natural object recognition*, Springer, Berlin, 1992.
203. T.F. Syeda-Mahmood, Detecting perceptually salient texture regions in images, *Computer Vision and Image Understanding*, vol. 76, no. 1, pp. 93-108, 1999.
204. H. Tamura, S. Mori and T. Yamawaki, Textural features corresponding to visual perception, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 6, pp. 460-473, 1978.
205. Y. Taniguchi, Y. Shirai and M. Asada, Scene interpretation by fusing intermediate results of multiple visual sensory information processing, Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligence, pp. 699-706, Las Vegas, 1994.
206. S. Theodoridis and K. Koutroubas, *Pattern recognition*, Academic press, San Diego, 1999.
207. F. Tomita, Y. Shirai and S. Tsuji, Description of textures by a structural analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, no. 2, pp. 183-191, 1982.
208. G.T. Toussaint, The use of context in pattern recognition, *Pattern Recognition*, vol. 10, pp. 189-204, 1978.
209. M. Tuceyran and A.K. Jain, Texture analysis, in *The Handbook of Pattern Recognition and Computer Vision* (2nd edition), C.H. Chen, L.F. Pau and P.S.P. Wang (eds.), pp. 207-248, World Scientific, 1998.
210. A. Tversky, Feature of similarity, *Psychological Review*, vol. 84, no. 4, pp. 327-350, 1977.
211. M. Unser, Local linear transforms for texture measurements, *Signal Processing*, vol. 2, pp. 61-79, 1986.

212. A. Vailaya, A. Jain, and H.J. Zhang, On image classification: city images vs. landscapes, *Pattern Recognition*, vol. 31, no. 12, pp. 1921-1935, 1998.
213. L. van Gool, P. Dewaele and A. Oosterlinck, Texture analysis: Anno 1983, *Computer Vision, Graphics and Image Processing*, vol. 29, pp. 336-357, 1985.
214. Vistex database, <http://vismod.www.media.mit.edu/vismod/imagery/VisionTexture>
215. R.F. Walker, P. Jackway and I.D. Longstaff, Improving co-occurrence matrix feature discrimination, Proc. of DICTA'95, 3rd International Conference on Digital Image Computing: Techniques and Applications, pp. 643-648, 1995.
216. L. Wang and D.C. He, Texture classification using texture spectrum, *Pattern Recognition*, vol. 23, pp. 905-910, 1990.
217. Z. Wang, G. Sylos, M. Labini, M.D. Sario and R. Magnuolo, Unsupervised texture image segmentation by improved neural network ART2, Proc. CRIFFSS'94 AIAA/NASA Conference on Intelligent Robots for Factory Field, Service, and Space, Lyndon, Houston, TX, 1994.
218. Z. Wang, A. Guerriero and M.D. Sario, Comparison of several approaches for the segmentation of texture images, *Pattern Recognition Letters*, vol. 17, pp. 509-521, 1996.
219. A. Webb, Statistical pattern recognition, Arnold, London, 1999.
220. H. Weschler and M. Kidode, A random walk procedure for texture discrimination, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 3, pp. 272-280, 1979.
221. H. Weschler and T. Citron, Feature extraction for texture classification, *Pattern Recognition*, vol. 12, pp. 301-311, 1980.
222. J.S. Weszka and A. Rosenfeld, An application of texture analysis to materials inspection, *Pattern Recognition*, vol. 8, pp. 195-199, 1976.
223. J.S. Weszka, C. R. Dyer and A. Rosenfeld, A comparative study of texture measures for terrain classification, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 4, pp. 269-285, 1976.
224. J.S. Weszka and A. Rosenfeld, Threshold evaluation techniques, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, pp. 622-629, 1978.
225. Y. Xu, V. Olman and E.C. Uberbacher, A segmentation algorithm for noisy images: design and evaluation, *Pattern Recognition Letters*, vol. 19, pp. 1213-1224, 1998.
226. L. Yang, F. Albrechtsen, Tor Lønnestad and P. Grøttum, A supervised approach to the evaluation of image segmentation methods, in *Computer Analysis of Images and Patterns*, V. Hlavac and R. Sara (eds.), pp. 759-765, Springer, Berlin, 1995.
227. F. Yarman-Vural and E. Ataman, Noise, histogram and cluster validity for Gaussian-mixed data, *Pattern Recognition*, vol. 20, no. 4, pp. 385-401, 1987.
228. W.A. Yasnoff, J.K. Mui and J.W. Bacus, Error measures for scene segmentation, *Pattern Recognition*, vol. 9, pp. 217-231, 1977.
229. M. Yeung, B.L. Yeo and B. Liu, Segmentation of video by clustering and graph analysis, *Computer Vision and Image Processing*, vol. 71, no. 1, pp. 94-109, 1998.
230. M. Yoshimura and S. Oe, Evolutionary segmentation of texture image using genetic algorithms towards automatic decision of optimum number of segmentation areas, *Pattern Recognition*, vol. 32, pp. 2041-2054, 1999.
231. N. Zahid, M. Limouri and A. Essaid, A new cluster validity for fuzzy clustering, *Pattern Recognition*, vol. 32, pp. 1089-1097, 1999.

232. Y.J. Zhang, Evaluation and comparison of different segmentation algorithm, *Pattern Recognition Letters*, vol. 18, pp. 963-974, 1997.
233. Y.J. Zhang and J.J. Gerbrands, Segmentation evaluation using ultimate measurement accuracy, Proc. SPIE vol. 1657, Image Processing Algorithms and Techniques III, pp. 449-460, 1992.
234. M. Zurada, *Introduction to artificial neural networks*, Wadsworth, 1992.