

Neural Networks for Meteorological Satellite Image Interpretation

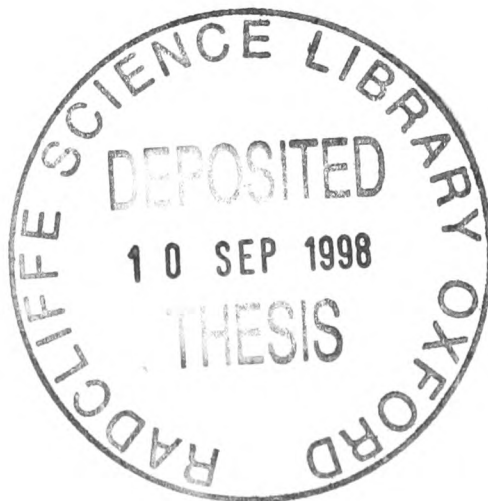
by

Michael Robert Brewer



Department of Engineering Science
University of Oxford

Michaelmas Term, 1997



This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

M. R. Brewer, Merton College

Copyright © 1997 Michael Robert Brewer
All Rights Reserved

Michael Robert Brewer
Merton College

Doctor of Philosophy
Michaelmas Term, 1997

Neural Networks for Meteorological Satellite Image Interpretation

Abstract

Meteorological satellite images at visible and infra-red wavelengths are an invaluable source of information on cloud systems because of their extensive coverage of the whole of the Earth's surface, providing data in areas that are only sparsely monitored, if at all, by other means. Although this information has been used subjectively by forecasters for many years, the lack of automatic, quantitative analysis techniques largely prevents its assimilation into numerical weather prediction (NWP) models, which are the basis of all modern weather forecasting. This thesis investigates the use of neural network techniques for the analysis of the images in order to make fuller use of the available data.

The recognition of a particular type of cloud is dependent on the determination of a set of features from the satellite image spectral bands that will give discriminating information. This feature extraction and selection process is dealt with in detail, and a feature selection process based on the radial basis function (RBF) neural network is presented. The high-dimensional feature space is visualized on a two-dimensional plane by the use of three techniques: the Kohonen map, the Sammon map, and a recently-developed technique known as the Generative Topographic Mapping (GTM). Classification results using a multi-layer perceptron (MLP) and an RBF neural network are presented.

The results of independently classifying each pixel in an image are compared with a method that makes use of contextual information, the Markov Random Field (MRF) model. The limitations of the pixel-based approach are highlighted, and a region-based approach is presented that enables the definition of large-scale regions of uniform cloud type. Two segmentation methods are used, the active contour (or snake) model, and the more recently-developed level set technique. The latter method was found to provide many benefits over the former. The region-based approach will facilitate the assimilation of cloud system information into NWP models in the future.

Acknowledgements

Thanks must go to:

- Catherine, for providing the motivation and direction needed to complete this work, and for lots of bunnies. This is dedicated to you.

Stepping out to Angelucci's for my coffee beans.

- My parents and family, for their support, financial and otherwise, to get me here.

Oh Father please help me.

- Lionel Tarassenko, my supervisor at Oxford University, for his guidance and advice throughout my research.

Oh and the iron will and the iron hand.

- Bryan Conway, my supervisor at the UK Meteorological Office, for helping to make such an interesting research project possible, and for invaluable input in many areas.

And my radio says tonight it's gonna freeze.

- George Pankiewicz, also of the UK Meteorological Office, for providing a wealth of useful advice and interesting ideas.

That's a genuine weathervane – it moves with the breeze.

- Ravi Malladi, of the Lawrence Berkeley Laboratory, California, for his calm help with the level set work.

Maybe he could talk about the tricks of the trade.

- Richard Bankert, of the Naval Research Laboratory, Monterey, California, for providing the labelled satellite images used in this thesis.

The mariner of dry dock land.

- Everyone in Lionel's research group, for interesting company and enlightening discourse ... plus the odd bit of code here and there.

You step inside but you don't see too many faces.

- The Engineering and Physical Sciences Research Council and the UK Meteorological Office for their financial support.

Now there's nothing left for luxury, nothing left to pay my heating bill.

- Chaka, my dog.

I'm a soldier of fortune, I'm a dog of war.

A long time ago came a man on a track ...

Contents

1	Introduction	1
1.1	The uses of satellite data	2
1.2	Cloud-type identification	4
1.2.1	The visible channel	4
1.2.2	The infra-red channel	6
1.2.3	The use of both channels	8
1.3	Schemes for automated satellite image interpretation	14
1.4	Datasets used	17
1.4.1	The Bankert database	18
1.4.2	The Meteosat database	20
1.5	Software used	21
1.6	Outline of the thesis	22
2	Bayesian classification methods and neural networks	25
2.1	Bayes classifiers	25
2.1.1	Estimating the class-conditional densities	26
2.1.1.1	Parametric approach	27
2.1.1.2	Non-parametric approach	29
2.2	The MLP	30
2.2.1	Structure of the MLP	30
2.2.2	Training the MLP – the error back-propagation algorithm	32
2.2.2.1	The output layer	34
2.2.2.2	The hidden layer	35
2.2.3	Output representation and decision rule	36
2.3	The RBF network	37
2.3.1	Structure of the RBF network	37
2.3.2	Training the RBF network	38
2.3.2.1	The hidden layer	39
2.3.2.2	The output layer	40
2.3.3	Further properties and comparison with MLPs	41
3	Features	43
3.1	Description and overview of features	44
3.2	Feature selection methods	49
3.2.1	Bhattacharyya distance measure	50
3.2.2	Radial basis function method	51
3.3	Feature selection results	53
3.3.1	Bhattacharyya distance measure results	53

3.3.2	Radial basis function method results	57
3.4	Conclusions	62
4	Visualization techniques	64
4.1	The Kohonen feature map	65
4.1.1	Implementation details	66
4.2	The Sammon mapping	67
4.2.1	Implementation details	68
4.2.1.1	Batch K -means algorithm	68
4.2.2	Initialization of the Sammon vectors	69
4.3	The Generative Topographic Mapping	70
4.4	Kohonen and Sammon results	75
4.4.1	Meteosat database	75
4.4.2	Bankert database	78
4.5	GTM results	80
4.5.1	Meteosat database	80
4.5.2	Bankert database	82
4.6	Conclusions	87
5	Classification tests	89
5.1	Overview of previous work	90
5.1.1	General classification work	90
5.1.2	Neural network research	93
5.1.3	Overview	96
5.2	Linear discriminant tests	96
5.3	MLP classification tests	97
5.4	RBF classification tests	100
5.5	Discussion	103
5.5.1	Bankert database	103
5.5.2	Meteosat database	106
5.6	Conclusions	106
6	Pixel-by-pixel image analysis	108
6.1	Pixel-by-pixel analysis with a trained MLP	109
6.1.1	MLP results and discussion	109
6.2	Comparison with an MRF model	115
6.2.1	Basic MRF theory and application	116
6.2.2	Results from the multiscale MRF model	119
6.3	Unsupervised classification using the Kohonen map	123
6.4	Conclusions	129
7	Large-scale segmentation	131
7.1	Overview and previous work	132
7.2	Method of posterior images	134
7.3	The active contour model	136
7.3.1	Gaussian region method	137
7.3.2	Fitting using the EM algorithm	139
7.3.3	A Gaussian region training run	141
7.4	The level set method	143

7.4.1	Level set theory	144
7.4.2	Application to cloud segmentation	147
7.5	Results	149
7.5.1	Snake method results	150
7.5.2	Level set method results	155
7.6	Conclusions	163
8	Conclusions and contributions	166
8.1	Future work	169
A	The common weather satellites	171
A.1	Meteosat satellites	171
A.2	NOAA satellites	173
A.3	The Landsat satellite	174
B	The Multiscale Random Field model	175
B.1	Multiscale segmentation approach	175
B.1.1	Multiscale Random Field model	176
B.1.2	Sequential MAP estimation	177
B.2	Segmentation algorithm	178
B.2.1	Quadtree model	180
B.2.2	Pyramidal graph model	181
B.2.3	Parameter estimation	183
C	Additional mathematics	188
C.1	Gaussian mixture model	188
C.2	The Expectation Maximization algorithm	190
C.3	Application of EM to the Gaussian mixture model	192
C.4	Numerical methods for the active contour model	196
C.5	Additional mathematics for the level set method	198
C.5.1	Outline of the algorithm	198
C.5.2	Fast numerical techniques	201
C.5.2.1	The narrow band extension	201
C.5.2.2	The fast marching method	202
C.5.2.3	Narrow band and fast marching combined	203
D	Full-size images	205

List of Figures

1.1	An example AVHRR image pair, annotated	7
1.2	An example Meteosat image pair, annotated	10
1.3	Two example AVHRR image pairs	11
1.4	Two more example AVHRR image pairs	12
1.5	A further two example AVHRR image pairs	13
1.6	A screen-shot from the <i>NetSat</i> system	23
2.1	Quadratic decision surface	28
2.2	The structure of a multi-layer perceptron (MLP)	30
2.3	The sigmoid function	32
2.4	MLP decision surfaces	37
2.5	The structure of a radial basis function (RBF) network	38
2.6	RBF representation	42
3.1	The GLDV homogeneity feature over an entire image	47
3.2	Algorithm for the RBF feature selection method	52
3.3	Distribution of the infra-red spectral mean feature for Cs and Cu	55
3.4	Distribution of the infra-red long run emphasis feature for Cc and Sc	55
3.5	Distribution of the infra-red spectral mean feature for Ns and Cu	55
3.6	RBF accuracy vs inputs removed (Meteosat database)	59
3.7	RBF accuracy vs inputs removed (Bankert database)	60
4.1	The Kohonen architecture	65
4.2	The Kohonen map training algorithm	66
4.3	The batch <i>K</i> -means algorithm	69
4.4	An illustration of the GTM	72
4.5	Two example GTM posterior distributions	73
4.6	Kohonen and Sammon plots for the Meteosat database	77
4.7	Kohonen and Sammon plots for the Bankert database	79
4.8	GTM output for a 12×12 latent space grid for the Meteosat database	81
4.9	GTM output for a 15×15 latent space grid for the Bankert database	83
4.10	Centres of gravity detail for the Bankert GTM	84
4.11	Mean posterior detail for the Bankert GTM	85
5.1	Stopping scheme for the MLP training procedure	98
6.1	First pixel-by-pixel classification by the trained MLP of Chapter 5	112
6.2	Second pixel-by-pixel classification by the trained MLP of Chapter 5	113
6.3	Third pixel-by-pixel classification by the trained MLP of Chapter 5	114
6.4	First classification by the multiscale MRF model	120

6.5	Second classification by the multiscale MRF model	121
6.6	Third classification by the multiscale MRF model	122
6.7	First unsupervised colouring from run one	125
6.8	Second unsupervised colouring from run one	126
6.9	First unsupervised colouring from run two	127
6.10	Second unsupervised colouring from run two	128
7.1	The process from original image to posterior image	135
7.2	External image forces on the snake	137
7.3	Example training run for the Gaussian region method	142
7.4	An illustration of curvature flow	144
7.5	An illustration of the method of level sets	145
7.6	Final positions of the trained Gaussians	151
7.7	Snake segmentation of Deep Convective areas	152
7.8	Snake segmentation of Cumulus and Cirrocumulus areas	154
7.9	Snake segmentation of Dynamic cloud areas	155
7.10	Evolution of the zero level set on the Cirrocumulus posterior image	157
7.11	Cirrocumulus level set segmentation	158
7.12	Cumulus and Nimbostratus level set segmentation	159
7.13	Shallow and Deep Convection level set segmentation	160
7.14	Dynamic and Deep Convection level set segmentation	162
7.15	The effect of ϵ on the level set segmentation	163
B.1	The structure of a doubly-stochastic random field	176
B.2	Pyramid structure of the MSRF	176
B.3	The quadtree structure, and its one-dimensional analogue	180
B.4	The augmented pyramidal graph structure	182
B.5	The one-dimensional analogue to the hybrid graph structure	183
B.6	A summary of the complete SMAP segmentation algorithm	187
C.1	The basic level set algorithm	201
C.2	The narrow band extension	202
C.3	The two-dimensional grid for the level set fast marching method	204
C.4	The fast marching algorithm	204
D.1	The visible Meteosat image from 6 January 1994	206
D.2	The infra-red Meteosat image from 6 January 1994	207
D.3	The visible Meteosat image from 3 April 1994	208
D.4	The infra-red Meteosat image from 3 April 1994	209
D.5	The visible Meteosat image from 13 September 1994	210
D.6	The infra-red Meteosat image from 13 September 1994	211
D.7	The visible Meteosat image from 25 October 1994	212
D.8	The infra-red Meteosat image from 25 October 1994	213
D.9	The visible Bankert (AVHRR) image from 13 December 1988	214
D.10	The infra-red Bankert (AVHRR) image from 13 December 1988	215

List of Tables

1.1	Typical albedo values for various cloud and land types	5
1.2	The cloud classes from the Bankert dataset	19
1.3	The cloud classes from the Meteosat dataset	21
3.1	Grey-level difference vector (GLDV) features	45
3.2	Sum and difference histogram (SADH) features	46
3.3	Run-length (RL) features	46
3.4	Bhattacharyya features for the Meteosat database	56
3.5	Bhattacharyya features for the Bankert database	56
3.6	RBF features for the Meteosat database	61
3.7	RBF features for the Bankert database	61
5.1	Confusion matrix for Linear Discriminants for the Meteosat database	101
5.2	Confusion matrix for 15:15:5 MLP for the Meteosat database	101
5.3	Confusion matrix for RBF tests for the Meteosat database	101
5.4	Confusion matrix for Linear Discriminants for the Bankert database	102
5.5	Confusion matrix for 15:15:10 MLP for the Bankert database	102
5.6	Confusion matrix for RBF tests for the Bankert database	102
5.7	Comparison of the average accuracy achieved for the three methods	103
7.1	The final values of $P(C_i j)$ for the two trial Gaussians in Figure 7.3	143
A.1	Meteosat channel characteristics	172
A.2	AVHRR/2 channel characteristics	173

Chapter 1

Introduction

The first operational meteorological satellite was launched on 1 April 1960 [44], providing images of the Earth via a simple television camera. Since then, more advanced radiometers have been developed, providing higher resolution and more frequent images [84], resulting in a deluge of information being transmitted back to Earth every day. The satellite sensors provide radiance data in a number of spectral channels, but this alone is not what most users require. In order to make the best use of this wealth of radiance data, processing is required to create derivative products [96]. The automated extraction of information on cloud types and cloud systems from satellite imagery is seen as an important task [73, 82]. The study of cloud patterns is crucial for the analysis of weather systems in the short-term [4], and clouds are understood to be influential in determining the climate of the planet in the long-term [16, 91, 99].

The work presented in this thesis concerns the broad application of neural network techniques [8] to the extraction of cloud-type information from satellite imagery. The major aim is the accurate identification of different types of cloud within the image, and the subsequent large-scale segmentation of the image into distinct regions of cloud type. Each stage in the process, from the raw image through extraction and analysis of information from that image to the segmentation of large-scale regions within it, is covered in detail within a neural network framework.

This chapter provides an introduction to the issues related to the work presented here, beginning in Section 1.1 with a brief survey of some of the current uses of satellite images. A guide to the basic cloud types and their identification is presented in Section 1.2. A more detailed overview of the aims of this thesis, and how they fit in with the overall goals of the UK Meteorological Office today, is set out in Section 1.3. The image databases and software used for this study are described in Sections 1.4 and 1.5, followed by an outline of the thesis in Section 1.6.

1.1 The uses of satellite data

Until the first pictures of the Earth from space were produced, our ideas of the form and shape of large-scale cloud features had remained mostly speculative [44]. The first working weather satellite, TIROS-1¹, provided imagery using a simple television camera, and was designed specifically to obtain cloud pictures. Since then, satellites have carried a wider range of sensors [84] designed to provide not only imagery, but also quantitative measurements of various atmospheric and surface parameters. This information has been used extensively for a variety of research and operational applications.

Satellite data are an invaluable source of information because of their extensive coverage of the whole of the Earth's surface, providing data in areas that are only sparsely monitored, if at all, by weather ships, buoys, aircraft and observing stations [39]. Visible and infra-red images of the clouds covering the Earth are important for a number of reasons. Fundamentally, clouds at different levels are tracers of atmospheric energy and moisture flows. The study of satellite imagery, and in particular a time-series thereof, provides insight into the dynamical and physical processes that occur in the atmosphere and is therefore useful for climate modelling. The images have enhanced our understanding of synoptic-scale² cloud features and systems.

Some current satellite sensors are designed to measure remotely the properties of the

¹Television and Infra-Red Observation Satellite.

²In meteorological terms, *synoptic* refers to space and time scales of the order of 500–2000 km and 100 hours respectively.

upper atmosphere that are difficult to measure by conventional means; for example profiles of ozone, carbon dioxide and other chemical concentrations [66, 96]. Wind velocities at various heights can be measured using the Doppler shift; alternatively, over the ocean, the scattering of microwaves off capillary wavelets on the ocean surface allows surface wind velocities to be calculated. A method using radiance data is also currently in use whereby cloud structures are identified from a time-sequence of images, and correlation calculations between images are then used to derive the wind velocity [66]. The calculation of sea surface temperatures (SSTs) and the monitoring of land-cover and biomass are other examples of the use of satellite data.

Satellite images are also vital sources of information for operational analysts and forecasters, for several reasons [4]. Firstly, they are used as an analysis tool, especially in data-sparse areas, for example to locate fronts, or to spot smaller convective systems which may not be evident from other data even in relatively well-monitored areas. Secondly, images are a direct aid to short-period forecasting of, for instance, cloud and precipitation or related phenomena such as flash floods and lightning. A time-series of images is especially useful for this purpose. Thirdly, the information gleaned from the images provides input into numerical weather prediction (NWP) models, helping to define the initial conditions and moisture fields of the model, especially in data-sparse regions [39]. Lastly, the images provide a means of monitoring the NWP model performance in the early stages of a forecast.

Clouds play a vital role in determining the so-called *Earth radiation budget* (ERB) [16]. For climate stability, it is required that incoming solar energy is balanced by outgoing reflected and emitted energy. Clouds exert a significant influence on the energy streams by two competing processes: the albedo effect and the greenhouse enhancement, which reduce and increase respectively the net incoming energy flux. It is a major goal of NASA's Earth Radiation Budget Experiment (ERBE) [106] to determine the components that make up the overall planetary albedo, which includes the important effect of clouds. Of particular interest is the effect of cirrus clouds. This thin cloud is often difficult to detect, especially over rela-

tively bright land and snow, where radiation from the underlying surface renders the cirrus almost invisible [4, 66]. Determining the amount of cirrus is considered important due to its influence on the planetary albedo. The detection of cirrus is also important for the accurate determination of SSTs, for corrections have to be made to the calculated SST due to the thermal contamination caused by a thin layer of cirrus [16].

Inextricably linked to ERB considerations is the need for a globally-consistent cloud climatology which accurately estimates cloud amount (total, and by cloud type). This is part of the goal of the International Satellite Cloud Climatology Project (ISCCP) [92, 93, 99], set up in 1983, which will enable a better understanding of the cloud-climate sensitivity relationship.

Harris and Barrett [45] give eight uses of a cloud-type map (nephanalysis) of a satellite image scene, including climatological research, aviation forecasting, and radiation budget considerations. An accurate nephanalysis would also be useful in the analysis stage of operational forecasting mentioned above, and could be a useful step in the eventual identification of synoptic cloud features such as fronts and cyclones [82].

1.2 Cloud-type identification

The work in this thesis is primarily concerned with the automated recognition of cloud types within satellite imagery (see Section 1.3). This section will introduce the cloud types that will be encountered in this thesis, and will present a simple guide for their identification using visible and infra-red satellite imagery, highlighting some of the issues involved in cloud-type identification, and introducing some of the meteorological terms used throughout the thesis. Details of the satellites and sensors can be found in Section 1.4 and Appendix A.

1.2.1 The visible channel

Although also affected by the relative positions of the sun and satellite, the brightness observed in the visible channel is strongly dependent on the albedo of the underlying surface. The albedo is defined as the reflectance of a body or surface integrated across all solar wave-

Table 1.1: Typical albedo values for various cloud and land types. Values are only an indication since they can vary markedly, especially for clouds. Source: Bader *et al* [4].

Earth's surface	(%)	Clouds	(%)
Oceans, lakes	8	Shallow, broken clouds:	
		cumulus	30
Land surfaces:		cirrus, cirrostratus, cirrocumulus	35
dark soils	14	stratus	40
vegetation	18		
sand, desert	27	Thick clouds:	
		cirrostratus	74
Ice and snow:		stratus	65
sea ice	35	altostratus, stratocumulus	68
old snow	59	cumulus	75
fresh snow	89	nimbostratus	85
		cumulonimbus	90

lengths, and is expressed as a fraction or percentage of the incident radiation [16]. The albedo of a cloud can be a useful distinguishing factor, and is determined by the thickness, composition and particle size distribution [4, 66]:

1. Thick cloud has a higher albedo than thin cloud.
2. Water cloud has a higher albedo than ice cloud.
3. Cloud composed of small droplets has a higher albedo than cloud composed of large droplets.

In addition, penetration of light from the ground can influence the apparent albedo of thin cloud. These factors compete against each other to determine the albedo of a cloud. For instance the top of a cumulonimbus cloud is composed of ice, but the cloud has a great vertical extent; the latter is the dominant factor in this case, and the cloud therefore has a high albedo. Although stratus is comparatively thin, it can have a high albedo due to its composition of very small water droplets. Table 1.1 presents a summary of typical albedo values for various cloud and land types.

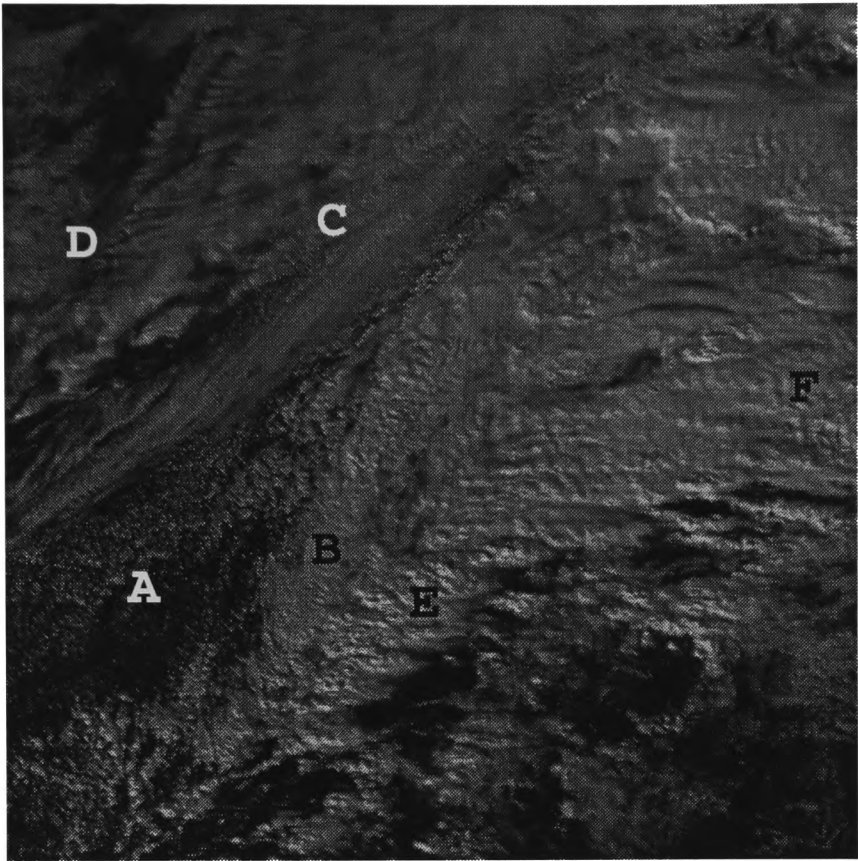
The texture of cloud patterns in the visible image provides important discriminatory information, and is caused both by variation in the albedo of the cloud and by shadows and shading. The standard cloud naming system used by meteorologists tends to label

clouds according to their physical appearance. Cumuliform cloud (such as cumulus and cirrocumulus) has a characteristic lumpy or cellular appearance, while stratiform clouds (such as stratus and cirrostratus) appear smoother and more uniform. Clouds exhibiting a fibrous or streaky appearance are likely to be cirriform. An example of a visible image is shown in Figure 1.1(a). The mottled texture characteristic of cumuliform cloud can be seen around A, with the smoother cirriform cloud around C and D.

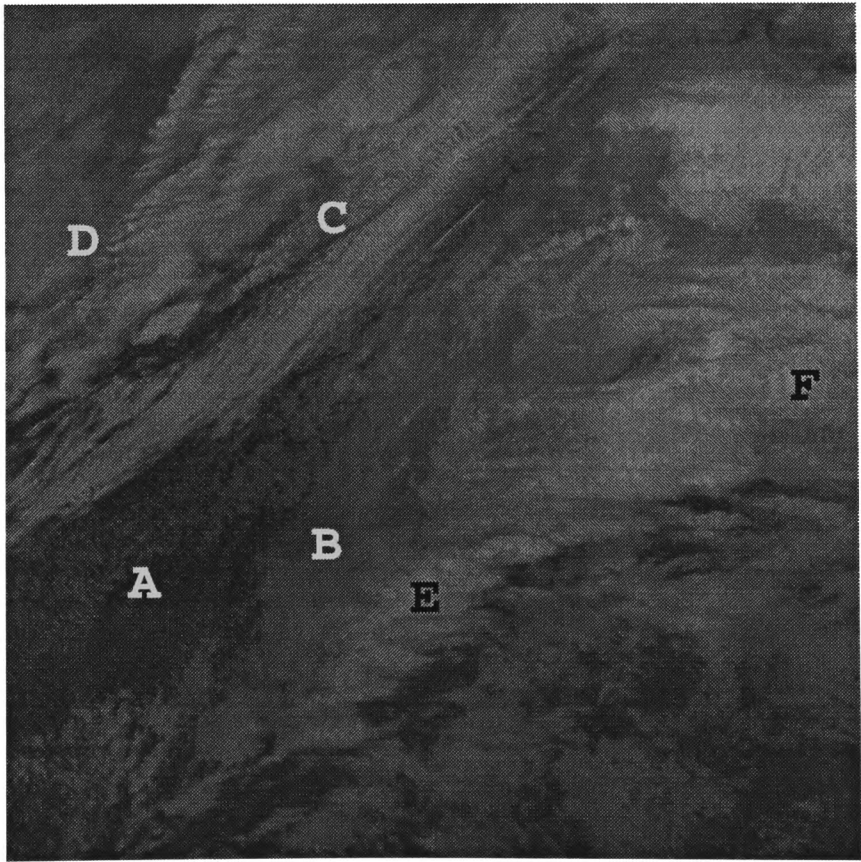
1.2.2 The infra-red channel

Unlike the visible channel, the infra-red channel primarily measures emitted rather than scattered radiation. The infra-red instrument counts are converted to brightness temperature using the simplifying assumption that they act as black-bodies [66]. Cloud tends to be colder than land, so if the brightness temperature were scaled directly to the image grey-levels, the clouds would appear dark. To make the images more intuitively consistent with the visible image, the scaling is conventionally reversed so that the relatively cold clouds appear white. As was the case for the visible channel, there is often contamination from radiation penetrating from the ground below, and in addition there is absorption from the atmosphere which varies with wavelength.

Since temperature generally decreases with height, the infra-red channel is essentially a measure of the height of the *top* of the cloud, and is important for distinguishing certain cloud types. In the standard naming system for clouds, a name starting with *cirro* implies high-level cloud, *alto* implies medium-level cloud, while no prefix implies low-level cloud. In the infra-red channel image shown in Figure 1.1(b), low-level cloud such as cumulus and stratocumulus, at A and B respectively, can be easily distinguished from high-level cloud such as cirrus (east of D), cirrostratus (north of C) and cirrocumulus (south-east of C). However, for thin or broken cloud, height assignment is often not straightforward because of the contamination problem mentioned above.



(a) Visible channel



(b) Infra-red channel

Figure 1.1: An example AVHRR image pair from 13 December 1988 off the coast of California. The image is 512 pixels wide, which is approximately 560 km. See text for a fuller description of the clouds contained within the image pair, Appendix A for details of the AVHRR sensor, and Figures D.9 and D.10 (on Pages 214 and 215) for larger versions.

1.2.3 The use of both channels

Identification of cloud type is best made using a combination of the spectral and textural information from more than one channel, often making use of context and a knowledge of the large-scale structure of the cloud systems within the image. A summary of the clouds present in the image pair of Figure 1.1 is as follows:

- Cumulus: A.
- Stratocumulus: B.
- Cirrocumulus: south-east of C.
- Cirrostratus: north of C.
- Cirrus: east of D.
- Altostratus: west of D.
- Cumulonimbus: south-west and north-east of E.
- Nimbostratus: north and west of F.

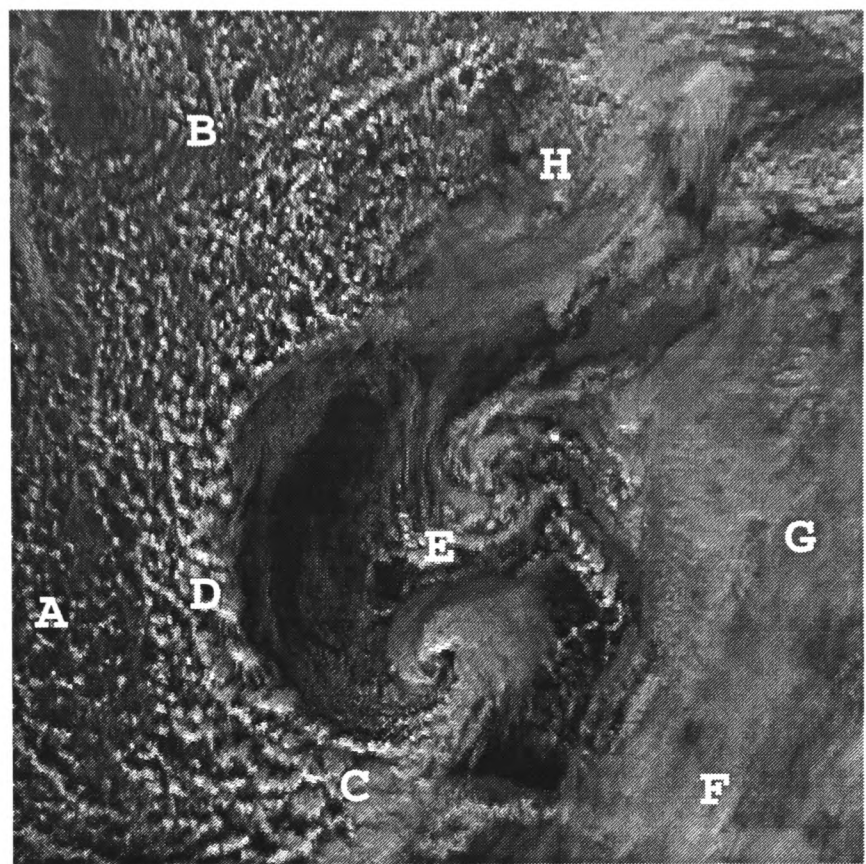
As mentioned before, these cloud names attempt to label the cloud according to its physical appearance. An alternative is to attach a label to describe the nature of the airmass associated with the cloud, which can be broadly divided into two types: convective and dynamic. Convective cloud is formed when air is heated by the warm ground, causing pockets of air to rise. The moisture condenses out as the rising air cools and saturates, giving rise to a pattern of small convective cells. Convective cloud can be further classified as *shallow* or *deep*. Cumulus clouds produced by the early stages of convective activity, when the tops are still at a low level, is termed *shallow* convection. If the air continues to rise, clouds of large vertical extent (such as cumulonimbus) are formed which may give rise to heavy precipitation. This is termed *deep* convection. Dynamic cloud is associated with larger-scale organized movements of airmasses, resulting in lifting and condensation (such as that which occurs at a front). Vertical motion is usually weaker than for convection, and the resulting

cloud masses are stratiform in nature (though with convective elements often embedded if the dynamic lifting releases instability).

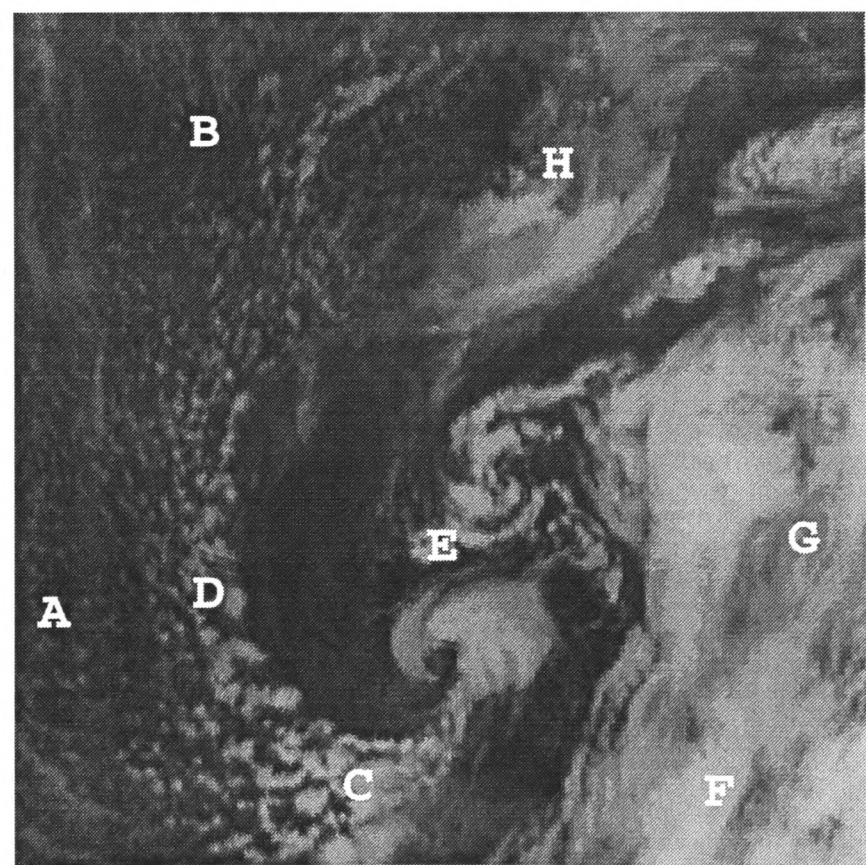
The Meteosat image pair in Figure 1.2 shows a large swathe of shallow convective cloud covering the western portion of the image from A around to B. A brighter band from D around to C and beyond is apparent in the infra-red image, indicative of deep convective activity. Deep convection is also present just to the north-east of E. It should be noted that cumulonimbus often appears brighter than this in the visible channel. This is due to the low solar elevation (the image is from 6 January 1994 over western Europe at 1200 GMT), so that the cloud is brightest where the sun reflects off the southern edges of the cloud. The large swathe of dynamic cloud to the east is associated with a front running from F to G, and then eastward and back around to H. There are three low pressure centres in the image, one just to the north-east of E, one south of E, and another at H.

Further examples of various cloud types are shown in Figures 1.3 to 1.5. Figure 1.3(a) shows a wedge of uniform cloud, identifiable as cirrostratus since it is very bright in the infra-red image (high), and has a very smooth appearance in the visible image. By contrast, cirrus exhibits a more fibrous nature, a good example being Figure 1.3(b). The more mottled high cloud in Figure 1.3(a) on the eastern side of the cirrostratus wedge is cirrocumulus. Some cumulus is also apparent to the south-east of Figure 1.3(a) and underneath the cirrus to the south-west of Figure 1.3(b).

When convective activity occurs, and the air pockets meet a stable inversion layer at a low level, the convective cells may be prevented from rising further and as a result spread horizontally so that they merge, forming a layer of stratocumulus. An example is shown in Figure 1.4(a). The texture is more uniform than for cumulus, although the convective cells are still apparent due to the variation in albedo caused by the varying thickness of cloud. The low brightness temperature apparent from the infra-red image shows the low-level nature of stratocumulus. If the air is allowed to continue to rise, deep convective cumulonimbus may be formed. Figure 1.4(b) shows an area of cirrostratus dispersed with pockets of cumulonimbus

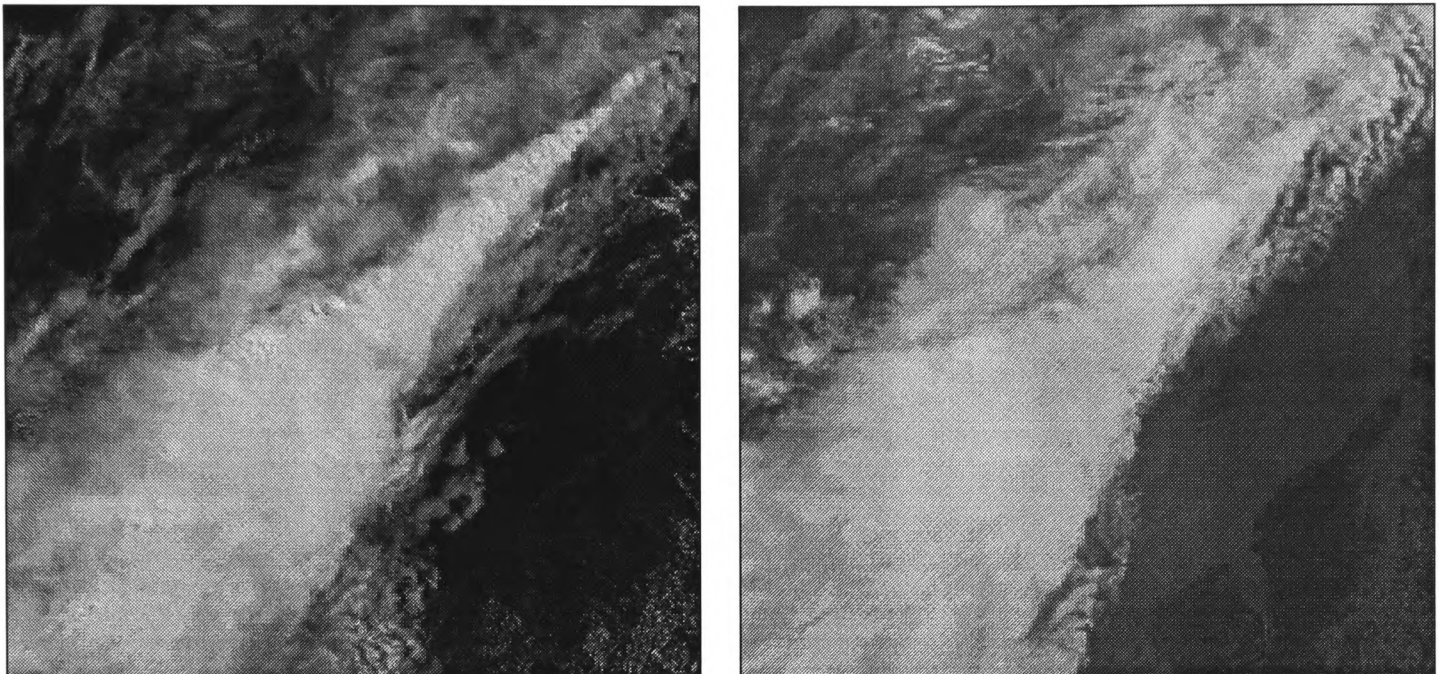


(a) Visible channel

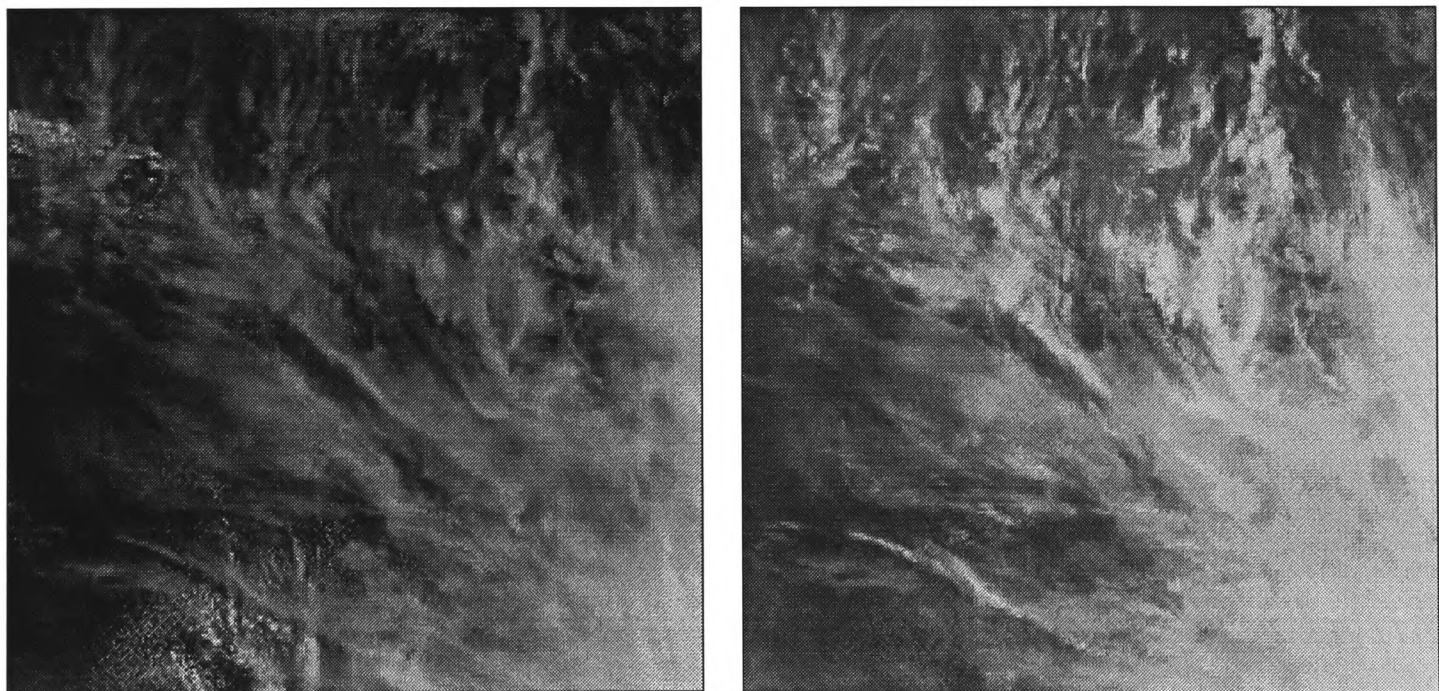


(b) Infra-red channel

Figure 1.2: An example Meteosat image pair from 6 January 1994 at 1200 GMT over western Europe. The image is 512 pixels wide, which is approximately 2000 km. See text for a fuller description of the clouds contained within the image pair, Appendix A for details of the Meteosat satellite, and Figures D.1 and D.2 (on Pages 206 and 207) for larger versions.

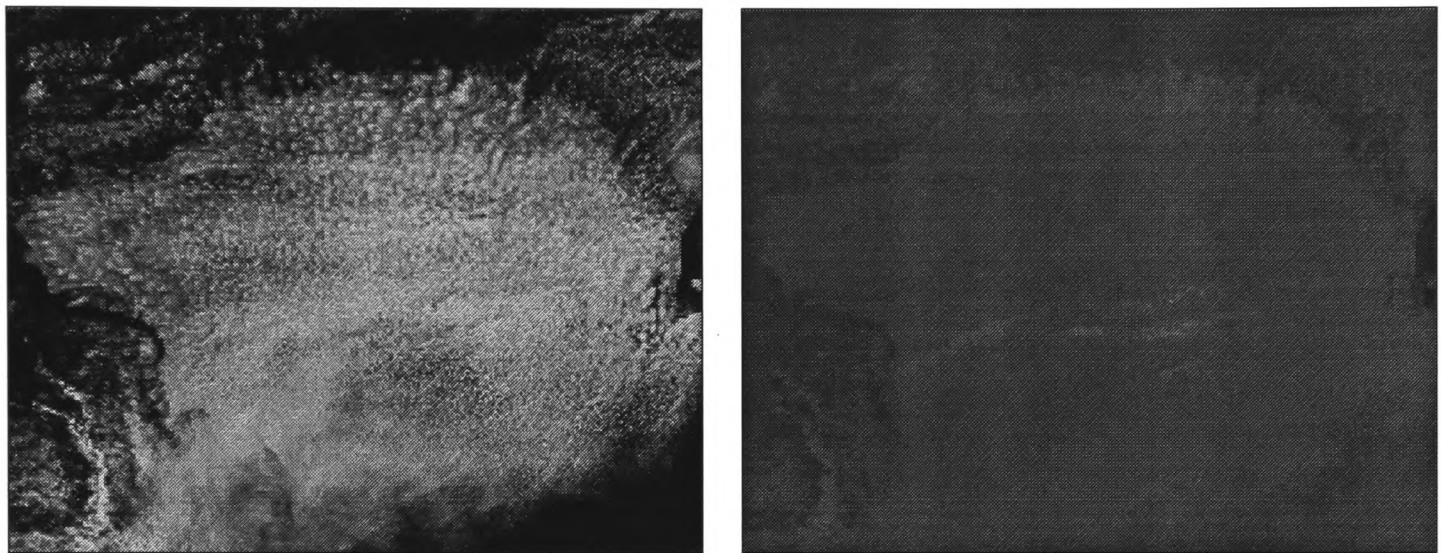


(a) 2 May 1989, Atlantic Ocean. Mainly cirrostratus, with some altostratus and cirrocumulus.

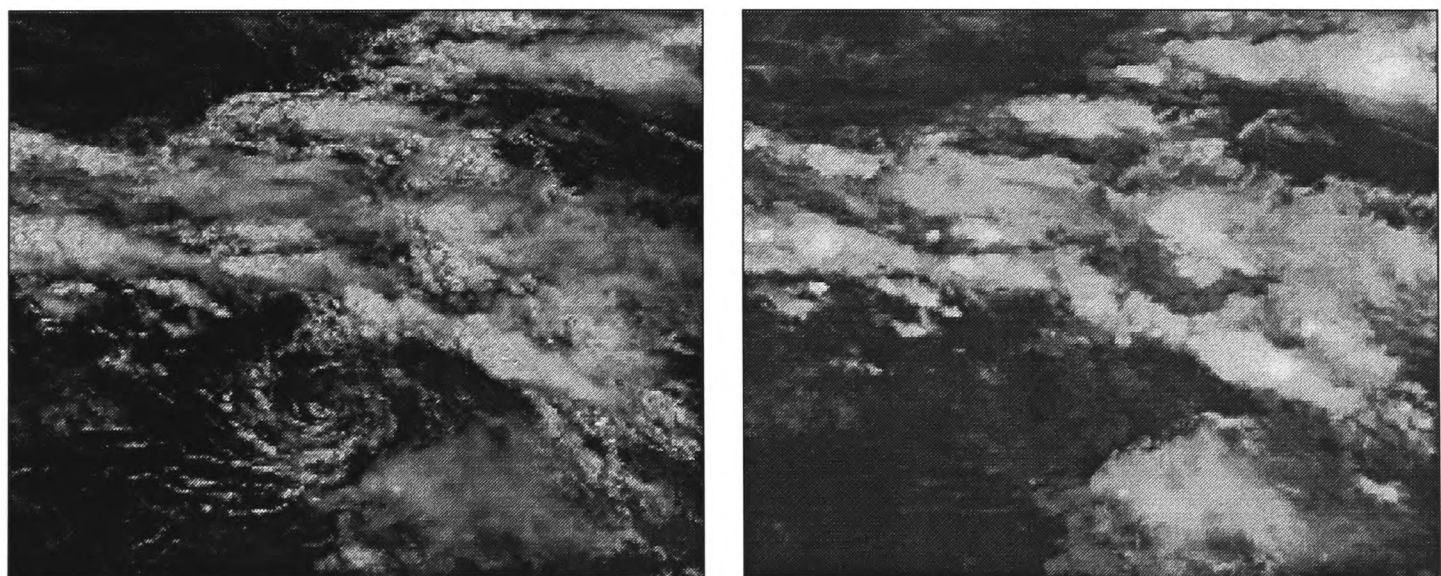


(b) 15 April 1989, Atlantic Ocean. Mainly cirrus.

Figure 1.3: Two AVHRR visible (left) and infra-red (right) image pairs from the Bankert database (see Section 1.4.1). Each image is 670 pixels wide, which is approximately 740 km. See text for a fuller description of the clouds contained within each image, and Appendix A for details of the AVHRR sensor.

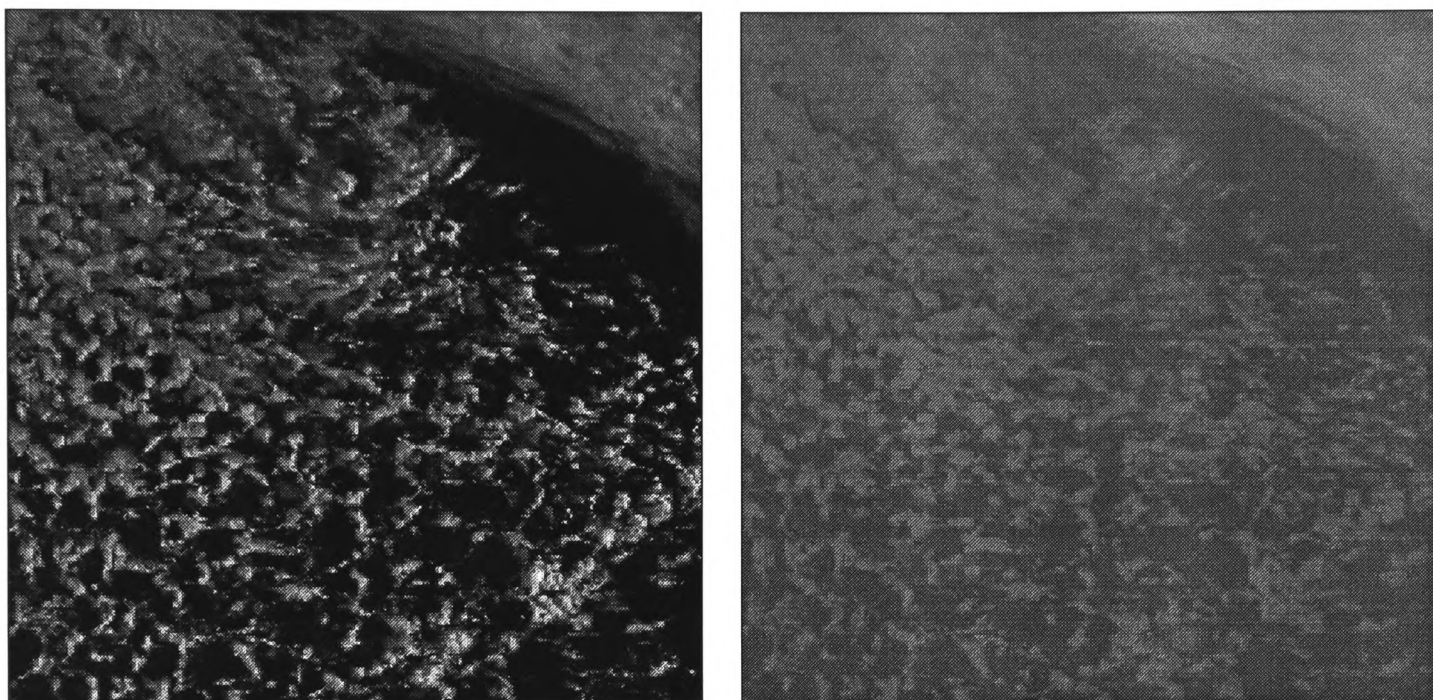


(a) 20 March 1989, China Sea. A sheet of stratocumulus.

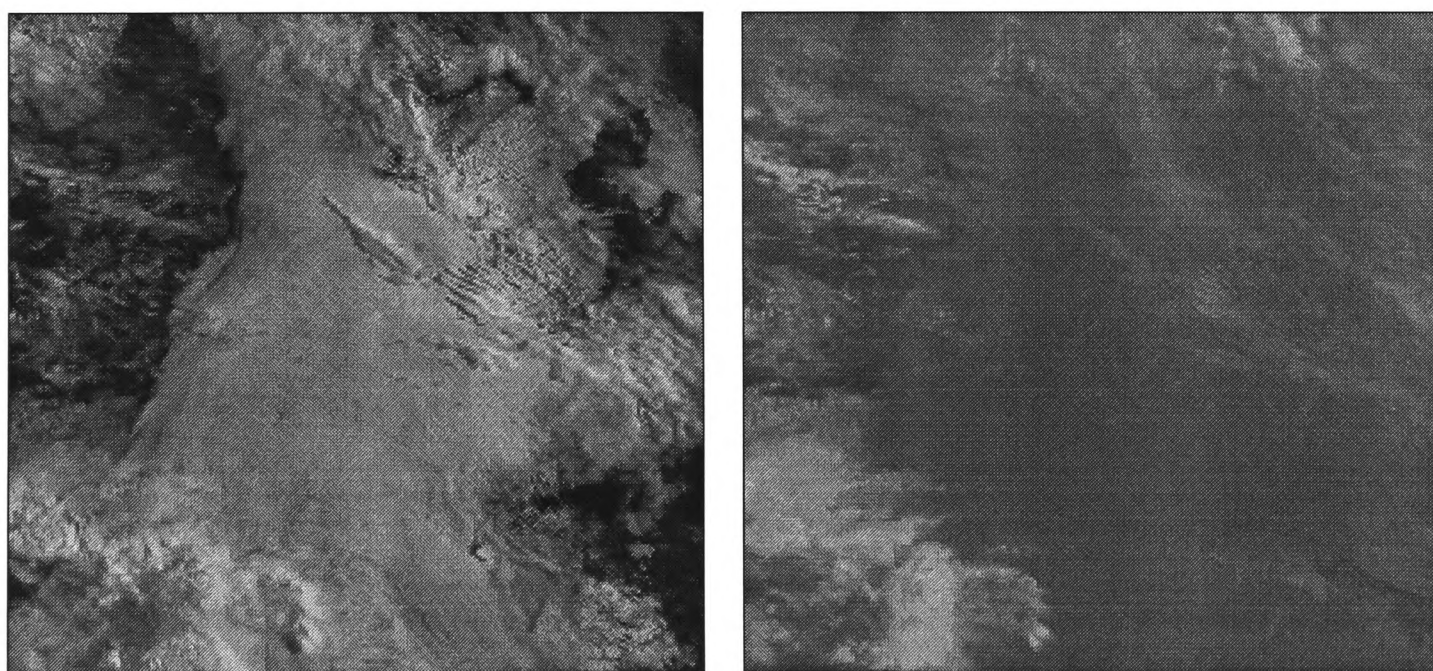


(b) 4 August 1989, China Sea. Contains cumulonimbus and cirrostratus.

Figure 1.4: Two AVHRR visible (left) and infra-red (right) image pairs from the Bankert database (see Section 1.4.1). Each image is 670 pixels wide, which is approximately 740 km. See text for a fuller description of the clouds contained within each image, and Appendix A for details of the AVHRR sensor.



(a) 15 April 1989, Pacific Ocean. Developing cumulonimbus cloud, with altostratus to north-east.



(b) 3 May 1989, Kuroshio Current. A sheet of Stratus covers the central area.

Figure 1.5: Two AVHRR visible (left) and infra-red (right) image pairs from the Bankert database (see Section 1.4.1). Each image is 670 pixels wide, which is approximately 740 km. See text for a fuller description of the clouds contained within each image, and Appendix A for details of the AVHRR sensor.

indicated by brighter regions in the infra-red image. Figure 1.5(a) shows convective cells which are developing into cumulonimbus, still being at a medium level as demonstrated by the infra-red image. The uniform area to the north-east is altostratus, a medium-level cloud with a fairly smooth texture.

Compare the visible images of Figures 1.3(a) and 1.5(b). They show a similar brightness and smoothness of texture. However, the infra-red channels indicate a marked difference, the former being high-level cirrostratus, and the latter being low-level stratus. This is a good illustration of the usefulness of the infra-red channel for separating certain cloud types.

1.3 Schemes for automated satellite image interpretation

In December 1993, the Satellite Image Applications Group of the UK Meteorological Office initiated a project whose aim was to develop a scheme for the detection and identification of clouds and cloud systems from meteorological satellite imagery, using pattern recognition techniques [71, 73]. The way in which such a scheme might be developed was left quite open, but some overall objectives were set out. An initial survey into pattern recognition techniques for cloud analysis was conducted by Dr Pankiewicz, of the UK Meteorological Office [71, 73]. A number of areas for future work as part of the project were suggested, and are summarized here (in no particular order):

- Production of a large training set for cloud-type recognition and synoptic feature identification.
- Determination of the best features³ calculated from the images for optimum classification.
- Investigation into the use of the Kohonen neural network.
- Development of a neural network to classify clouds.
- Research into other products from satellite imagery, such as precipitation estimation.

³The term “feature” in this context refers to the textural and spectral characteristics calculated from the image pixels for use as input to a classifier. See Chapter 3 for details.

- Development of a methodology for a pattern recognition system capable of identifying synoptic-scale meteorological objects.
- Investigations into the assimilation of pattern recognition products from satellite imagery into NWP models.

A major goal of the project was seen to be the automated identification of synoptic-scale features. A bottom-up approach to cloud system identification was set out by Pankiewicz [71, 73], whereby processing is done first at a local level, and then these local results are exploited for broader analysis. A three-level scheme was proposed, whereby simple cloud classes are first defined locally, followed by a broad segmentation of the image. Textural properties are then calculated over each region (avoiding the problem of calculating texture over boxes of a fixed size), and used as input to a second classifier together with the class from the first stage. The final classifier would attempt to allocate each region to a specific synoptic-scale system. Following this scheme, a neural network classifier was developed to classify clouds at a local level [72, 75], the first stage of the scheme. Work was subsequently done towards the partial realization of the second stage [76], the output from the first classifier being used to determine the size of region over which to calculate the textural information for the second classifier.

A neural network is currently being used in operational trials as part of the thunderstorm flood forecasting system, GANDOLF⁴ [74, 77]. The neural network estimates the cloud type (relating to airmass) within the satellite imagery, and this is then used (in a separate stage) to decide which of two forecast methods should be used – an extrapolative one for dynamic cloud conditions or an object-oriented developmental/decay one for convective situations. The production of a training set for airmass classification was undertaken due to the need for a suitable training set for use in this system. The training set is also used for this thesis and is described in Section 1.4.2.

The progress of the project was reviewed by Pankiewicz in January 1997 [78]. He also

⁴Generating Advanced Nowcasts for Deployment in Operational Land-based Flood forecasts.

reported on the construction of a feature selection and evaluation tool, and considered a number of pattern recognition products that might be of benefit to NWP models at the UK Meteorological Office, including cloud cover estimation and cloud-type determination. There has been a shift in the emphasis of the project recently from providing new tools for the forecaster to the direct use of pattern recognition products in the NWP model, via data assimilation. Currently, the best candidates at the mesoscale⁵ are cloud cover and precipitation estimation. It is planned to use both of these in Nimrod, a nowcasting⁶ system at the UK Meteorological Office, the former to detect cloud that is difficult to identify using thresholding techniques (such as thin cirrus or stratocumulus at night), and the latter to improve the precipitation analyses outside the areas covered by radar, or to delete false radar echoes. At the synoptic scale, the estimation of potential vorticity from water vapour and infra-red imagery is seen as being potentially important. Potential vorticity is a diagnostic used to understand the dynamics of the atmosphere, and can be important for initiating the correct dynamical state of the atmosphere in the NWP model.

The work for this DPhil project is being supported by a Cooperative Award in Science and Engineering (CASE) from the Engineering and Physical Sciences Research Council (EPSRC), with the industrial collaborator being the UK Meteorological Office. This work was not intended to be part of the project described above, but naturally fitted alongside it, so that parallel but distinct research could be undertaken. This research is concerned with the aspects of the UK Meteorological Office's project relating to cloud-type recognition and synoptic-scale analysis of cloud fields.

This thesis broadly follows the bottom-up approach suggested by Pankiewicz, proceeding from local to large-scale processing, and makes major use of neural network methods. The neural network model used for the UK Meteorological Office studies is called the multi-layer perceptron (see Chapter 2), which is also considered in detail here. In addition, the use of

⁵*Mesoscale* implies a spatial scale from a few kilometres to several hundred kilometres, and a temporal scale of approximately one to twelve hours. Structures on these scales are often apparent within synoptic-scale systems.

⁶Forecasting in the very short-term, up to approximately six hours ahead.

a radial basis function neural network has been investigated, and compared with the results of a simpler, linear technique as well as a method that takes into account the local spatial context of each pixel. A major component of this thesis is the analysis of the optimum set of features for cloud-type classification, using a different technique to that considered by the UK Meteorological Office.

One of the UK Meteorological Office project's original objectives was the consideration of unsupervised techniques such as the Kohonen neural network (see Chapter 4). Unsupervised methods work without the need for class labels supplied by a human expert, and are useful for the analysis of the features and datasets being used. A number of techniques, including the Kohonen network, are investigated in this thesis for the thorough analysis of the data. Working towards the UK Meteorological Office's objective of the automated recognition of synoptic-scale features, large-scale segmentation techniques are also considered towards the end of the thesis. These techniques act on the results of processing at a local level, which fits in with the scheme suggested by Pankiewicz [78].

In summary, the research described in this thesis concerns principally the application of neural network techniques to the problem of cloud-type determination and large-scale segmentation of cloud images, considering in detail each stage in the process. The research complements that being done at the UK Meteorological Office which has been described in this section. As the work covers a number of distinct areas, previous work conducted by other researchers is summarized in the appropriate parts of the thesis, mainly in Sections 3.1, 3.2, 5.1, 6.2.1 and 7.1.

1.4 Datasets used

Two sets of images have been used for this study. The first is a set of images from the Meteosat geostationary satellite, and the second is a set from a National Oceanic and Atmospheric Administration (NOAA) polar-orbiting satellite. Details of these satellites and the sensors used for the capture of the images can be found in Appendix A.

The two sets of images constitute a total of 458 visible and infra-red image pairs, or approximately 700 megabytes of information. For the automatic classification of regions of image into cloud type, as described in Section 1.3, a set of regions that have been expertly labelled according to cloud type is necessary for the purposes of setting the parameters of the classifier. Sets of labelled samples have been provided for each dataset, which are also described in this section.

1.4.1 The Bankert database

Following contact between the UK Meteorological Office and the Naval Research Laboratory (NRL) in Monterey, California, a set of images has been obtained, along with a set of labelled regions. The data have kindly been supplied by Dr Bankert of the NRL. The images are from a NOAA satellite, which has on board a second-generation Advanced Very High Resolution Radiometer (AVHRR), giving sub-satellite resolution of 1.1 km. For further details of this satellite, see Appendix A. This database is the same as that used by Bankert for various cloud studies [1, 5, 6].

The database consists of 95 AVHRR images for channels one (in the visible range) and four (infra-red), obtained between October 1988 and October 1989 (although one is from January 1988). They are taken from seven maritime regions in the Northern Hemisphere and therefore are limited neither to a specific time of year nor to a particular region. The seven maritime regions are the Atlantic Ocean, Pacific Ocean, Mediterranean Sea, Persian Gulf, Barents Sea, Kuroshio Current and China Sea.

Channel one is calibrated to albedo and channel four to brightness temperature, and both channels are scaled to grey-levels between 0 and 255. This range of values is representative of 0-100% albedo and 185-312 K respectively. Channel one is also normalized by the cosine of the solar zenith angle⁷ to correct for different solar elevations. Only images that were taken with a solar zenith angle of less than 75° were used.

⁷Angle subtended from the local vertical by the sun.

Table 1.2: The cloud classes from the Bankert dataset. Listed are the classes used along with their common abbreviation and the number of samples for each class.

Cloud type	Code	Total
Cirrus	Ci	212
Cirrocumulus	Cc	72
Cirrostratus	Cs	166
Altostratus	As	154
Nimbostratus	Ns	59
Stratocumulus	Sc	248
Stratus	St	225
Cumulus	Cu	118
Cumulonimbus	Cb	128
Clear	Clr	220

Sample areas of 16×16 pixels have been expertly labelled as one of the ten cloud classes listed in Table 1.2; note that a “clear” category is included. The 16×16 pixel sample had to be covered by at least 75% of a particular cloud type for it to be used (50% for Stratocumulus), and all samples were taken over sea so that the relatively bright and warm land does not interfere with the study. An initial collection of 610 samples from four images was provided by experts at the Naval Postgraduate School (NPS), and expanded to a total of 1633 by experts at the NRL [24]. For the NPS samples, two experts independently classified each region, and only those regions that had both experts in agreement on the class were used for the dataset. For the NRL portion of the dataset, only samples on which four experts had a majority agreement were used [24], this being only 59% of the original samples. This percentage shows the difficulty posed even to human experts.

Following an examination of the images for this thesis, the infra-red bands of two of the images from the China Sea were found to be wholly or partly missing. Therefore these two images, and their labelled samples, were removed. In addition, some samples were found to be over land, so these were also removed. This left a total of 93 images and 1592 labelled samples.

Table 1.2 shows how the 1592 samples are divided up into the various classes. The dataset is not equally balanced between the classes, and the proportions are not intended to

be representative of the probabilities of finding each cloud type in nature.

Note that channel two of the AVHRR instrument is usually the one referred to as the “visible” channel even though it is actually in the near infra-red beyond the range of the human eye. However, in this thesis the term “visible” will refer exclusively to channel one which *is* in the visible range of humans.

1.4.2 The Meteosat database

The samples were taken from a set of 365 Meteosat daytime visible and infra-red image pairs collected over the course of 1994 [75, 77], in the period from two hours after local sunrise to two hours before local sunset. These come from the UK Meteorological Office’s Autosat-2 automatic processing system [2] (now superseded by Autosat-3), in an area covering north-west Europe, the North Sea and part of the eastern Atlantic Ocean. The visible channel is calibrated to albedo and the infra-red channel to brightness temperature, and both channels are scaled to grey-levels between 0 and 254 (the value 255 indicates missing data). This range of values is representative of 0-100% albedo and 198-308 K respectively. The visible channel is also normalized by the cosine of the solar zenith angle to correct for different solar elevations.

There is an image pair from each day of the year, except for nine days in September where the quality of data was poor due to a data format change. An extra image was taken from each of the five days before and the four days after that period to replace the poor quality ones. The images have been re-projected into polar stereographic coordinates, using the nearest neighbour method to assign grey-levels to the re-projected pixels, such that pixel resolution at 60°N is 3.84 km. The size of each image is 640×640 pixels, which is approximately 2460 km across.

An expert at the UK Meteorological Office has labelled 2190 17×17 pixel regions into one of the five classes shown in Table 1.3. Each class is not equally represented in the dataset; however the numbers were chosen to be representative of the distribution actually found in

Table 1.3: The cloud classes from the Meteosat dataset. Listed are the classes used along with their abbreviation and the number of samples for each class.

Cloud type	Code	Total
Clear sea	Sea	225
Clear land	Land	207
Dynamic cloud	Dyn	986
Shallow Convective cloud	Shal	652
Deep Convective cloud	Deep	120

the imagery. The samples have been labelled according to the physical nature of the airmass present, rather than using the more subjective labels such as Cirrus and Cumulus which are used for the Bankert database described in the previous section.

1.5 Software used

An integrated system has been developed for this thesis for the display and analysis of satellite imagery. Called *NetSat*, it has been designed to be modular in order to facilitate the addition of new image and sample databases, the production of new algorithms, and their integration into the system.

NetSat has been developed in the C++ programming language, which is an object-oriented language, using the *TargetJr* integrated image understanding environment. *TargetJr* is a C++ library produced from a collaboration between the Visual Geometry Group at Oxford, GE/CRD⁸ in the USA, and INRIA⁹ in France. *TargetJr* provides a comprehensive user interface framework, as well as a broad suite of optional image processing algorithms.

The algorithms and methods used in and developed for this thesis have been organized into modules and integrated into the *TargetJr* environment. They can be broadly divided

⁸General Electric Corporate Research and Development.

⁹Institute National de Recherche en Informatique et en Automatique.

into three major types:

- Data modules:

Management of the satellite image databases and their samples.

Extraction of features (see Chapter 3) from the pixel information.

Creation and management of datasets.

- Algorithm modules:

Neural network methods.

Segmentation algorithms.

- Interface modules:

Display of images.

Display of coloured overlays for various purposes.

The algorithm, data and interface modules are designed to interact with each other, exchanging information as required. These functions are in addition to those provided by *TargetJr*.

An example screen-shot is shown in Figure 1.6, showing some of the features of *NetSat*. The functions in the pull-down menus from *File* to *Enhance* are provided by the *TargetJr* environment, while the rest have been developed as part of the work presented in the rest of the thesis. There is the facility to zoom in and pan across the displayed image, to change the displayed spectral channel, as well as to load multiple images and switch between them instantly.

1.6 Outline of the thesis

The specific aims of the work presented in this thesis have been outlined in Section 1.3, and represent a thorough investigation into each aspect in the process from raw spectral information to large-scale definition of cloud regions. Each stage in the process is dealt with separately in consecutive chapters, providing a natural and self-contained progression from initial aims to completed goals.

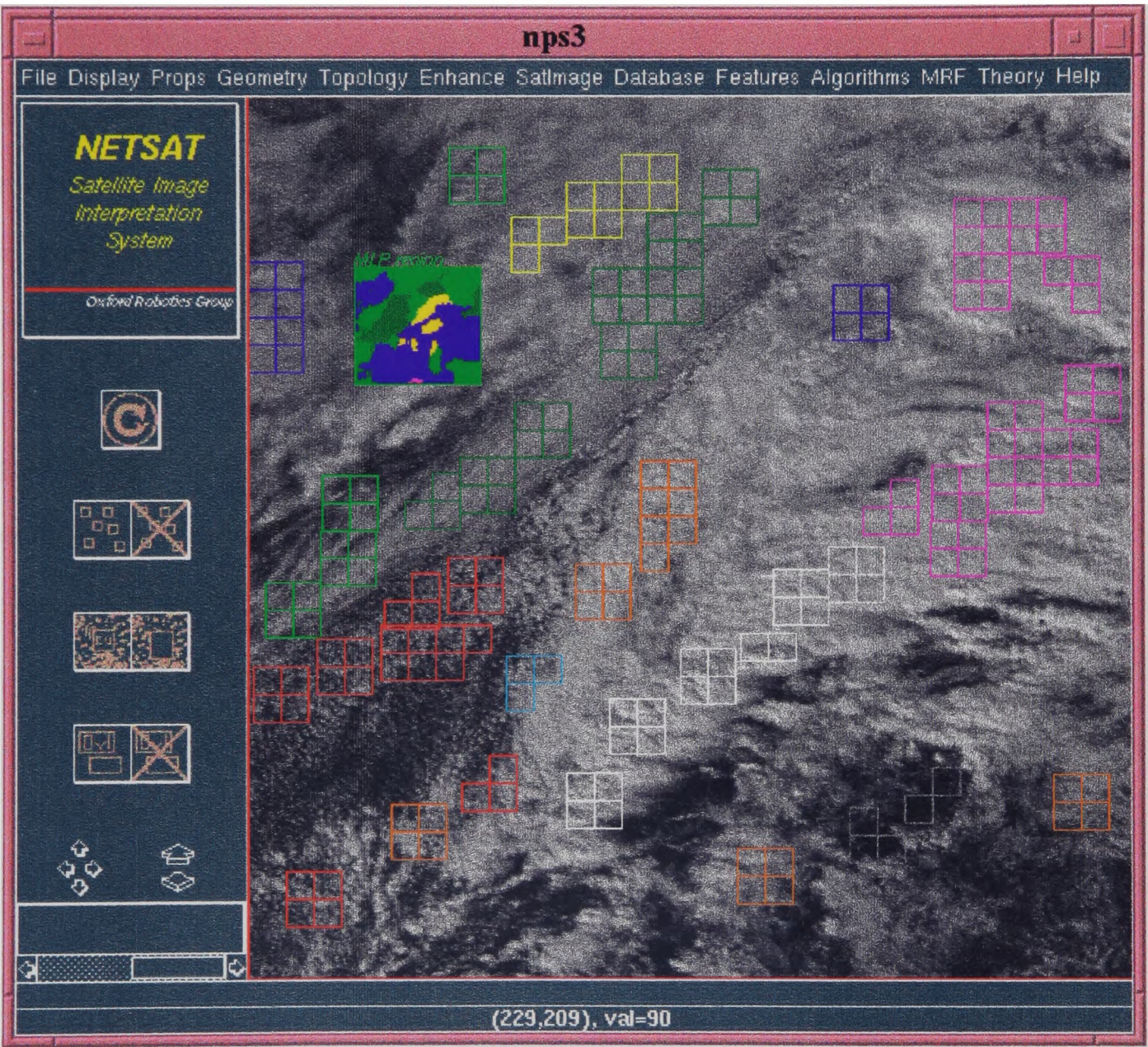


Figure 1.6: A screen-shot from the *NetSat* system. The image seen is the visible image from the Bankert database that is shown in Appendix D on Page 214. All labelled samples for this image from the Bankert database (described in Section 1.4.1) are shown as coloured boxes, each colour representing a different class (a key is available via a menu command). Also displayed is a small colour overlay showing the results from a neural network classification for each pixel (see Chapter 6). The buttons on the left-hand side access various functions directly, and there is the facility for zooming and panning the displayed image to the lower left.

The theory of the two main neural network models considered, the multi-layer perceptron (MLP) and the radial basis function (RBF) network, is presented in Chapter 2 within the framework of statistical pattern recognition. For good classification results, textural and spectral information (features) must be extracted from the raw grey-levels of the images, followed by a feature selection method to determine the most useful set of features. This is dealt with in Chapter 3. It is important for the design of any cloud classification system to gain a full understanding of the dataset being used. Chapter 4 presents three techniques for visualizing points in the dataset, and how the cloud classes are related to each other, which provide important information for the analysis of the results of the subsequent chapters.

Having considered the neural network models, the extraction of information from the images, and the analysis of this information, Chapter 5 deals with the classification of the cloud samples from each of the databases described in Section 1.4. Chapter 6 is a natural extension of this, applying the classifier to every pixel from a number of satellite images. The limitations of such an approach are discussed, and a Markov Random Field approach is introduced as an alternative. Chapter 7 draws from all the results of previous chapters, and addresses the problem of large-scale segmentation of the satellite image into distinct regions of cloud type. Finally, conclusions and suggestions for future work are presented in Chapter 8, as well as a summary of the main contributions of this work.

The technical details of the satellites and the sensors used for the capture of the images used in this thesis are found in Appendix A, while Appendix B gives the detailed theory of the multiscale Markov Random Field model which is used in Chapter 6. Various mathematical methods required for the segmentation work of Chapter 7 are presented in Appendix C. Full-size versions of almost all of the images used for the display of results throughout the thesis are in Appendix D, showing both the visible and infra-red images as well as details about the date and time of the images. As already mentioned, the work in this thesis covers a number of distinct areas. For this reason, previous work conducted by other researchers is summarized in the appropriate parts of the thesis, mainly in Sections 3.1, 3.2, 5.1, 6.2.1 and 7.1.

Chapter 2

Bayesian classification methods and neural networks

Two different models of neural network have been used for this work, the multi-layer perceptron (MLP) and the radial basis function (RBF) neural network. The biology of the animal nervous system may be the inspiration for certain aspects of these models [27], but they are best understood from a mathematical perspective and it is important to place them within the framework of statistical pattern recognition theory [8, 32, 88].

2.1 Bayes classifiers

If a set of m measurements $x_1, x_2, \dots, x_m$, represented by the vector $\mathbf{x}$, is made on a sample drawn from the general population, it is the goal of a classifier to assign this sample correctly to one of a pre-determined number of classes. Let the assigned class be denoted by C_i , where i is the class index. In its most general form, a classifier may be formulated in terms of a set of *discriminant functions* [8]. For an n -class problem, there will be n such functions $g_1(\mathbf{x}), \dots, g_n(\mathbf{x})$. A simple decision rule is then used to assign $\mathbf{x}$ to a class: if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$, then $\mathbf{x}$ is assigned to class C_i .

Bayes' theorem can be stated as follows, using a lower-case p for probability density

functions and an upper-case P for discrete probabilities:

$$P(C_i | \mathbf{x}) = \frac{p(\mathbf{x} | C_i)P(C_i)}{p(\mathbf{x})} \quad (2.1)$$

where $P(C_i | \mathbf{x})$ is the *a posteriori* (or simply *posterior*) probability that the class is C_i given the values of the measurements $\mathbf{x}$. $P(C_i)$, the *a priori* (or simply *prior*) probability of class i , may either be calculated on theoretical grounds or based on prior observations. Note that $p(\mathbf{x})$ is a normalization factor to ensure that the posterior probabilities sum to one, which leads to $p(\mathbf{x}) = \sum_i p(\mathbf{x} | C_i)P(C_i)$, where the sum is over all classes.

Bayes' theorem expresses the posterior probability in terms of quantities that are easier to estimate. Assuming for the moment that these quantities can and have been estimated, it is clear that $P(C_i | \mathbf{x})$ can be used as the discriminant function for class C_i , so that $g_i(\mathbf{x}) = P(C_i | \mathbf{x})$. The decision rule is now

$$\text{Assign } \mathbf{x} \text{ to class } C_i \text{ if } P(C_i | \mathbf{x}) > P(C_j | \mathbf{x}) \text{ for all } j \neq i. \quad (2.2)$$

If this rule, the Bayes decision rule, is used for classification, then it can be shown that the probability of misclassification is minimized [8, 32]. A classifier that uses this rule is known as a *Bayes classifier*.

Decision surfaces can also be defined which divide the input space into *decision regions*. The surface separating classes i and j is given by the equation

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \quad (2.3)$$

$$\text{or } P(C_i | \mathbf{x}) = P(C_j | \mathbf{x}). \quad (2.4)$$

2.1.1 Estimating the class-conditional densities

The task of estimating the class-conditional density $p(\mathbf{x} | C_i)$ on the right-hand side of Equation 2.1 may be non-trivial. For a simple one- or two-dimensional problem with a large

quantity of measurements, it is possible to use histograms of the data within each class for this purpose [8]. For any higher dimensions, a choice of a histogram bin size that gives reasonable resolution along any one axis would lead to far too many bins; most bins would either be empty or contain only a few points. This is referred to as the *curse of dimensionality* [8].

Hence for most practical applications, the solutions fall into one of the following three categories:

- Make an assumption about the distribution of data within a class. The Gaussian density function is most commonly used for $p(\mathbf{x} | C_i)$. These are the so-called “parametric methods” because the model is based upon parameters estimated from the data.
- Estimate $p(\mathbf{x} | C_i)$ using non-parametric methods. Here, no model is assumed; instead some other approximation is used.
- Do not compute $p(\mathbf{x} | C_i)$, but estimate the posterior probabilities $P(C_i | \mathbf{x})$, which are the discriminant functions $g_i(\mathbf{x})$, directly from the data.

The first two solutions will be discussed briefly in the next two sections. The MLP falls into the third category, and will be covered in more detail in a separate section later in the chapter.

2.1.1.1 Parametric approach

As stated above, the most common function used to model $p(\mathbf{x} | C_i)$ is the Gaussian distribution:

$$p(\mathbf{x} | C_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) \right\} \quad (2.5)$$

for a d -dimensional feature vector $\mathbf{x}$. The covariance matrix Σ_i and mean vector $\boldsymbol{\mu}_i$ are estimated from the data for class i only.

The discriminant function is given by:

$$g_i(\mathbf{x}) = P(C_i | \mathbf{x}) = p(\mathbf{x} | C_i) P(C_i) \quad (2.6)$$

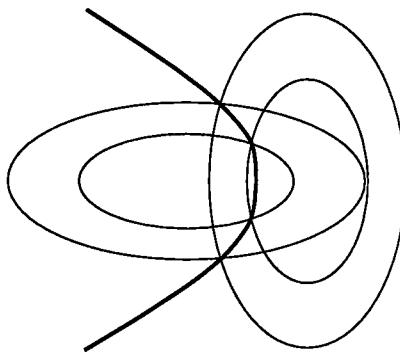


Figure 2.1: Quadratic decision surface between two Gaussian distributions. Contours of equal probability for each class are shown.

from Equation 2.1, dropping the $p(\mathbf{x})$ since it is a normalizing factor common to all classes. Since only the magnitude is needed, any monotonic function may be applied to this equation. Taking logarithms, the following expression is obtained:

$$g_i(\mathbf{x}) = \ln p(\mathbf{x} | C_i) + \ln P(C_i). \quad (2.7)$$

Inserting the Gaussian distribution for $p(\mathbf{x} | C_i)$ from Equation 2.5, and dropping constant terms, the expression becomes:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(C_i). \quad (2.8)$$

This is a *quadratic* discriminant function, and the decision surfaces are therefore also quadratic [32]. For example, in two dimensions the surface is a *conic section*, either a parabola, hyperbola, ellipse or straight line. An example for a simple two-dimensional case is shown in Figure 2.1.

If it is further assumed that the covariance is the same for each class, then $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}$. All the terms in Equation 2.8 that are now class-independent (i.e not having the subscript i) can be dropped, leaving only terms linear with $\mathbf{x}$:

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0} \quad (2.9)$$

where

$$\mathbf{w}_i^T = \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1} \quad (2.10)$$

$$w_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i + \ln P(C_i). \quad (2.11)$$

This gives rise to *linear discriminants* where the decision surfaces are hyperplanes in input space.

2.1.1.2 Non-parametric approach

The k -nearest neighbour classifier [8] is an example of a non-parametric method based on the Bayes discriminant function (or decision rule), and is the only such example that will be outlined here.

Consider a dataset containing n feature vectors, with n_i vectors within each class i . The class-conditional densities are estimated by considering a small hypersphere centred at $\mathbf{x}$. If the volume V of this hypersphere is chosen such that it encloses exactly k data points, with k_i from class i , these values can be used to estimate the probabilities required [8]:

$$p(\mathbf{x} | C_i) \approx \frac{k_i}{n_i V} \quad (2.12)$$

$$p(\mathbf{x}) \approx \frac{k}{n V} \quad (2.13)$$

$$P(C_i) \approx \frac{n_i}{n}. \quad (2.14)$$

Inserting these into Bayes' formula, the discriminant function is obtained:

$$g_i(\mathbf{x}) = P(C_i | \mathbf{x}) = \frac{p(\mathbf{x} | C_i) P(C_i)}{p(\mathbf{x})} \approx \frac{k_i}{k}. \quad (2.15)$$

Thus a new sample will be assigned to the class that has the greatest number of data points within the sample's hypersphere.

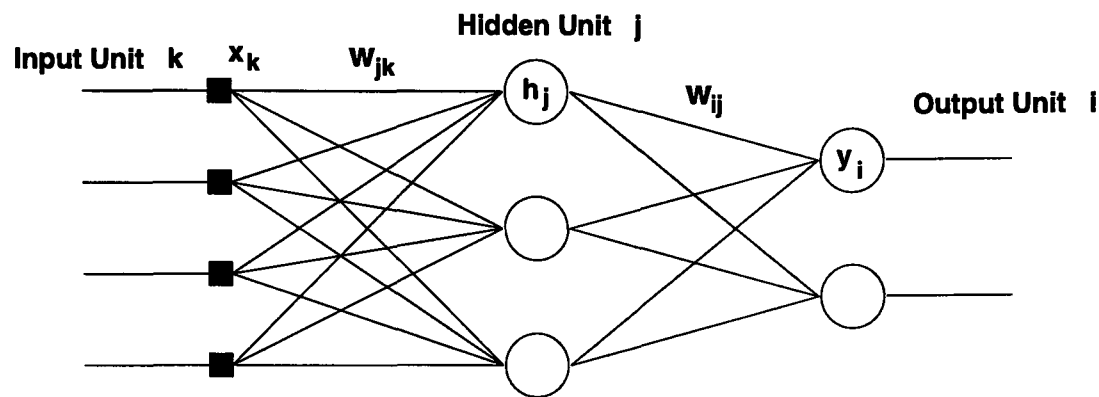


Figure 2.2: The structure of a multi-layer perceptron (MLP). The bias units are not shown in the diagram.

2.2 The MLP

Although at first sight the MLP seems a very different approach to pattern classification compared with the Bayesian methods, it does share some important underlying principles. As will be illustrated later, an MLP computes a set of decision surfaces like the Bayesian methods, but without the need for any simplifying assumptions about the density functions. Furthermore, for a particular type of MLP architecture and training procedure, the outputs of the MLP are direct estimates of the posterior probabilities $P(C_i | \mathbf{x})$.

The theory of the MLP dates back to the work of McCulloch and Pitts [65] in 1943 who investigated the use of threshold logic units. Single-layer networks of these threshold units were then studied by Rosenblatt [90] in 1958 under the name *perceptrons*, and by Widrow and Hoff [116] in 1960 who called them *adelines*. These can be combined into multiple layer perceptrons (MLPs), the training of which remained a problem until the work of Werbos [114] in 1974.

2.2.1 Structure of the MLP

The basic structure of an MLP is shown in Figure 2.2. It creates a non-linear mapping from an input vector to an output vector, the internal elements enabling this mapping to be achieved. The input units (denoted by squares) perform no function other than to distribute the applied inputs to all subsequent units in the hidden layer via the w_{jk} weights. The units in the hidden and output layers (denoted by circles) perform a weighted sum of the inputs

to that unit, before a non-linear function f is applied to the resulting sum:

$$h_j = f \left(\sum_k w_{jk} x_k + \text{bias}_j \right) \quad (2.16)$$

$$y_i = f \left(\sum_j w_{ij} h_j + \text{bias}_i \right) \quad (2.17)$$

where bias_j and bias_i are the *bias* terms for each unit. These bias terms can be incorporated into the weight matrices w_{jk} and w_{ij} by specifying that the input unit x_0 and hidden unit h_0 are *bias units* having a constant output of 1.0, and the weights w_{j0} and w_{i0} are the bias weights bias_j and bias_i . The justification for these terms comes from the presence of the w_{i0} term in Equation 2.9. The weighted sums now simplify to:

$$h_j = f \left(\sum_k w_{jk} x_k \right) \quad (2.18)$$

$$y_i = f \left(\sum_j w_{ij} h_j \right). \quad (2.19)$$

The form of the function f will be explained later.

Variations include having only partial inter-connection between units in adjacent layers, and applying a *linear* function at the output layer. It is also possible to use more than one hidden layer. However, MLPs with one hidden layer have been shown to be capable of approximating *any* non-linear mapping (the *universal approximation* property [26]), although one hidden layer is not necessarily optimal in terms of ease and duration of training, nor is it possible to predict how many units are required in the hidden layer to approximate a given mapping.

In the discussion that follows, the following notation will be applied. Units in the input, hidden and output layers are identified by the letters k , j , and i respectively. Input values are denoted by x_k . Since the input layer performs no mapping, x_k is just the component of the input vector applied to that unit. The values at the outputs of the hidden and output layers are h_j and y_i respectively. The desired or “target” response is t_i . The weight for the

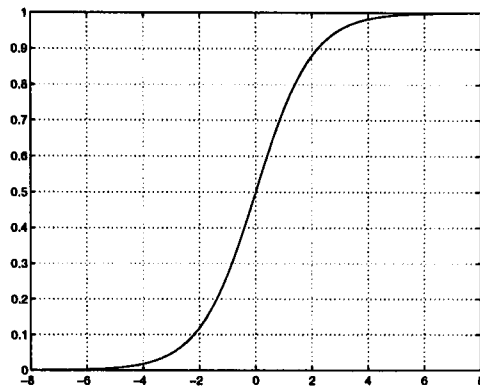


Figure 2.3: The sigmoid function of Equation 2.21.

connection between unit k in the input layer and unit j in the hidden layer is w_{jk} . The non-linear function used is denoted by f_σ . For instance the output of unit j in the hidden layer is given by:

$$h_j = f_\sigma \left(\sum_k w_{jk} x_k \right). \quad (2.20)$$

The most common non-linearity in use is the *sigmoid* function, shown in Figure 2.3, which is defined by:

$$f_\sigma(x) = \frac{1}{1 + e^{-x}}. \quad (2.21)$$

This is a differentiable function, which is vital to the training algorithm described next. Its derivative can easily be shown to be given by: $f'_\sigma(x) = f_\sigma(x)(1 - f_\sigma(x))$.

2.2.2 Training the MLP – the error back-propagation algorithm

The MLP's internal weights are set during a period of supervised training. A popular method is the method of error back-propagation, and any further mention of the term “MLP” will imply an MLP trained by error back-propagation. The basis of back-propagation was first presented by Werbos [114] in 1974 but was only brought to a wide readership by Rumelhart *et al* [94, 95] in 1986.

The following error function is introduced:

$$E = \frac{1}{2} \sum_i (y_i - t_i)^2 \quad (2.22)$$

which is the summed square error for one presentation of a training pattern $\mathbf{x}$ (with an arbitrary factor of $1/2$ which will simplify equations later). A *gradient-descent* algorithm is used to minimize this error. The error may be viewed as a function of the weights in the MLP:

$$E = E(w_1, w_2, \dots, w_N) \quad (2.23)$$

where the term w_i is one of the weights w_{ji} or w_{kj} and N is the total number of weights in the MLP. Note that this error surface is different for every input vector, because the surface is formed by considering the error for all possible weights for the *current* input vector.

Each weight will be perturbed by a small amount to change the error E :

$$\delta E = \frac{\partial E}{\partial w_1} \delta w_1 + \frac{\partial E}{\partial w_2} \delta w_2 + \dots + \frac{\partial E}{\partial w_N} \delta w_N = (\nabla E) \cdot (\delta \mathbf{w}) \quad (2.24)$$

where $\delta \mathbf{w}$ is the vector of weight changes. In order to achieve the maximum error decrease for a given magnitude of $\delta \mathbf{w}$, it must be in the opposite direction to ∇E :

$$\delta \mathbf{w} = -\alpha \nabla E \quad (2.25)$$

where α is the *learning rate*, which controls how large the step down the error surface is each time. The weights are updated after every presentation. An alternative, not used in this thesis, is to store up the updates and to change the weights only after presenting the entire dataset (the *batch* version of gradient descent). In addition to the above weight changes, an optional *momentum* term may be added, which is a fraction of the *previous* change for that weight. This usually helps to speed training.

Each of the two sets of weights associated with the hidden-layer and output-layer weights will now be considered in turn, deriving the rule for weight update.

2.2.2.1 The output layer

Considering individual weights, Equation 2.25 can be written as

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}. \quad (2.26)$$

Now recall Equation 2.22 and the fact that

$$y_i = f_\sigma \left(\sum_j w_{ij} h_j \right) = f_\sigma(a_i) \quad \text{where} \quad a_i = \sum_j w_{ij} h_j. \quad (2.27)$$

Using the chain rule on Equation 2.26, and evaluating the resulting parts using expressions given thus far, the expression for the weight change becomes:

$$\begin{aligned} \Delta w_{ij} &= -\alpha \frac{\partial E}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \\ &= -\alpha \delta_i h_j \end{aligned} \quad (2.28)$$

where

$$\begin{aligned} \delta_i &= \frac{\partial E}{\partial a_i} \\ &= \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial a_i} \\ &= \{y_i - t_i\} \{y_i(1 - y_i)\}. \end{aligned} \quad (2.29)$$

Equation 2.28 is used to adjust the weights in the output layer.

2.2.2.2 The hidden layer

In a similar way to the output-layer calculations, Equation 2.25 can be written as

$$\Delta w_{jk} = -\alpha \frac{\partial E}{\partial w_{jk}} \quad (2.30)$$

and

$$h_j = f_\sigma \left(\sum_k w_{jk} x_k \right) = f_\sigma(b_j) \quad \text{where} \quad b_j = \sum_k w_{jk} x_k. \quad (2.31)$$

The chain rule is again used on Equation 2.30:

$$\begin{aligned} \Delta w_{jk} &= -\alpha \frac{\partial E}{\partial b_j} \frac{\partial b_j}{\partial w_{jk}} \\ &= -\alpha \delta_j x_k \end{aligned} \quad (2.32)$$

where

$$\begin{aligned} \delta_j &= \frac{\partial E}{\partial b_j} \\ &= \frac{\partial E}{\partial h_j} \frac{\partial h_j}{\partial b_j} \\ &= \frac{\partial E}{\partial h_j} \{h_j(1 - h_j)\}. \end{aligned} \quad (2.33)$$

There is no simple expression like Equation 2.22 for the error E in terms of the hidden-layer outputs h_j , which there was for the output layer. However, use can be made of the fact that there is an expression for the delta values δ_i from Equation 2.29, and $\frac{\partial E}{\partial h_j}$ can therefore be

re-expressed in terms of this parameter:

$$\begin{aligned}\frac{\partial E}{\partial h_j} &= \sum_i \frac{\partial E}{\partial a_i} \frac{\partial a_i}{\partial h_j} \\ &= \sum_i \delta_i \frac{\partial a_i}{\partial h_j} \\ &= \sum_i \delta_i w_{ij}.\end{aligned}\tag{2.34}$$

Thus Equation 2.33 becomes:

$$\delta_j = \left(\sum_i \delta_i w_{ij} \right) \{h_j(1 - h_j)\}\tag{2.35}$$

and the weight update rule for the hidden layer is given by Equations 2.32 and 2.35. Note that it includes the error terms δ_i from the output layer, which explains the description “back-propagation of errors”. The sum term of Equation 2.35 is just the weighted sum of all output delta values going back into hidden unit j .

2.2.3 Output representation and decision rule

The exact form of the target output vectors t_i has not yet been specified. There are a number of different ways to encode a classification into a vector; the preferred one is the *one-of-m* encoding [87], where for m possible output classes, there are m outputs. The target vector is binary, with 0 in all elements except that of the designated class, which is set to 1.

With this output representation, it has been shown [87] that *once trained*, the outputs of the MLP are direct estimates of the Bayesian posterior probabilities $P(C_i | \mathbf{x})$. Thus the Bayesian decision rule can be used in order to choose the class on the basis of the values in the output vector:

$$\text{Assign } \mathbf{x} \text{ to class } C_i \text{ if } y_i > y_j \text{ for all } j \neq i.\tag{2.36}$$

In other words, assign the class to that represented by the largest output.

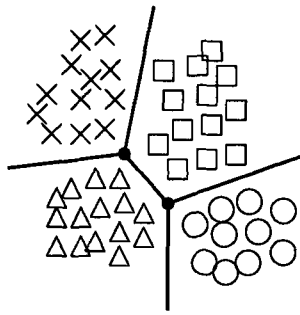


Figure 2.4: Representation of MLP decision surfaces. The planar decision surfaces are infinite in extent and arise from the the hidden-layer representation of the MLP.

In this way the data space is divided up into different output class regions. A significant difference between MLPs and RBFs, which are described in the next section, is the way in which the data space is segmented. An MLP uses non-linear combinations of the planar decision surfaces (formed by the hidden layer) to segment the space, as schematically illustrated in Figure 2.4, whereas the RBF uses localized hidden-layer kernel functions.

2.3 The RBF network

This section describes the architecture and training procedure for RBF networks. They are trained in a two-stage process by a mixture of supervised and unsupervised training, which can be useful when there is plentiful unlabelled data but fewer labelled data, since all the former can be used for training, unlike for the MLP. The RBF network can trace its origins to exact interpolation methods [86], and is now widely used in the neural network community [14, 68]. Despite the different architecture, there are many important similarities to the MLP, which are described in Section 2.3.3.

2.3.1 Structure of the RBF network

Superficially, the structure of an RBF network (see Figure 2.5) is similar to that of an MLP. The main difference lies in the hidden-layer representation. In the MLP, the weights for hidden unit j consisted of a vector used to perform the weighted sum of the inputs; in the

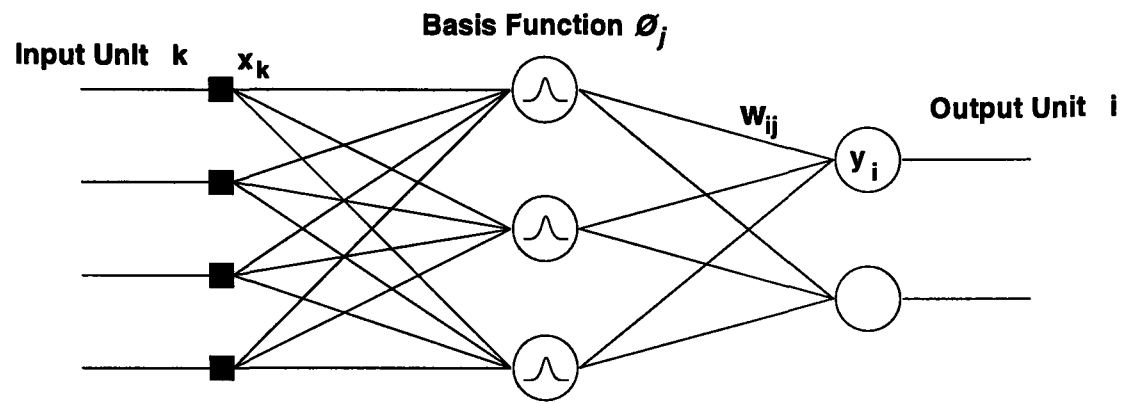


Figure 2.5: The structure of a radial basis function (RBF) network

RBF network the free parameters are the mean μ_j and the covariance Σ_j of a multivariate Gaussian $\phi_j(\mathbf{x})$,

$$\phi_j(\mathbf{x}) = \exp \left[-\frac{1}{2}(\mathbf{x} - \mu_j)^T (\Sigma_j)^{-1} (\mathbf{x} - \mu_j) \right]. \quad (2.37)$$

The output of each hidden unit is just $\phi_j(\mathbf{x})$, where $\mathbf{x}$ is the input vector. The output layer then simply performs the weighted sum of its inputs. Therefore the input-to-output mapping process can be written as:

$$y_i(\mathbf{x}) = \sum_j w_{ij} \phi_j(\mathbf{x}). \quad (2.38)$$

As for the MLP, a bias weight is usually included in the above sum; this may again be thought of as an extra hidden-layer node with a constant output of 1.0 connected to each output node. Note that the Gaussian function in Equation 2.37 is not normalized because the normalizing constant has been effectively absorbed into the weights w_{ij} in Equation 2.38.

2.3.2 Training the RBF network

As seen in the previous section, the two layers perform very different functions, and as a result they are trained separately, one after the other. First, the hidden-layer basis function centres and widths are determined in an *unsupervised* fashion. The training vectors are then mapped into the basis function space by feeding them through the first layer, and these mapped vectors are then used to train the output layer.

2.3.2.1 The hidden layer

A number of unsupervised methods can be employed to set the parameters of the hidden-layer basis functions. Usually the number of basis functions used is much greater than the number of clusters in the data in order to get adequate coverage of input space, but less than the number of training vectors available. It is possible to determine both the centres and the widths of the basis functions using a maximum likelihood algorithm, such as the Expectation Maximization (EM) algorithm [28], if the distribution of the input data is modelled by a Gaussian mixture model.

Alternatively, a random subset of the training data points can be chosen as initial centres, and then optimized using techniques such as K -means clustering [8, 68], Kohonen's self-organizing map [53], or the Generative Topographic Mapping (GTM) [9]. The widths of the basis functions are subsequently set using a heuristic that ensures a certain degree of overlap to give a relatively smooth representation of the training data. The Gaussians are restricted to be spherical, such that Σ_j in Equation 2.37 is replaced by σ_j^2 .

One simple heuristic is to set σ_j to be the Euclidean distance to the nearest neighbour kernel, or the average distance to the two nearest neighbours. An alternative approach is to use the following heuristic:

$$\sigma_j^2 = \frac{1}{n_j} \sum_{\mathbf{x} \in C_j} (\mathbf{x} - \boldsymbol{\mu}_j)^T (\mathbf{x} - \boldsymbol{\mu}_j) \quad (2.39)$$

where $\sum_{\mathbf{x} \in C_j}$ denotes a sum over those n_j training patterns $\mathbf{x}$ closest to basis function j , with centre $\boldsymbol{\mu}_j$.

Although the hidden-layer training procedure is essentially unsupervised, it is sometimes desirable to make use of the labels in the training data. It is important that each class is equally represented in the basis function space. The above methods will assign more basis functions to regions of high input data density; if a class is well spread or under-represented then it will not be adequately covered by hidden-layer basis functions, and as a result the

classification accuracy may suffer.

Therefore the following approach can be taken. Select a subset n_c patterns from each class (from the total for that class, N_c) at random as the basis function centres. Each basis function representing the same class is assigned the same width, σ_c , calculated as follows:

$$\sigma_c^2 = \frac{1}{N_c} \sum_{\mathbf{x} \in \text{class } c} (\mathbf{x} - \boldsymbol{\mu}_c)^T (\mathbf{x} - \boldsymbol{\mu}_c) \quad (2.40)$$

where $\boldsymbol{\mu}_c$ is the mean of *all* training vectors in class c . This is equivalent to the sum of the diagonal terms of the covariance matrix for vectors from class c .

2.3.2.2 The output layer

Now that the basis functions have been fixed, the values of the output layer weights can be determined. Each training vector is mapped onto the basis function space using Equation 2.37. The $\phi_j(\mathbf{x})$ values from the hidden layer are then used to train the output layer, which is just a linear version of the MLP output layer (*i.e.* no sigmoid function f_σ is used).

There are two approaches for minimizing the sum-of-squares error function given in Equation 2.22. The first is to follow a similar iterative procedure to that used for training the output layer of the MLP. Modifying Equation 2.28, the following expression is obtained:

$$\Delta w_{ij} = -\alpha \delta_i \phi_j \quad (2.41)$$

where

$$\delta_i = \frac{\partial E}{\partial a_i} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial a_i} = (y_i - t_i) \quad (2.42)$$

since $y_i = a_i$ for linear outputs. This is the update rule for training the output layer. Since it is a linear optimization problem, this iterative procedure will find the global minimum.

Although the above method is simple to implement, a non-iterative solution exists. Taking

all $N_{patterns}$ training patterns at once, Equation 2.38 can be re-written as:

$$\mathbf{Y} = \mathbf{W}\Phi \quad (2.43)$$

where $\mathbf{Y}$ is a $N_{outputs} \times N_{patterns}$ matrix of outputs, Φ is a $N_{hidden} \times N_{patterns}$ matrix of basis function outputs and $\mathbf{W}$ is the $N_{outputs} \times N_{hidden}$ output-layer weight matrix. The solution to $\mathbf{W}$ is wanted when $\mathbf{Y} = \mathbf{T}$, the matrix of target outputs. This gives:

$$\mathbf{W} = \mathbf{T}\Phi^\dagger \quad (2.44)$$

where $\Phi^\dagger$ is the pseudo-inverse of Φ (since Φ is not necessarily square) which can be found by singular value decomposition.

2.3.3 Further properties and comparison with MLPs

Both the MLP and RBF network input-output mappings may be viewed as variants of generalized linear discriminants [8], the definition of which has already been seen in the context of RBF functions in Equation 2.38:

$$y_i(\mathbf{x}) = \sum_j w_{ij} \phi_j(\mathbf{x}). \quad (2.45)$$

For the RBF network, the $\phi_j(\mathbf{x})$ values are the hidden-layer basis functions, while for the MLP they are the weighted sum functions of the hidden layer. The MLP also has a sigmoid function applied to its outputs.

Due to the different hidden-layer structure, different hidden-layer representations of input space are formed. For the MLP the hidden layer forms a distributed representation, with hyperplanes extending to infinity (see Figure 2.4). For the RBF network the hidden layer is composed of localized basis functions in space, giving rise to a localized representation of input space, as depicted in Figure 2.6.

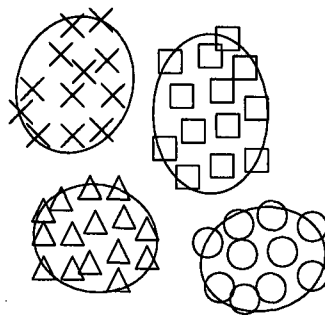


Figure 2.6: The equivalent localized RBF representation of that shown for the MLP in Figure 2.4.

Both architectures are similar in that they can approximate arbitrary non-linear functional mappings from an input vector $\mathbf{x}$ to an output vector $\mathbf{y}$. When an RBF network is used for classification, with the *one-of-m* encoding described in Section 2.2.3 for the MLP, it shares the same important property of MLPs that its outputs are estimates [8, 87] of the posterior probabilities $P(C_i | \mathbf{x})$. In addition, the property of universal approximation is also valid for RBF networks [80], so the architecture described in this section is sufficient in principle to approximate any arbitrary mapping.

Chapter 3

Features

Each pixel in a satellite image can be represented by an n -dimensional vector, where n is the number of spectral channels, each vector element being the raw spectral value from each channel. This is a primitive “feature vector” which requires no processing but which will provide only limited discriminatory information for determining the cloud class for that pixel [40]. Much more information can be obtained by considering the textural characteristics of the image data. For example, dynamic cloud such as stratus and cirrus tends to have a smoother texture than convective cloud such as cumulus and stratocumulus [4]. Furthermore, the latter two convective cloud types can be distinguished from each other by the fact that stratocumulus has larger convective cells than cumulus, and hence has different textural characteristics (see Section 1.2 for further information on the basic identification of cloud types).

Spectral information, when used in conjunction with textural information, remains important for determining cloud type [40]. For instance the two dynamic cloud types cirrus and stratus have similar textural characteristics and may therefore be difficult to distinguish using the texture alone. However, the former is a high-level cloud and the latter a low-level cloud and so the infra-red spectral value, which indicates the cloud-top temperature (and therefore height [16]) of the cloud top, is a very useful distinguishing feature.

A feature, therefore, is a value calculated (or simply obtained) from the image data at a

particular point which encapsulates some textural or spectral property of the image at that point. This chapter describes all the features considered, and the reasons for using them. Many of the features will be highly correlated, and it is necessary to select only the most discriminatory features for reasons explained in Section 3.2. Two feature selection methods are investigated in this chapter. The first uses the Bhattacharyya distance measure, and the second is a novel feature selection method based on the RBF networks presented in Chapter 2.

3.1 Description and overview of features

A feature vector is to be assigned to each pixel individually, but texture can only be defined for a region of an image. Therefore a 16×16 pixel region surrounding the pixel is used, and a set of statistical measures describing the spatial distribution of grey-levels in the region [41, 42] is calculated. The measures attempt to quantify such subjective notions as fineness, smoothness, granulation, randomness, lineation and mottledness. The term “spectral feature” will include simple statistics performed on all the grey-levels within the 16×16 pixel region, such as the mean, maximum, median, and standard deviation. For the Meteosat database described in Section 1.4.2, 17×17 pixel regions are provided; in order to be consistent with the Bankert database, these regions are reduced to 16×16 pixels.

Gu *et al* [40] provide a comprehensive analysis of 58 textural and spectral features and their applicability for distinguishing 13 cloud types. The best features for discriminating all the combinations of class pairs were investigated, as well as a ranking for the best overall features for separating the most difficult pairs. They came to the conclusion that while the textural features are highly important, they should only be considered as adding value to the spectral features which are of more value than *any* of the textural features. Also noted is the usefulness of textural features that respond to different scales in the image.

The results from studies concerned directly with classification are described in Section 5.1. The studies use and compare various textural, physical and spectral features that are appropriate for cloud classification. The most common textural features used are the Grey-Level

Table 3.1: Grey-level difference vector (GLDV) features; m is the absolute difference of grey-levels one pixel apart (0° or 90°), and $P(m)$ is the probability density function for m . (From Bankert [5]).

Mean difference	$\mu = \sum_m mP(m)$
Standard deviation	$\sigma = \left\{ \sum_m (m - \mu)^2 P(m) \right\}^{1/2}$
Angular second moment	$\sum_m [P(m)]^2$
Entropy	$-\sum_m P(m) \log P(m)$
Local homogeneity	$\sum_m \frac{P(m)}{1 + m^2}$
Contrast	$\sum_m m^2 P(m)$
Cluster shade	$\frac{\sum_m (m - \mu)^3 P(m)}{\sigma^3}$
Cluster prominence	$\frac{\sum_m (m - \mu)^4 P(m)}{\sigma^4} - 3.0$

Difference Vector (GLDV) measures [115] presented in Table 3.1, the Sum and Difference Histogram (SADH) measures [110] presented in Table 3.2 and the Run-Length (RL) statistics [34, 38] presented in Table 3.3. These textural measures were found to provide sufficient discriminatory information for good classification accuracy. In addition, the Grey-Level Co-occurrence Matrix (GLCM) features [81, 107] were found to provide very similar textural information to the GLDV and SADH descriptors but at a greater computational cost [112].

Hence the textural features used for this work are restricted to those presented in the tables. The measures can be calculated using either the horizontal or vertical pixel directions. For this thesis, the average of the features calculated in both directions is used. It will also be noticed that the local homogeneity and contrast features are identical for the GLDV and SADH features. As an illustration of one of the textural features, the GLDV homogeneity feature has been calculated for every pixel of an example Meteosat image, and is shown in Figure 3.1.

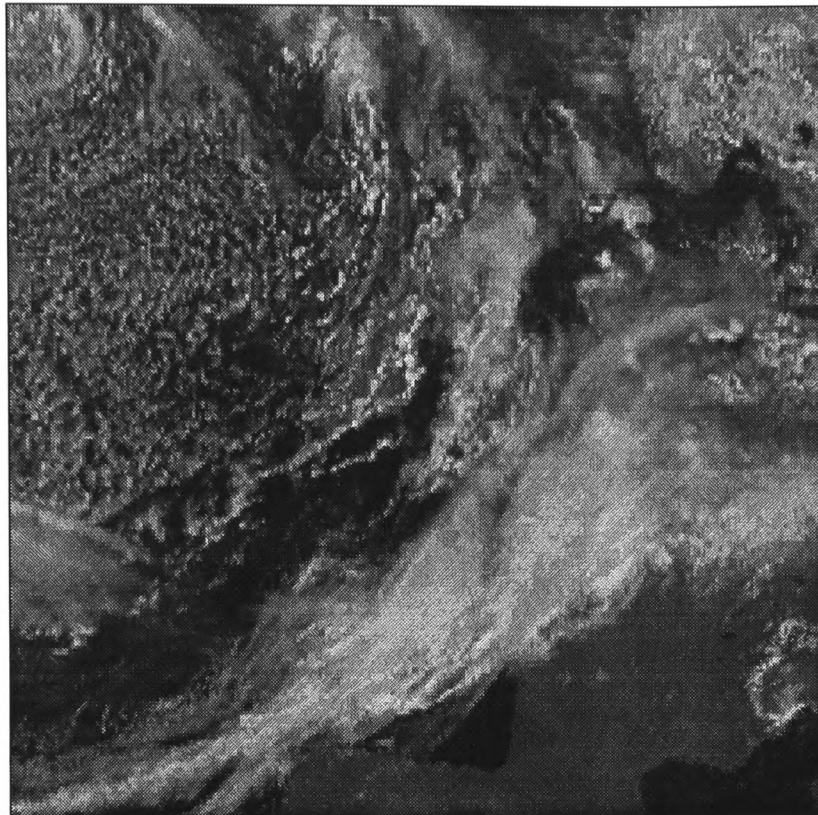
Welch *et al* [111] investigated how the GLDV features vary with resolution and pixel

Table 3.2: Sum and difference histogram (SADH) features; l is the difference of grey-levels one pixel apart (0° or 90°), k is the sum, $P(l)$ is the probability density function for l , and $P(k)$ is the probability density function for k . (From Bankert [5]).

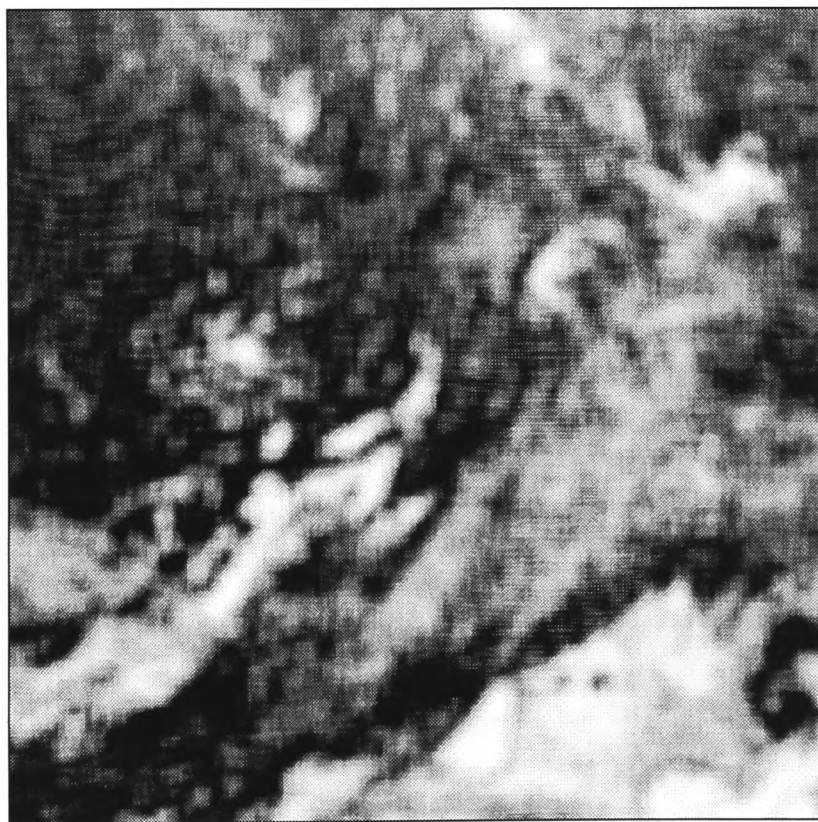
Mean sum	$\mu = \sum_k kP(k)$
Standard deviation	$\sigma = \left\{ \frac{1}{2} \left[\sum_k (k - \mu)^2 P(k) + \sum_l l^2 P(l) \right] \right\}^{1/2}$
Angular second moment	$\sum_k [P(k)]^2 \sum_l [P(l)]^2$
Entropy	$-\sum_k P(k) \log P(k) - \sum_l P(l) \log P(l)$
Local homogeneity	$\sum_l \frac{P(l)}{1 + l^2}$
Contrast	$\sum_l l^2 P(l)$
Cluster shade	$\frac{\sum_k (k - \mu)^3 P(k)}{\sigma^3}$
Cluster prominence	$\frac{\sum_k (k - \mu)^4 P(k)}{\sigma^4}$
Correlation	$\frac{\sum_k (k - \mu)^2 P(k) - \sum_l l^2 P(l)}{2\sigma^2}$

Table 3.3: Run-length (RL) features; i is the grey-level (0 to 255), j is the length of a run of one grey-level (1 to 16 for a 16×16 pixel sample area), $P(i, j)$ is the number of occurrences of runs of length j having grey-level i , $T_r = \sum_i \sum_j P(i, j)$, and T_p is the number of sample area pixels (256 for a 16×16 region). (From Bankert [5]).

Short run emphasis	$\frac{1}{T_r} \sum_i \sum_j \frac{P(i, j)}{j^2}$
Long run emphasis	$\frac{1}{T_r} \sum_i \sum_j j^2 P(i, j)$
Grey-level distribution	$\frac{1}{T_r} \sum_i \left\{ \sum_j P(i, j) \right\}^2$
Run-length distribution	$\frac{1}{T_r} \sum_j \left\{ \sum_i P(i, j) \right\}^2$
Run percentages	$\frac{1}{T_p} \sum_i \sum_j P(i, j)$



(a) Visible image



(b) GLDV homogeneity feature

Figure 3.1: (a) Meteosat visible image from 6 September 1994 at 1200 GMT, (b) GLDV homogeneity feature for the visible image, scaled from 0 to 255 and contrast enhanced for display.

separation, using Landsat images. Although Landsat has a spatial resolution of 57 m (see Appendix A for a comparison with AVHRR and Meteosat), it was found that a resolution of only $\frac{1}{4}$ km to $\frac{1}{2}$ km was needed for good classification accuracy.

Baraldi *et al* [7] considered the GLCM features [81, 107] and how they relate to each other, and which are appropriate in describing certain types of texture within a satellite image. Two parameters, energy and contrast, were considered to be the most efficient for discriminating between cloud textural patterns, while there was shown to be some correlation between the contrast and variance measures.

Morphological studies of individual cloud types have also been carried out, most notably by Sengupta *et al* [101] for cumulus cloud fields and Kuo *et al* [55] for cirrus cloud fields. These were in-depth investigations into the spatial, textural and structural properties of the cloud fields, including analysis of the variation of these properties with the cloud cell size and separation. GLCM features were used as texture descriptors, and for the latter study the contrast and correlation features were found to be the best discriminators of cloud field structure for cirriform clouds.

Not related specifically to satellite imagery, a recent study by Jain *et al* [46] considered a neural network model that acts as the texture feature extractor and selector itself, computing features from the pixels that are most appropriate for the classification, and then performing the classification. This has not been applied to satellite image interpretation.

For the spectral features, the mean, median, standard deviation, maximum, minimum and range of the 16×16 pixels are calculated, as well as the value of the central pixel within the 16×16 sample. All of the spectral and textural features are calculated for the visible and infra-red channels separately, and constitute a total of 54 features. Visible Fraction, calculated from the visible channel, is an estimate of the fraction of the sample covered in cloud [35], and is included as the last feature, making a total of 55 features. These are the same features as used by Bankert [5] and similar to those of the Welch *et al* study [113]. The

visible fraction feature was calculated by labelling a pixel as cloudy if the albedo is greater than 0.22, as justified by Garand [35].

3.2 Feature selection methods

Many of the features listed in Section 3.1 will be highly correlated, and this makes reducing the number of features desirable. There is, however, a further reason for wanting to reduce the number of features. Although a neural network trained in a supervised fashion acts as a feature selector in itself, adjusting its weights to ignore features with no discriminatory powers, there is a danger in having too many inputs. The use of more inputs than is strictly necessary produces a larger network and hence more free parameters. This will lead to the network fitting itself too well to the training data, learning peculiarities in the data which are not representative of the general population, leading to poor generalization capability on test data [8]. Of course, this is a problem only for finite size training sets; if the set were infinitely large, as many features as possible could be used.

This section describes two methods for selecting a subset of features. The first uses the Bhattacharyya distance measure, and the second is a novel feature selection method based on RBF networks. The Bhattacharyya distance measure for feature selection is common in the cloud classification literature [5, 113], while others include the divergence distance measure, the Fisher distance [33, 35] and principal components analysis [49]. A study by Bankert and Aha [6] considered both sequential forward and backward selection (explained later), with a nearest neighbour classifier used to assess performance. They noted an improved performance of the resulting feature set over the set chosen using the Bhattacharyya distance measure¹. Penaloza and Welch [85] used a fuzzy expert-system to determine the effectiveness of three approaches: the divergence, histogram analysis and discriminant analysis, concluding that the former provided the best set of features. A good general reference to feature selection

¹Using the same dataset and features as the Bankert and Aha study [6], no improvement in accuracy over the features chosen using the Bhattacharyya method was found.

methods can be found in Jain and Zongker [47], which contains application to Synthetic Aperture Radar satellite images.

3.2.1 Bhattacharyya distance measure

The Bhattacharyya distance measure [30, 113] is a measure of the discriminatory power of a set of features to separate two classes. It is defined as

$$J_B(i, j) = \frac{1}{8}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)^T \left[\frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2} \right]^{-1} (\boldsymbol{\mu}_j - \boldsymbol{\mu}_i) + \frac{1}{2} \ln \frac{\frac{1}{2} |(\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)|}{[|\boldsymbol{\Sigma}_i| |\boldsymbol{\Sigma}_j|]^{1/2}} \quad (3.1)$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are the means of the feature vectors for classes i and j , and $\boldsymbol{\Sigma}_i$ and $\boldsymbol{\Sigma}_j$ are the corresponding covariance matrices.

The first term provides a high contribution for those distributions whose means are well separated, and the second term provides a high contribution for those whose covariances are very different. For example, two Gaussian distributions that peak at the same mean, but which have different variances, will still have a finite separability due to the second term.

The Bhattacharyya distance is designed to provide a separability measure for *two* classes. However, for more than one class (as for the Bankert and Meteosat datasets), one may simply sum over all the possible pairs, introducing the prior probabilities of the classes involved as a weighting factor to produce the overall separability measure:

$$C_b = \sum_{i=1}^n \sum_{j=i+1}^n J_B(i, j) P_i P_j \quad (3.2)$$

where n is the number of classes. Note that the prior probability P_i is the probability that if a sample is selected at random from the *population*, then it will be of class i . It is not necessarily the same as the probability of class i in the dataset, for the distribution of classes in the dataset may be biased. For the case of the cloud classes used for this thesis, assume that each cloud type is equally likely to occur in nature [113], and hence the $P_i P_j$ term can

be dropped from the expression:

$$C_b = \sum_{i=1}^n \sum_{j=i+1}^n J_B(i, j). \quad (3.3)$$

The Bhattacharyya distance measure also requires that a particular set of features has already been chosen, in order to find the class separability using those features. A method is needed to choose feature sets in order to determine the best set using Equation 3.3. The Sequential Forward Selection (SFS) method [30, 113] is used for this purpose; it is a simple bottom-up procedure where one feature at a time is added to the current feature set. At each stage, a new feature is selected from the remaining features, so that the new enlarged feature set yields a maximum C_b .

3.2.2 Radial basis function method

A major drawback of the Sequential Forward Selection approach is that once added, a feature may not be removed, even if it is made redundant or irrelevant by the features added later on. In addition, certain features may be useful only in combination with others. By building up the vector one element at a time, these features may be overlooked.

An alternative is to consider the opposite of Sequential Forward Selection, where all the features are present at the start and features are sequentially removed until the number required are left [6]. In this way, all features are present with all others at the start, so that combinations of features can take effect. A feature is removed that contributes the least, in combination with any others, to the discriminatory powers of the feature vector as a whole.

In order to judge the level of discriminatory information provided by the feature vector at each stage, an RBF or MLP neural network as described in Chapter 2 can be used. The classification accuracy achieved is an indication of the usefulness of the set of input features. If a feature is removed, then the classification accuracy should decrease the most for features that provide a high level of discrimination. The process is summarized in the algorithm given in Figure 3.2. The RBF network is particularly suited for this feature selection method, while

1. Start with F -dimensional feature vectors, and identify each feature by f_n , where n is the index of the feature in the feature vector. Start n on the first element, $n = 1$.
2. Create $(F - 1)$ -dimensional feature vectors by removing one feature, f_n .
3. Train the RBF using these features and record the accuracy on the test set.
4. Increment n . If $n > F$ then goto 5, else goto 2.
5. The feature vector that gave the highest classification accuracy is identified. The missing feature for that feature vector is deemed to be the least useful and is removed permanently. Hence decrement F by one: $F \rightarrow F - 1$.
6. Set $n = 1$ and repeat from 2 until F is the dimension required.

Figure 3.2: The algorithm for the RBF feature selection method. All features are initially present, and the least useful ones are removed in turn until the required number are left.

the MLP is unsuitable. There are a number of reasons why this is the case.

Firstly, the MLP has a distributed representation, where for a given input vector many hidden units typically contribute to the output. The mapping is highly non-linear and there is much cross-coupling between the hidden units [8]. When the removed feature is changed from one to another, it is not certain whether a small change in accuracy is attributable to the feature change itself, or to a secondary effect which results in a different local minimum. The RBF output layer is linear, and the hidden layer forms a local, simpler representation of feature space, as explained in Chapter 2. Any change in accuracy is likely to be attributable to the feature change, whereas this is not the case for the highly non-linear MLP.

Secondly, for each feature removal it is desirable to set up the classifier from a similar, controlled starting point for training in order to minimize the number of possible reasons for a change in accuracy. This is impossible to do for the MLP, since the distributed influence of the weights is hard to control with a new set of inputs. With the RBF one can set the positions of the hidden-layer kernel functions on certain data points which remain the same for all training runs. The output layer is linear and the weights are determinable analytically from the data points and hidden-layer parameters.

Lastly, the effect of misleading features, as opposed to simply irrelevant features, is different for the two architectures. The MLP weights are adjusted to ignore the misleading information during training, so that the removal of those features will have little effect on the accuracy. However, a misleading input to an RBF network will affect the accuracy to a greater extent, for the following reason. The misleading input has the effect of spreading the data noisily along that feature axis, providing no useful information. However, the hidden-layer kernel functions have their positions and widths set according to this data density, and hence are more distributed than they should be, and this leads to a lower accuracy. Therefore misleading features are likely to be removed first by the RBF method.

Hence the MLP is unsuitable for the feature selection task and the RBF network will be used. The results are presented in Section 3.3.2.

3.3 Feature selection results

3.3.1 Bhattacharyya distance measure results

For some features, the Clear category in the Bankert database is sufficiently well separated from the other cloud classes that it causes problems for the feature selection procedure. For example, the run-length features are particularly good at separating Clear from all other classes. This is because a Clear sample tends to be very uniformly dark, giving rise to long run-lengths of the same pixel value. The cloud classes are much less uniform and it is rare for a run-length of more than one or two. Therefore some cloud classes may not be well separated from each other, but the individual cloud-clear distances are so large that the run-length features are ranked highly even though they may not be good *overall*. The same effect occurs for the visible fraction feature, the large mean-difference causing the first term in Equation 3.1 to dominate.

Another peculiarity occurs when the variance of a feature for a particular class is much smaller than that of all the other classes, so that the contribution from the second term of the Bhattacharyya distance in Equation 3.1 is large. Hence the contributions to the

overall separability C_b from pairs involving that class swamp all others, which may lead to a poor choice for an overall feature. An example of this occurs for the distance between the Nimbostratus and Cumulus classes for the infra-red long run emphasis feature.

Some illustrations of these effects can be given. Figure 3.3 shows the distribution of values for the infra-red spectral mean feature for the Cirrostratus and Cumulus types. Applying the Bhattacharyya method to this one feature, the contributions from the first (mean-difference) term is 0.78 and from the second (variance-difference) term is 0.32, a total of 1.10.

Figure 3.4 separately shows the distributions for the infra-red long run emphasis feature for classes Cirrocumulus and Stratocumulus. The first term contributes 0.05 and the second 2.08, a total of 2.13. This is almost double the value for the previous example, even though it is clear from the histograms that the previous classes are easier to separate. This stems from the fact that the second example has one class with a very small variance (Cirrocumulus with a variance of 2×10^{-6} compared with 0.04 for Stratocumulus), causing a very large contribution from the second term. This effect occurs most frequently for the Clear category, which often has a very narrow peak for some of the features.

These distributions should be compared with the distribution shown in Figure 3.5 for the infra-red spectral mean feature for the Nimbostratus and Cumulus types. The two classes are very well separated, with a Bhattacharyya distance of 10.15; the contribution from the second term is only 0.08.

In order to overcome the difficulties highlighted above, the method was modified as follows. Firstly, since the Clear category is sufficiently different from the other cloud categories, it can be left out of the feature selection process. Secondly, because some covariances are so small and distort the process, the *average* covariance matrix over all the classes will be used instead (as described in the study by Welch [113]) so that $\Sigma_i = \Sigma_j = \Sigma$. This removes the second term in Equation 3.1, leaving the following distance measure:

$$J_B(i, j) = \frac{1}{8}(\mu_j - \mu_i)^T \Sigma^{-1}(\mu_j - \mu_i). \quad (3.4)$$

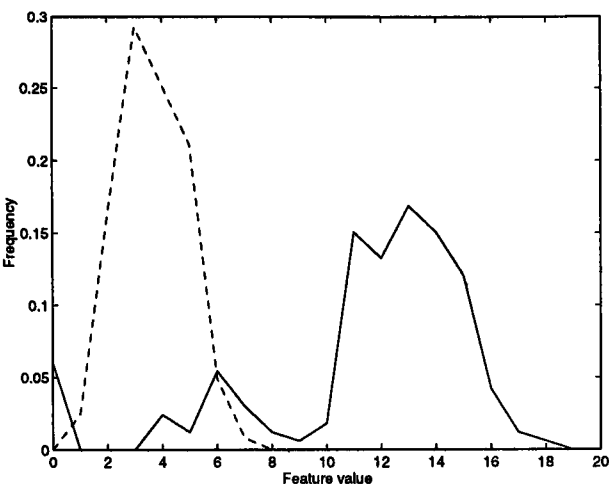


Figure 3.3: Distribution of the infra-red spectral mean feature for Cirrostratus and Cumulus (dashed).

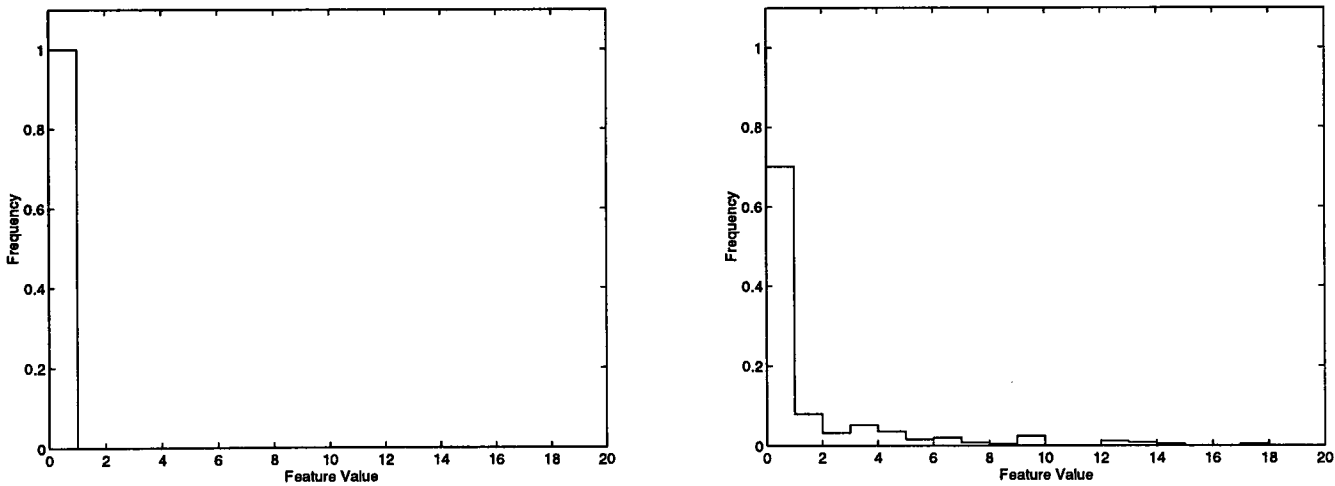


Figure 3.4: Distribution of the infra-red long run emphasis feature for Cirrocumulus (left) and Stratocumulus.

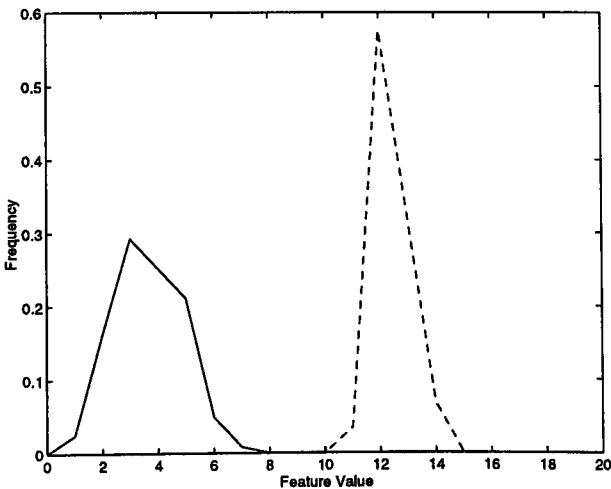


Figure 3.5: Distribution of the infra-red spectral mean feature for Nimbostratus (dashed) and Cumulus.

Table 3.4: Twenty features chosen by Bhattacharyya and SFS for the Meteosat database.

Rank	Channel	Feature
1	VIS	SADH Entropy
2	IR	Spectral Maximum
3	IR	Spectral Minimum
4	VIS	SADH Homogeneity
5	VIS	SADH Mean
6	VIS	GLDV Entropy
7	VIS	Run Percentages
8	VIS	SADH Cluster Shade
9	IR	Run-length Distribution
10	IR	GLDV Mean
11	IR	GLDV Entropy
12	VIS	GLDV Mean
13	IR	SADH Contrast
14	IR	Spectral Range
15	VIS	Grey-level Distribution
16	VIS	Spectral Centre
17	VIS	Visible Fraction
18	VIS	GLDV Angular 2nd Moment
19	IR	SADH Mean
20	VIS	SADH Standard Deviation

Table 3.5: Twenty features chosen by Bhattacharyya and SFS for the Bankert database.

Rank	Channel	Feature
1	IR	Spectral Maximum
2	VIS	GLDV Entropy
3	IR	SADH Homogeneity
4	VIS	GLDV Standard Deviation
5	IR	SADH Correlation
6	IR	GLDV Entropy
7	IR	Spectral Minimum
8	VIS	Spectral Maximum
9	VIS	Visible Fraction
10	VIS	SADH Contrast
11	IR	Spectral Median
12	VIS	GLDV Cluster Shade
13	IR	GLDV Angular 2nd Moment
14	IR	Spectral Range
15	IR	Run Percentages
16	IR	GLDV Cluster Shade
17	IR	SADH Angular 2nd Moment
18	IR	Spectral Standard Deviation
19	IR	GLDV Mean
20	VIS	GLDV Angular 2nd Moment

In fact, this is now equivalent to the Mahalanobis distance measure, with a scaling factor of $1/8$. Using this procedure with all 55 features and all the samples in both the Bankert and Meteosat databases, the first twenty features chosen are shown in Tables 3.4 and 3.5 respectively. Discussion of the results will be deferred until Section 3.4 where comparison will be made with the results from the RBF feature selection method presented in the next section.

3.3.2 Radial basis function method results

The RBF feature selection method starts with all features present as inputs. This would necessitate using a 55-dimensional input vector, which might cause problems due to the curse of dimensionality [8]. It is therefore desirable to reduce manually the number of features before using the algorithm. It is apparent that the GLDV and SADH statistics capture very similar textural information, and indeed two of the measures (local homogeneity and contrast) are identical. The initial features were therefore restricted to include *either* the GLDV set *or* the SADH set. The SADH feature set includes one feature not named in the GLDV set, correlation, and so this was added to the GLDV set for those trials.

Many of the spectral features are also very similar to each other. For instance, the minimum, maximum, mean, median and range are all related. Therefore only the median and range features were included in the initial set, since these provide sufficient information. The standard deviation feature was shown to be unimportant by the Bhattacharyya tests.

As a result of this, a reduced set of 33 initial features was assembled. An RBF network with 200 hidden-layer kernels was used with one output for each class. The per-class heuristic described in Section 2.3.2.1 was used to set the hidden-layer kernel positions and widths. As explained in Section 3.2.2, it is desirable to set up the hidden-layer kernels as consistently as possible for each run in the feature deletion process. This was achieved by applying the scheme described below for the Meteosat dataset; the same method was also applied to the Bankert dataset.

Since the division between classes of the full dataset is unequal (see Section 1.4.2), a balanced dataset was first created, with 120 samples selected at random from each class, *i.e.* a total of 600 samples in this case. The balanced dataset was then divided into training and test sets in the ratio 3:1. This procedure was performed three times, using three different random allocations of the data points, to create three different partitions of the data into training and test sets. Subsequently, the data within each of these sets were given five different random orderings, creating a total of fifteen training and testing set pairs. The ordering of points within these sets remained fixed throughout the remainder of the procedure for reasons explained below.

For each training run, using the per-class heuristic, the 200 RBF centres were allocated equally to the five output classes *i.e.* 40 per class. The positions of the kernels for each class were set to the first 40 points in the training set belonging to that class. Therefore the kernels were fixed on the same point in feature space (with varying features and dimensions) for every training run, in accordance with the need described in Section 3.2.2 for consistent kernel positions. The widths of the kernels were then set using Equation 2.40. For every feature deletion, training and testing was carried out using all fifteen dataset partitions, so that an average classification accuracy could be computed. The feature removal process was continued until only two features remained in the feature vector. These last two features are therefore of equal rank.

The results are shown in Figures 3.6 and 3.7 for the Meteosat and Bankert datasets respectively. They are plots of the average classification accuracy achieved on the test sets by the RBF, versus the number of features removed. The (a) and (b) plots show the results for the SADH and GLDV features respectively, using all the output classes and balanced datasets. The accuracy can be seen to rise slowly as features are initially removed, which could be due to two effects. Firstly, features that contain little or no information introduce noise in the input space. As these are removed, the accuracy of an RBF classifier will increase. Secondly, the fact that the number of basis functions was fixed at 200 throughout will have

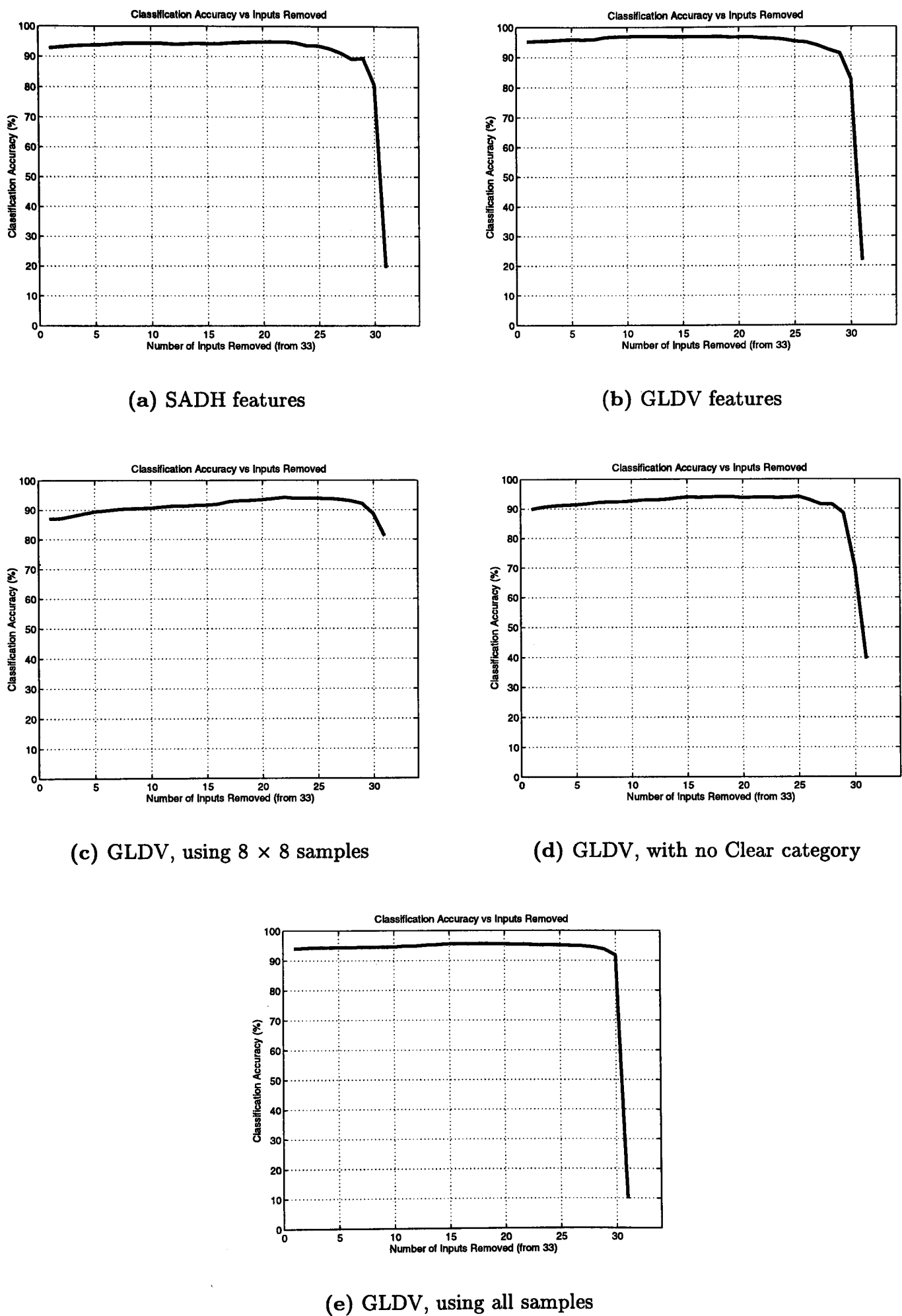
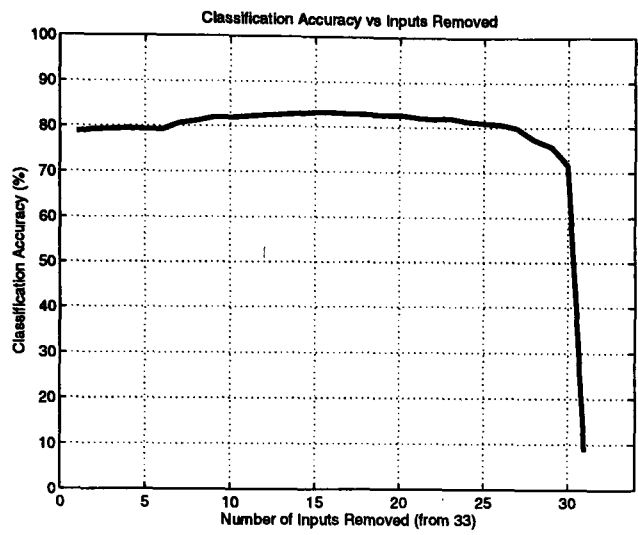
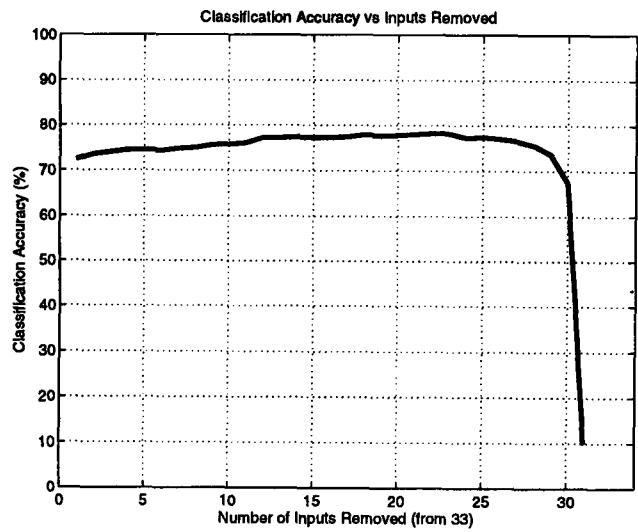


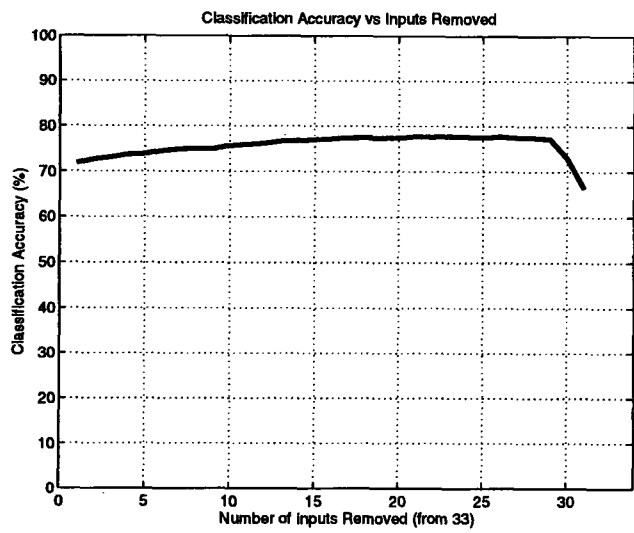
Figure 3.6: Meteosat database. Average classification accuracy of an RBF network with 200 kernel centres versus the number of inputs removed (from a total of 33).



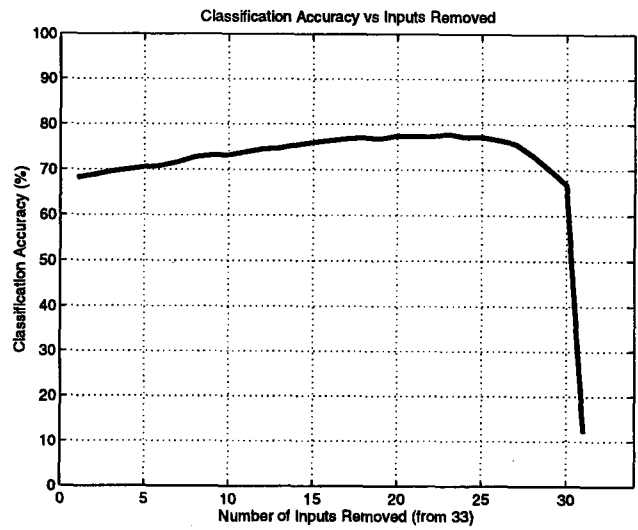
(a) SADH features



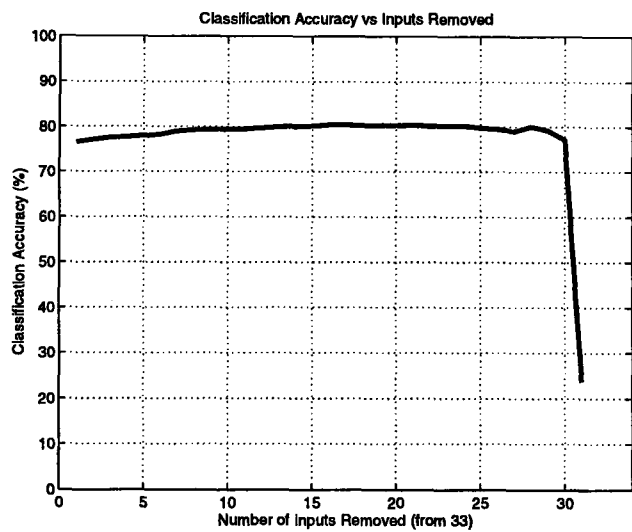
(b) GLDV features



(c) GLDV, using 8×8 samples



(d) GLDV, with no Clear category



(e) GLDV, using all samples

Figure 3.7: Bankert database. Average classification accuracy of an RBF network with 200 kernel centres versus the number of inputs removed (from a total of 33).

Table 3.6: Twenty features chosen by the RBF method for the Meteosat database and the GLDV features. The first two features are of equal ranking.

Rank	Channel	Feature
1=	IR	Spectral Median
1=	VIS	Spectral Median
3	VIS	Run-length Distribution
4	IR	GLDV Contrast
5	VIS	Spectral Range
6	IR	Run-length Distribution
7	VIS	SADH Correlation
8	VIS	GLDV Cluster Shade
9	IR	Spectral Range
10	IR	SADH Correlation
11	VIS	GLDV Mean
12	VIS	Long Run Emphasis
13	IR	GLDV Standard Deviation
14	IR	Run Percentages
15	VIS	GLDV Prominence
16	VIS	GLDV Standard Deviation
17	VIS	GLDV Homogeneity
18	VIS	Visible Fraction
19	IR	GLDV Entropy
20	VIS	GLDV Entropy

Table 3.7: Twenty features chosen by the RBF method for the Bankert database and the SADH features. The first two features are of equal ranking.

Rank	Channel	Feature
1=	IR	Spectral Median
1=	VIS	SADH Entropy
3	VIS	Spectral Median
4	IR	Run-length Distribution
5	VIS	SADH Contrast
6	IR	Spectral Range
7	VIS	SADH Correlation
8	VIS	SADH Cluster Shade
9	IR	SADH Mean
10	VIS	Visible Fraction
11	VIS	Spectral Range
12	IR	SADH Entropy
13	VIS	SADH Prominence
14	VIS	SADH Homogeneity
15	IR	SADH Correlation
16	IR	SADH Standard Deviation
17	IR	Short Run Emphasis
18	VIS	SADH Angular 2nd Moment
19	IR	SADH Contrast
20	IR	SADH Angular 2nd Moment

an effect. There is expected to be an optimum ratio of the number of inputs to the number of basis functions, and this is reflected in the shape of the plot. After approximately 20 features have been removed, the accuracy is dominated by the information content of the input features, which is seen to fall off dramatically when there are eight, or fewer, features left.

Other variations were also tried. The (c) plots were obtained using 8×8 pixel sample boxes instead of 16×16 , showing marginally lower accuracy. The benefit of using 8×8 sample boxes is the time saved when calculating the features. Using no Clear category (as was done for the Bhattacharyya results), produces the results shown in the (d) plots. Using all samples in the datasets (*i.e.* unbalanced datasets), results in the (e) plots which are similar to the others but with a steeper fall-off.

For the Meteosat dataset, the best results were achieved using the GLDV features, shown in Figure 3.6(b), reaching a maximum accuracy of 96.9% with 18 features. For the Bankert dataset, the best results were achieved using the SADH features, shown in Figure 3.7(a), reaching a maximum accuracy of 83.2% with 17 features. However, the curves are flat in these regions so using more or fewer features will not have a great effect on the accuracy. The feature rankings for these runs are shown in Tables 3.6 and 3.7. These rankings will be used for the selection of the necessary number of features for all further work presented in this thesis.

3.4 Conclusions

A comprehensive treatment of the feature extraction and selection process has been presented in this chapter, and related to other research in the field. A new feature selection method using an RBF network was used to rank the best twenty features for each image database.

The RBF network has been shown to be well suited to the task of feature selection, due to the characteristics described in Section 3.2.2. A good mix of spectral and textural features was selected by the RBF method; in particular it should be noted that the spectral median

features for both channels were ranked highly for both datasets, since it is known that these are important features [40]. For the Meteosat database the spectral median features were the two left at the end, while for the Bankert database the visible spectral median feature was the last to be removed.

The smoothness of the plots shown in Figures 3.6 and 3.7 are evidence of the properties of the RBF (set out in Section 3.2.2) that make it suited to feature selection. The removal of each feature vector in turn resulted in a small change in the accuracy in a consistent direction, producing reasonably smooth curves. This is unlike the behaviour of the MLP where the removal of a feature is less likely to affect the accuracy in a controlled manner. In addition, the RBF method does not suffer from the deficiencies of the Bhattacharyya method described in Section 3.3.1.

It is difficult to compare directly the results of the Bhattacharyya method and the RBF method since some of the features chosen by the former method were not used in the initial feature set of the latter method. It is also difficult to compare the results of the two databases for the RBF method for the same reason. However, it is apparent that the order of the features chosen by each method varies considerably, which implies that the features provide reasonably similar discriminatory information. The two spectral median features stand out as being the most important, but only when in combination with the other textural features as demonstrated by the severe fall in accuracy when few features are left.

The absolute accuracy values for the RBF network shown in this chapter should not be compared with those in Chapter 5. For these feature selection results, there is to some extent a bias towards the particular datasets used for the training runs, and the samples used to initialize the hidden-layer kernel functions. Features were removed that would attempt to maximize the accuracy for those datasets, and hence the accuracy is likely to be slightly higher than for trials using the same features on different datasets.

Chapter 4

Visualization techniques

It is important for the design of any cloud classification system to gain a full understanding of the dataset being used for the following reasons:

- To inform the choice of a suitable classifier.
- To help explain errors of the classifier in terms of class overlap in certain regions of feature space.
- To identify anomalies in the dataset.
- To determine whether a more appropriate labelling of cloud type is possible by grouping cloud types that are close in feature space [1].

It is therefore useful to have a visualization method to detect and identify structure in the multi-dimensional feature space occupied by the feature vectors calculated from the cloud samples (see Chapter 3). Humans are only able easily to visualize one-, two- or three-dimensional structures. Hence what is needed is a dimensionality reduction method to project the multi-dimensional space onto a two- or three-dimensional space, whilst preserving as much of the structure that is present in the multi-dimensional space as is possible. This enables one to see how various types of cloud are related to each other, how they are ordered in feature space, and whether any significant clusters occur.

Three such methods are investigated: the Kohonen feature map, the Sammon mapping and the Generative Topographic Mapping (GTM). All the methods use *unsupervised* training

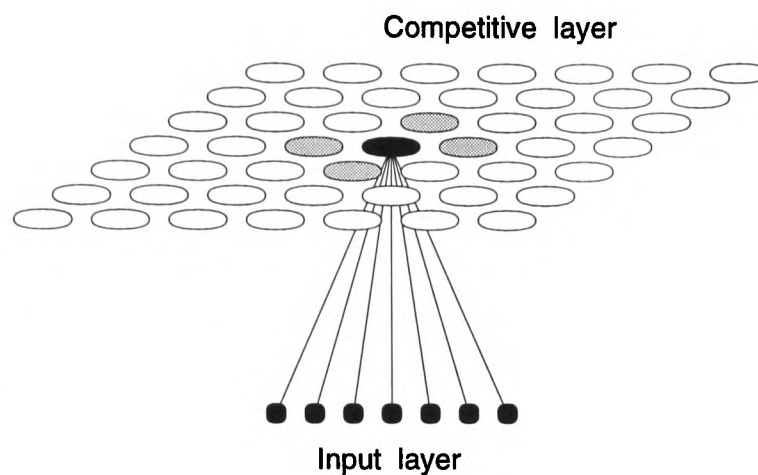


Figure 4.1: The Kohonen architecture. Shown is a 7×7 square grid of units with a seven-dimensional input vector. The unit shaded black is the activated unit and the units shaded grey are its nearest neighbours.

procedures, which do not require any class label attached to the feature vector for the purposes of training. The work in this thesis is concerned mainly with *two*-dimensional maps, so where the original theory may allow for higher dimensions, details are restricted to two dimensions. The theory is trivially extended to three dimensions, which will be used in Chapter 6 for the production of “unsupervised classification” maps. These are images with pixels coloured according to cloud type, but for which no cloud class labels are used at any stage of the process.

4.1 The Kohonen feature map

The Kohonen feature map [53] may be considered as a two-layered neural network. The first layer is the input layer, and the second is a competitive layer with units arranged in a two-dimensional lattice structure, either as a rectangular grid of units, as shown in Figure 4.1, or in a hexagonal structure. Topological ordering in the second layer results in similar input patterns (*i.e.* close in feature space) activating units in the second layer layer that are spatially close to each other. The network is trained in an *unsupervised* fashion.

The training algorithm is a modified adaptive K -means procedure [8] (see Section 4.2.1.1). Each unit in the competitive layer has a weight vector of the same dimension as the input layer size. During training, topological ordering occurs which causes the competitive-layer weight vectors from nearby grid points to become close to each other in vector space. This is

1. Select an initial random set of weight vectors, usually chosen such that each element in the vector has a small value, for example in the range ± 0.1 .
2. Select a training pattern $\mathbf{x}$, and compute the Euclidean distance between $\mathbf{x}$ and each weight vector $\mathbf{w}_{ij}$ (i.e. $|\mathbf{x} - \mathbf{w}_{ij}|$). The unit with the smallest distance is the *winning unit*, denoted by u_* .
3. Adjust *all* units within a neighbourhood $\mathcal{N}$ of units surrounding u_* according to the following formula:

$$\Delta \mathbf{w}_{ij} = \alpha (\mathbf{x} - \mathbf{w}_{ij}) \quad \{ij\} \in \mathcal{N}$$

where α is the learning rate.

4. Reduce the neighbourhood size and learning rate by a pre-determined amount. Return to 2 unless these two parameters have reached their final values.

Figure 4.2: The Kohonen map training algorithm. Each unit in the competitive layer, referred to as u_{ij} where i and j represent the coordinates of the unit on the two-dimensional grid, has a weight vector $\mathbf{w}_{ij}$ of the same dimension as the input layer size. Input vectors are denoted by $\mathbf{x}$.

achieved through the Kohonen training algorithm shown in Figure 4.2. The neighbourhood $\mathcal{N}$ referred to in Step 3 is a set of units surrounding the winning unit, of arbitrary shape but usually approximately circular. Both α and the size of $\mathcal{N}$ are reduced slowly as training progresses; the exact functional form for this is not important. At the start of training the neighbourhood should be large (containing most of the units) and the learning rate high (around 0.8), both decreasing to small values by the end of training. The training set is usually presented more than once (see below).

4.1.1 Implementation details

The competitive layer used in the *NetSat* system is a square lattice of units and the neighbourhood region $\mathcal{N}$ is radially symmetric. All units within a certain radius of the winning unit (defined in terms of the distance between two adjacent lattice points) are included in the neighbourhood. For example, Figure 4.1 shows shaded all those units within a radius of one from the winning unit (in black). A radius of zero is just the winning unit itself.

Both α and the size of $\mathcal{N}$ are reduced after every pattern presentation, the decrease of

both being *linear* with respect to the number of presentations, with a rate of decrease such that α reaches zero after the required number of presentations. For the Bankert database there is only a maximum of 1592 training samples, and Kohonen [53] suggests that the total number of presentations should be at least 500 times the number of units in the competitive layer. For a typical 10×10 grid, this means the entire training set has to be presented approximately 30 times to satisfy this guideline.

It was found that a final radius of one resulted in slightly better-defined boundaries between cloud classes, although the results were not highly sensitive to the choice. Training at the end with no neighbourhood is equivalent to the adaptive K -means algorithm where each unit is altered with no influence on its neighbours, possibly destroying some of the topological ordering.

4.2 The Sammon mapping

The Sammon mapping [97] is an algorithm to map N points from a high-dimensional space, say D , to a lower-dimensional space, typically with two or three dimensions, whilst preserving approximately the inherent structure of the data. This is achieved by fitting the N points into the two-dimensional space such that the inter-point distances d_{ij} approximate the inter-point distances in the original D -dimensional space δ_{ij} . A gradient-descent approach is employed, using the following error criterion:

$$E^2 = \frac{1}{\sum_{i < j} \delta_{ij}} \sum_{i < j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}. \quad (4.1)$$

If element k of the i -th vector in two-dimensional space is denoted by y_{ik} , then the following update rule is employed after each presentation of a vector from D -space:

$$y_{ik}(m+1) = y_{ik}(m) - (MF) \cdot \Delta_{ik}(m) \quad (4.2)$$

where m is the iteration number and

$$\Delta_{ik}(m) = \frac{\partial E(m)}{\partial y_{ik}(m)} \bigg/ \left| \frac{\partial^2 E(m)}{\partial y_{ik}(m)^2} \right| \quad (4.3)$$

and MF is a learning rate factor which was empirically determined [97] to be about 0.3. Further details can be found in Sammon's original paper [97].

4.2.1 Implementation details

It is not realistic to use all the available input vectors to perform the Sammon mapping for computational reasons [97]. A smaller set of prototype patterns are needed to represent the training data. It would be possible to use the weight vectors from the competitive layer of the trained Kohonen network. However this would not be ideal because Kohonen's algorithm is essentially an adaptive K -means clustering procedure (see below) with an additional constraint to cause topological ordering on a fixed two-dimensional grid. This additional spatial constraint, when selecting prototype vectors for the Sammon algorithm, is counter-productive because the Sammon algorithm is itself trying to produce a two-dimensional representation.

Instead the non-stochastic *batch* version of the K -means algorithm, described in the next section, is used to choose the prototype vectors to be mapped onto the two-dimensional visualization space.

4.2.1.1 Batch K -means algorithm

The K -means algorithm [8] is an example of an iterative clustering algorithm which aims to partition a set of vectors into K clusters, where K is chosen by the user. For a given value of K , it aims to minimize the error function

$$E_K^2 = \sum_{k=1}^K \sum_{\mathbf{x} \in C_k} |\mathbf{x} - \mathbf{m}_k|^2 \quad (4.4)$$

1. Select a set of K vectors $\{\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_K\}$ to form the initial cluster centres, or prototype vectors.
2. Assign each of the patterns $\mathbf{x}$ to its closest cluster centre.

i.e. $\mathbf{x}$ belongs to cluster C_i if $|\mathbf{x} - \mathbf{m}_i| < |\mathbf{x} - \mathbf{m}_j|$ for all $i \neq j$.
3. Compute new cluster centres as the centroids of the K clusters. This minimizes the cost function E_K^2 defined in Equation 4.4.
4. If the position of any cluster centres changes, return to Step 2; otherwise, stop.

Figure 4.3: The batch K -means algorithm. The initial set of centres chosen in Step 1 may simply be K patterns chosen at random from the patterns in the dataset. Different choices lead to different final clusterings corresponding to different local minima.

where $\mathbf{m}_k$ is the k -th cluster centre, or prototype vector, and $\mathbf{x}$ is a vector belonging to cluster C_k . The algorithm proceeds as shown in Figure 4.3.

This is the *batch* version of the K -means algorithm. The *adaptive* version has already been seen in the context of the Kohonen network, where vectors are presented one at a time and the cluster centres are adjusted after *every* presentation using the update rule $\Delta \mathbf{m}_i = \alpha(\mathbf{x} - \mathbf{m}_i)$, where α is the learning rate parameter and $\mathbf{m}_i$ is the cluster centre nearest to the input vector $\mathbf{x}$.

4.2.2 Initialization of the Sammon vectors

It was found that the choice of initial points on the two-dimensional surface was important; some starting configurations led to a high final error, while others quickly led to a good solution. A purely random initialization led to poor performance. The original paper by Sammon [97] suggests a crude form of principal components analysis [8], where the initial configuration is found by using the two coordinates of the D -dimensional data having the largest variance. This was found to give good convergence most of the time, but occasionally one of the chosen coordinates had a high variance, but an unusual distribution, which made it a bad choice. Such a case occurred when the coordinate had values that were repeated

several times in the original data.

Instead, all possible pairs of coordinates of the D -dimensional vectors were used in turn to form the initial two-dimensional data points with which to begin the procedure, and the pair giving the lowest initial squared error E^2 (as in Equation 4.1) was chosen to continue the process. This was found to give consistently good convergence to a low error. In addition, 10–20% noise was added to the elements of the initial two-dimensional vectors to add some variation; otherwise the same data would always lead to the same final Sammon map.

The training procedure was stopped according to the following criteria:

- Passes through the training set must have reached at least 150.
- *Either* the error E is less than 0.04 *or* the change in error from the previous pass is less than 0.00001 (error reducing).
- The maximum number of passes is set at 300.

These rules were obtained empirically for this work and will not necessarily be applicable to other datasets. Precautions were taken to prevent any two points in the two-dimensional space from becoming identical, for then the partial derivatives in Equation 4.3 would be undefined. This was done by adding 10% noise to both elements of both points. Occasionally two points would become very close, causing the error to rise sharply, but it would soon relax down to a more stable value.

4.3 The Generative Topographic Mapping

Unlike the Sammon mapping, the Kohonen map does not have a cost function to minimize, but this is addressed in a new algorithm called the Generative Topographic Mapping (GTM). The derivation of Bishop *et al* [9, 10] is followed here.

Consider again a D -dimensional data space $\mathbf{x}$, and an L -dimensional latent space $\mathbf{y}$ in which the data is to be visualized (hence L is usually two or three). A mapping $\mathbf{g}(\mathbf{y}; \mathbf{W})$ is defined *from* latent space points $\mathbf{y}$ *to* corresponding points $\mathbf{g}(\mathbf{y}; \mathbf{W})$ in data space, where

$\mathbf{W}$ is the matrix of parameters governing the transform. If a probability distribution $p(\mathbf{y})$ is defined in the latent space, a corresponding distribution $p(\mathbf{g} \mid \mathbf{W})$ can be generated in data space. Since $L < D$, $p(\mathbf{g} \mid \mathbf{W})$ would be confined to a lower-dimensional manifold embedded in the higher-dimensional space, and hence would be singular. It is appropriate, therefore, to introduce a noise model in the data space, so that a single point $\mathbf{y}$ in latent space is mapped to a spherical Gaussian (the noise model) in data space centred on $\mathbf{g}(\mathbf{y}; \mathbf{W})$ and having variance β^{-1} . Hence the probability distribution $p(\mathbf{x})$ of points in data space is dependent on the distribution $p(\mathbf{y})$ via the following conditional relationship:

$$p(\mathbf{x} \mid \mathbf{y}, \mathbf{W}, \beta) = \left(\frac{\beta}{2\pi} \right)^{\frac{D}{2}} \exp \left\{ -\frac{\beta}{2} |\mathbf{x} - \mathbf{g}(\mathbf{y}; \mathbf{W})|^2 \right\}. \quad (4.5)$$

It is now possible to see more clearly how the mapping is used for visualization purposes. When the mapping parameters $\mathbf{W}$ and β have been optimized for the current dataset, as described later, Equation 4.5 can be inverted using Bayes' theorem to get $p(\mathbf{y} \mid \mathbf{x}, \mathbf{W}, \beta)$ (the posterior probabilities). This is a probability distribution in the low-dimensional latent space for a corresponding data point $\mathbf{x}$ in high-dimensional space; each point in data space is visualized as a probability distribution in the lower-dimensional latent space.

The distribution in data space is given by integration over the latent distribution:

$$p(\mathbf{x} \mid \mathbf{W}, \beta) = \int p(\mathbf{x} \mid \mathbf{y}, \mathbf{W}, \beta) p(\mathbf{y}) d\mathbf{y}. \quad (4.6)$$

It remains to define the probability distribution $p(\mathbf{y})$ in the latent space. To make the integral in Equation 4.6 tractable, reference to the Kohonen model is made by considering $p(\mathbf{y})$ to be composed of a set of K delta functions arranged on a uniform grid as with the Kohonen map:

$$p(\mathbf{y}) = \frac{1}{K} \sum_{i=1}^K \delta(\mathbf{y} - \mathbf{y}^i) \quad (4.7)$$

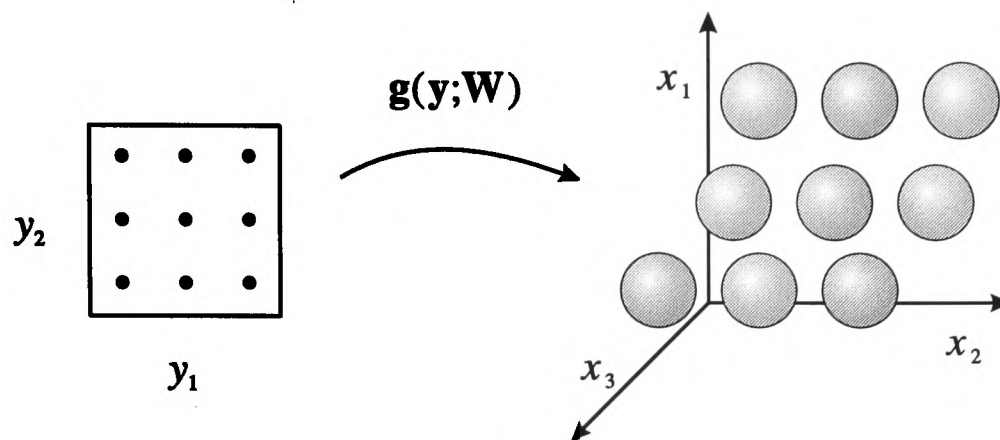


Figure 4.4: The relationship between latent space (left) and data space (right) in the GTM. The distribution $p(\mathbf{x} \mid \mathbf{W}, \beta)$ from Equation 4.8 is shown, which uses the mapping defined in Equation 4.5. A grid of delta functions in the latent space is mapped via $\mathbf{g}(\mathbf{y}; \mathbf{W})$ to a set of spherical Gaussians on a manifold in data space. (From Bishop *et al* [9]).

so that Equation 4.6 becomes:

$$p(\mathbf{x} \mid \mathbf{W}, \beta) = \frac{1}{K} \sum_{i=1}^K p(\mathbf{x} \mid \mathbf{y}^i, \mathbf{W}, \beta). \quad (4.8)$$

This distribution is illustrated in Figure 4.4. Note that $p(\mathbf{y})$ is referred to as the *prior* distribution because it represents the probability distribution in latent space with no *a priori* knowledge of the data points; points may lie on any of the grid points with equal probability. For a known data point $\mathbf{x}$, this becomes the *posterior* probability $p(\mathbf{y} \mid \mathbf{x}, \mathbf{W}, \beta)$, which is again a grid of delta functions, but with the prior probability $1/K$ of Equation 4.7 replaced by:

$$\frac{p(\mathbf{x} \mid \mathbf{y}^i, \mathbf{W}, \beta)}{\sum_{i'=1}^K p(\mathbf{x} \mid \mathbf{y}^{i'}, \mathbf{W}, \beta)}. \quad (4.9)$$

This provides an entire distribution in latent space for *every* data point. Examples are shown in Figure 4.5 for two different data points $\mathbf{x}$. The radius of the circle on each grid point is scaled in proportion to the magnitude of the posterior probability on that grid point. If a whole set of points is to be visualized at once, then the position of the *mean* (or centre of gravity) of the posterior distribution for each data point may be plotted, which will usually not lie on any grid point.

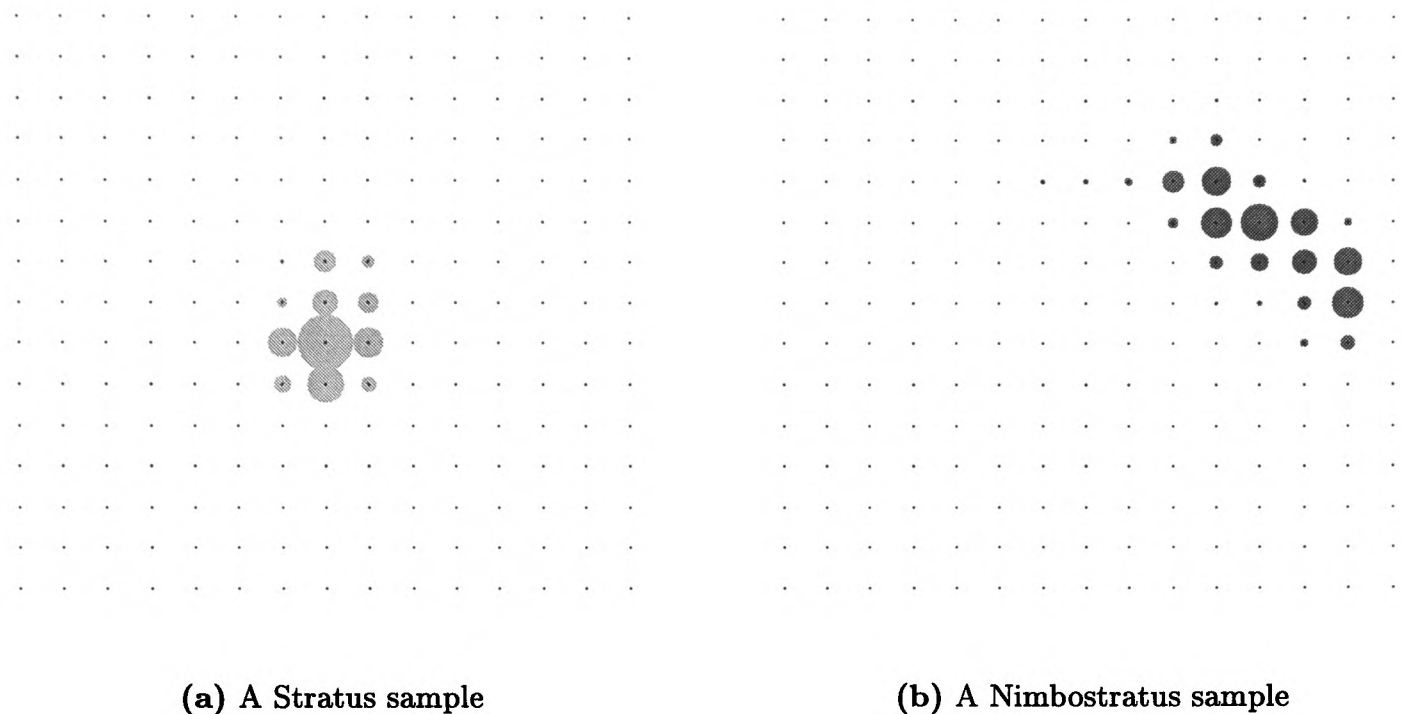


Figure 4.5: Two example GTM posterior distributions $p(\mathbf{y} \mid \mathbf{x}, \mathbf{W}, \beta)$ for two different data points $\mathbf{x}$. The results are from a training run described in Section 4.5.

The model used for the mapping $\mathbf{g}(\mathbf{y}; \mathbf{W})$ is a generalized linear network, with M basis functions $\phi_j(\mathbf{y})$ (fixed throughout the training procedure):

$$g_k(\mathbf{y}; \mathbf{W}) = \sum_{j=1}^M w_{kj} \phi_j(\mathbf{y}) \quad (4.10)$$

or

$$\mathbf{g}(\mathbf{y}; \mathbf{W}) = \mathbf{W} \boldsymbol{\phi}(\mathbf{y}) \quad (4.11)$$

where $\mathbf{W}$ is a $D \times M$ matrix and $\boldsymbol{\phi}(\mathbf{y})$ is a vector of basis function activations of dimension M . The basis functions are chosen to be radially-symmetric Gaussians and lie on a grid in latent space, with positions unrelated to those of the K delta functions in latent space. The widths of these fixed basis functions are set at the start, and determine the smoothness of the final GTM [10].

Until now, it has been assumed that the parameters governing the mapping, $\mathbf{W}$ and β , were known. These are determined by computing the likelihood of Equation 4.8 over the

entire dataset, and maximizing this likelihood using the Expectation Maximization (EM) algorithm [28]. For N data points $\mathbf{x}^n$, define the following objective function which is to be minimized:

$$\mathcal{L}(\mathbf{W}, \beta) = \ln \prod_{n=1}^N p(\mathbf{x}^n | \mathbf{W}, \beta). \quad (4.12)$$

Inserting the expression for $p(\mathbf{x}^n | \mathbf{W}, \beta)$ from Equation 4.8, this becomes

$$\mathcal{L}(\mathbf{W}, \beta) = \sum_{n=1}^N \ln \left\{ \frac{1}{K} \sum_{i=1}^K p(\mathbf{x}^n | \mathbf{y}^i, \mathbf{W}, \beta) \right\}. \quad (4.13)$$

Recalling Equations 4.5 and 4.11, and maximizing Equation 4.13 with respect to $\mathbf{W}$, the following is obtained [9]:

$$\sum_{n=1}^N \sum_{i=1}^K R_{in}(\mathbf{W}, \beta) \{ \mathbf{x}^n - \mathbf{W} \phi(\mathbf{y}^i) \} \phi^T(\mathbf{y}^i) = 0 \quad (4.14)$$

where Bayes' theorem has been used to introduce the posterior probabilities, or *responsibilities*, for the mixture components given, as they were in Equation 4.9, by

$$\begin{aligned} R_{in}(\mathbf{W}, \beta) &= p(\mathbf{y}^i | \mathbf{x}^n, \mathbf{W}, \beta) \\ &= \frac{p(\mathbf{x}^n | \mathbf{y}^i, \mathbf{W}, \beta)}{\sum_{i'=1}^K p(\mathbf{x}^n | \mathbf{y}^{i'}, \mathbf{W}, \beta)}. \end{aligned} \quad (4.15)$$

Similarly, maximizing with respect to β ,

$$\frac{1}{\beta} = \frac{1}{ND} \sum_{n=1}^N \sum_{i=1}^K R_{in}(\mathbf{W}, \beta) |\mathbf{x}^n - \mathbf{W} \phi(\mathbf{y}^i)|^2. \quad (4.16)$$

An iterative scheme, known as the Expectation Maximization (EM) method, is used to solve for $\mathbf{W}$ and β . Suppose that the current weight matrix is given by $\mathbf{W}^{\text{old}}$ and the current value of β is β^{old} . These values are used to evaluate the responsibilities R_{in} for the current iteration (the E-step). Subsequently, these estimates of R_{in} are substituted into Equations 4.14

and 4.16, which are then solved to give the new values of $\mathbf{W}^{\text{new}}$ and β^{new} (the M-step). These steps are repeated until stable values are reached. The EM method is described in detail in Appendix C.

Equation 4.14 can be written in the following form:

$$\Phi^T \mathbf{S}^{\text{old}} \Phi (\mathbf{W}^{\text{new}})^T = \Phi^T \mathbf{R}^{\text{old}} \mathbf{X} \quad (4.17)$$

where Φ is a $K \times M$ matrix with elements $\Phi_{ij} = \phi_j(\mathbf{y}^i)$, $\mathbf{X}$ is an $N \times D$ matrix with elements x_k^n , $\mathbf{R}$ is a $K \times N$ matrix with elements R_{in} , and $\mathbf{S}$ is a $K \times K$ diagonal matrix with elements

$$S_{ii} = \sum_{n=1}^N R_{in}(\mathbf{W}, \beta). \quad (4.18)$$

Equation 4.17 is solved for $\mathbf{W}^{\text{new}}$ using standard matrix-inversion techniques. Note that Φ is constant throughout the algorithm, and so need only be evaluated once at the start.

4.4 Kohonen and Sammon results

4.4.1 Meteosat database

The first fifteen features from the RBF feature selection method shown in Table 3.6 were used as inputs to the Kohonen and Sammon maps. Each feature vector was scaled to zero-mean and unit variance prior to running the algorithms. For the Kohonen map, a 10×10 square grid was chosen. The weights were initialized in the range ± 0.2 . The initial learning rate was 0.8, with a neighbourhood radius of seven units. During 50 passes through the training set, the neighbourhood radius was gradually reduced to one.

A balanced dataset was used for training, with equal numbers from each class represented. It is not necessary to have a separate test set since the aim is not classification; rather it is to visualize how the classes are related in feature space. After training, all samples are again applied, this time making use of the class labels attached to them. The *cumulative*

total of activations for each unit is recorded separately for each of the classes. Each unit is then displayed in the colour of the dominant class for that unit. The dominant class is that class which has activated the unit most *in proportion to* how many samples of that class were presented to the network. For example, if 51 out of 100 presented Dynamic cloud samples and 100 out of 200 presented Clear samples activate a particular unit, the more dominant class of the two would be Dynamic cloud.

Figures 4.6(b) and 4.6(c) show the result of such a procedure, using the *NetSat* system. It can be seen that each class has been mapped to a separate area of the grid, and there is very good separation between the classes. Different starting weights and order of presentation of training points were used for each of the two runs, giving different two-dimensional projections but preserving the relationships between classes. The two mappings are very consistent, being almost mirror images of each other. The Clear Sea and Deep Convective classes have been mapped to opposite corners of the grid, as might be expected since sea is relatively dark and warm compared with bright, cold cumulonimbus cloud tops. The Clear Land class surrounds the Clear Sea class.

Broadly speaking, it is apparent that the mapping of Figure 4.6(b) is ordered such that the low (warm) classes are to the left of the map, and the classes with uniform texture in the lower region. The Dynamic class incorporates both cirrus (high-level) and stratus (low-level) and so has a projection from the right edge of the map which continues towards Shallow Convection (low-level cumulus). The clean boundaries between classes indicate that they are well separated in feature space.

The Kohonen network was trained with a number of combinations of the following parameters: learning rate, initial and final neighbourhood radius, size of competitive layer, number of inputs, initial random weight range and number of passes through the training set. The final mapping was found not to be very sensitive to changes in these parameters, provided they remained within the sensible ranges set out previously.

The batch *K*-means algorithm was used to produce 100 prototype vectors from the Me-

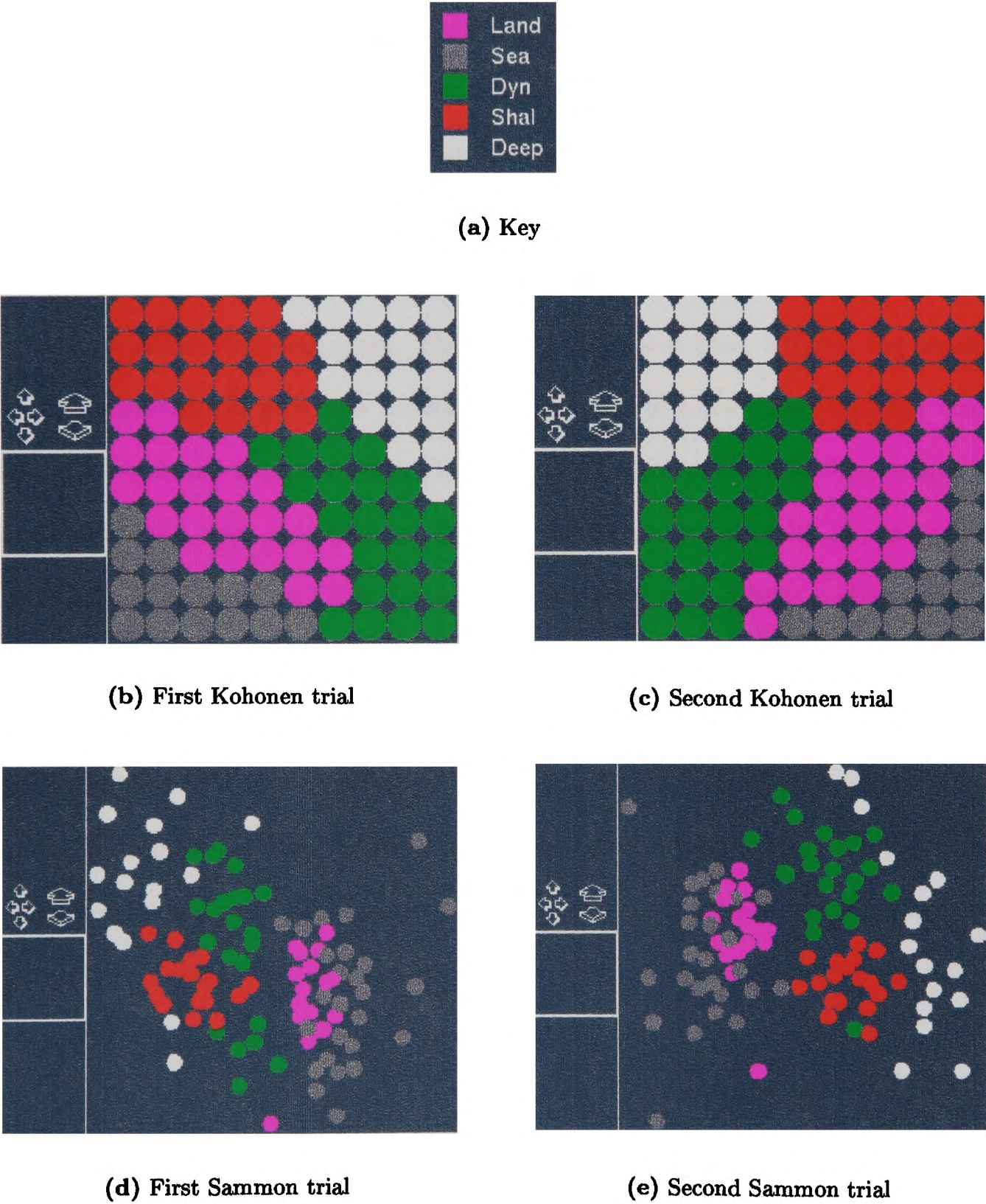


Figure 4.6: Kohonen and Sammon trials for the Meteosat database using the *NetSat* system. Plots (b) and (c) show two separate training results for a 10×10 Kohonen map while (d) and (e) show the results of Sammon mappings of two separate sets of 100 prototype vectors obtained with the batch *K*-means algorithm.

teosat database, and these were used as inputs to the Sammon mapping. Two different sets of prototype vectors were generated and the resulting Sammon mappings are shown in Figures 4.6(d) and 4.6(e). As with the Kohonen maps, the plots are produced by presenting all labelled samples after training and assigning each unit to the dominant class. For the Sammon case, the “activated unit” is that which has the smallest Euclidean distance to the presented vector. The position of each unit on the display is simply that given by the final positions of the Sammon data points y_{ik} of Equation 4.2.

It is evident that the projections of the class regions onto the Sammon maps are very similar to those of the Kohonen maps. The main difference is that, without the constraint of a square two-dimensional grid, the points corresponding to Clear Land and Sea tend to be well separated spatially from the other classes, but close to each other. This might indicate that in classification tests (see Chapter 5) there will be more difficulty in separating Clear Land from Clear Sea, but not Clear Land or Sea from the rest. There is also a gap between the Land/Sea classes and the others. The Dynamic class appears to be divided into two regions in Figure 4.6(d). This is probably due to the fact that cirrus tends to be high and cold, with a similar profile to cumulonimbus, whereas stratus, also a dynamic cloud, is lower and therefore closer to the lower-level classes. In addition, evidence from the Bankert results in the next section suggests that many cirrus samples have a profile that is contaminated by infra-red radiation from the warmer surface below it. Again, the ordering is quite consistent, Figure 4.6(d) being almost a mirror image of Figure 4.6(e).

4.4.2 Bankert database

Figure 4.7 shows the results for the Bankert dataset, using, as before, the first fifteen features from the RBF feature selection method shown in Table 3.7 and balanced datasets. The same training procedures were employed as before, the Kohonen maps being shown in Figures 4.7(b) and 4.7(c), and the Sammon maps in Figures 4.7(d) and 4.7(e).

It is apparent that the classes are not as well separated as the Meteosat classes, with the

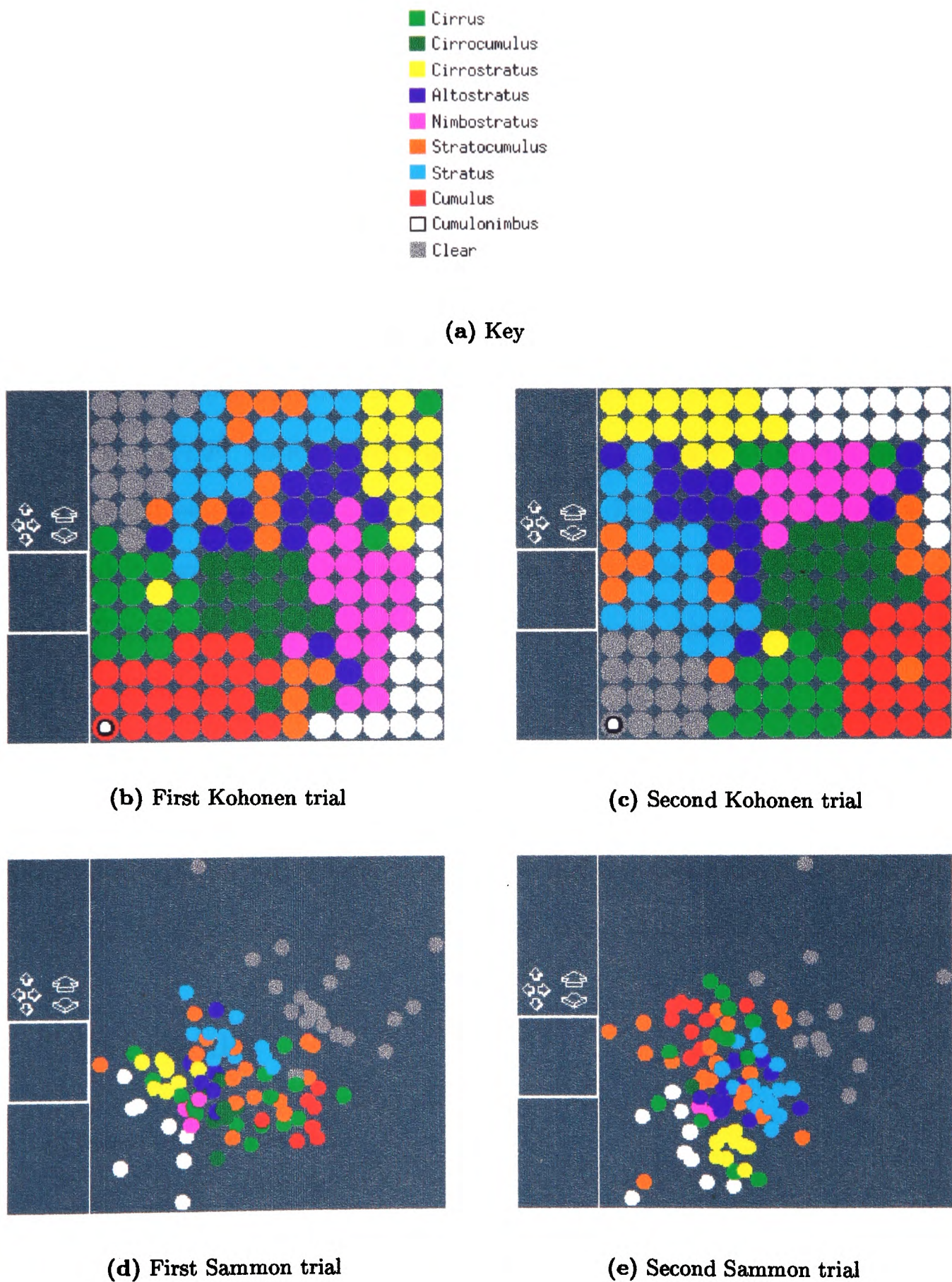


Figure 4.7: Kohonen and Sammon trials for the Bankert database using the *NetSat* system. Plots (b) and (c) show two separate training results for a 13×13 Kohonen map while (d) and (e) show the results of Sammon mappings of two separate sets of 100 prototype vectors obtained with the batch *K*-means algorithm.

Stratocumulus and Altostratus classes in particular not being confined to a particular region. The Sea category is again placed in the opposite corner to the Cumulonimbus category, and for Figure 4.7(b) the ordering on the map is approximately as follows: low classes to the left and smoothly-textured classes to the top. An obvious exception is Cirrus, which is a high cloud, but which has been placed next to the low clouds. This is probably due to the fact that cirrus is a thin cloud and infra-red radiation from the warm surface below it penetrates the cloud and contaminates the radiation from the cloud itself. Thus areas containing cirrus cloud may appear to be warmer than they actually are.

A comparison can be made with the Kohonen maps from the Meteosat database. Rotating Figure 4.7(c) through 90° anticlockwise and comparing it to Figure 4.6(c), it is apparent that there is broad agreement between the two orderings. Deep Convective cloud is mapped to the upper-left corner (Cumulonimbus), Shallow Convective cloud to the upper right (Cumulus), Clear Sea to the lower right (Clear), and Dynamic cloud to the lower left (Stratus, Cirrostratus, Altostratus).

The two Kohonen maps from the Bankert database are also consistent with each other, with Figure 4.7(b) being almost a rotated version of Figure 4.7(c). The structure of the classes in feature space revealed from the Kohonen and Sammon plots is very similar to that revealed by the GTM results in Section 4.5.2, and further discussion is deferred until then.

4.5 GTM results

4.5.1 Meteosat database

For training the GTM, the first six features from Table 3.6 were used, compared with fifteen for the Kohonen map. Fewer inputs was found to give a more consistent ordering on the final mapping for the GTM. All (unlabelled) samples were used, with 144 latent space points arranged on a 12×12 grid, and 25 basis functions arranged on a 5×5 grid with standard deviations equal to the distance between basis function centres. The labelled samples were then presented again for visualization purposes. Each point produces a posterior distribution

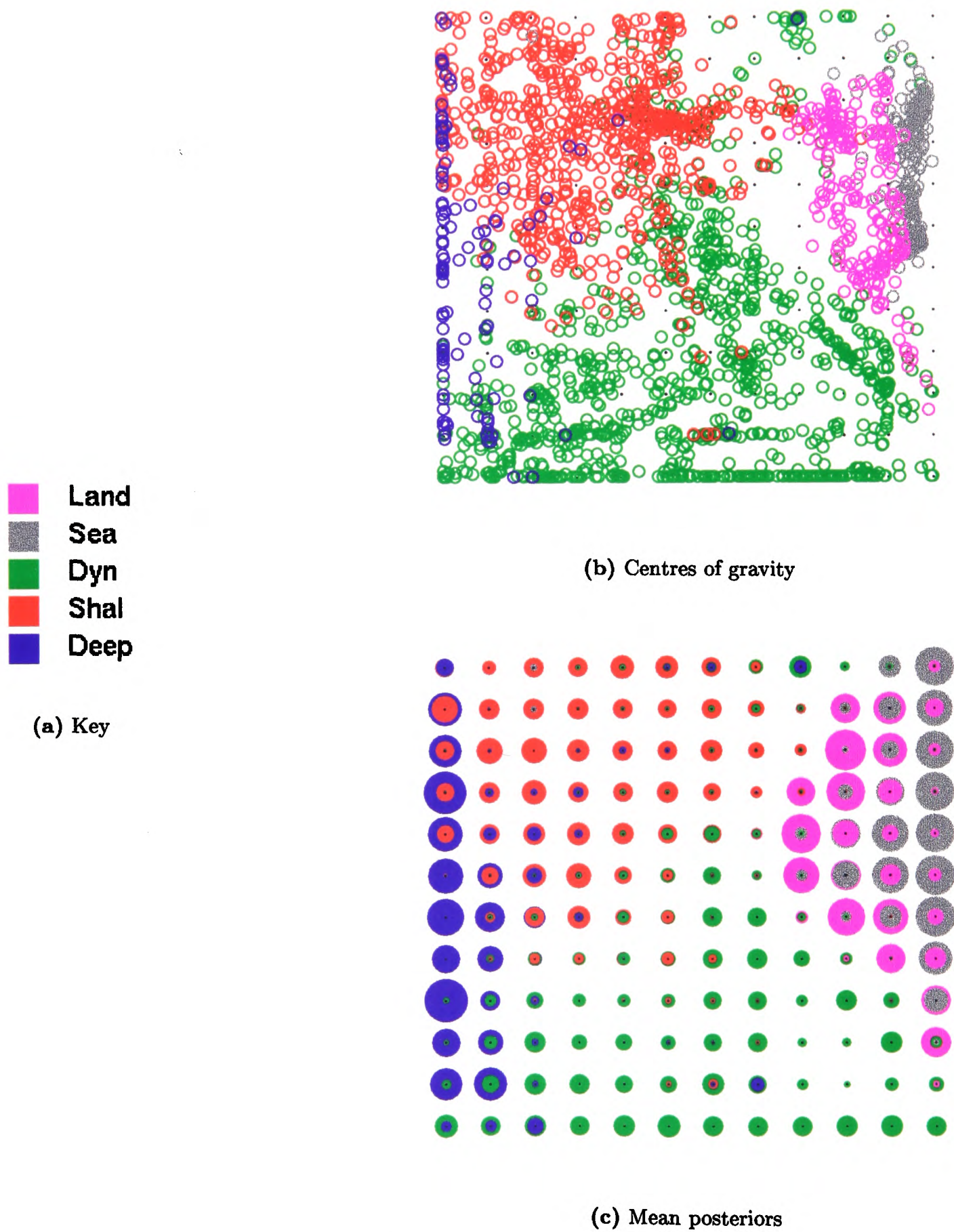


Figure 4.8: GTM output for a 12×12 latent space grid with 6 input dimensions for the Meteosat database. A 5×5 grid of basis functions was used, each with standard deviation equal to the spacing between the basis functions.

on the grid points. Figure 4.8(b) shows the centres of gravity of the distribution for each sample, coloured according to class. The mean posterior for each grid point over all the samples is plotted in Figure 4.8(c). This is formed by summing the posterior contribution to each grid point for every sample, and dividing by the number of samples. This is done separately for each cloud class, which are shown in differently-coloured circles with radius in proportion to the mean for that grid point.

A very similar ordering to the Kohonen maps is apparent. The Clear Land class surrounds the Clear Sea class, as it did for the Kohonen maps of Figures 4.6(b) and 4.6(c), and the grouping together of these two categories away from the others is like that of the Sammon plots of Figures 4.6(d) and 4.6(e). Clear Sea is also mapped to the opposite side of the map to Deep Convection. If the GTM map of Figure 4.8(c) is rotated through 180° , it is extremely similar to the Sammon map of Figure 4.6(e).

Since each point can be seen individually in Figure 4.8(b), a small amount of contamination of each class within the neighbouring class is apparent, and the Dynamic class is split in a similar fashion to that seen in the Sammon plot of Figure 4.6(d). Overall, very good class separation is again apparent, even with fewer input features. This indicates that six features is enough to provide good discrimination. The results of the RBF feature selection method from Section 3.3.2 also showed that the RBF accuracy dropped only slowly as features were removed.

4.5.2 Bankert database

The first six features from the RBF feature selection method shown in Table 3.7 were used as before. All samples were used, with 225 latent space points arranged on a 15×15 grid, and 25 basis functions arranged on a 5×5 grid with standard deviations equal to the distance between basis function centres. The results are shown in Figure 4.9. Details using the same mapping, but displaying fewer classes on each plot, are shown in Figures 4.10 and 4.11.

The Clear class forms a reasonably tight cluster which is well separated from other classes.

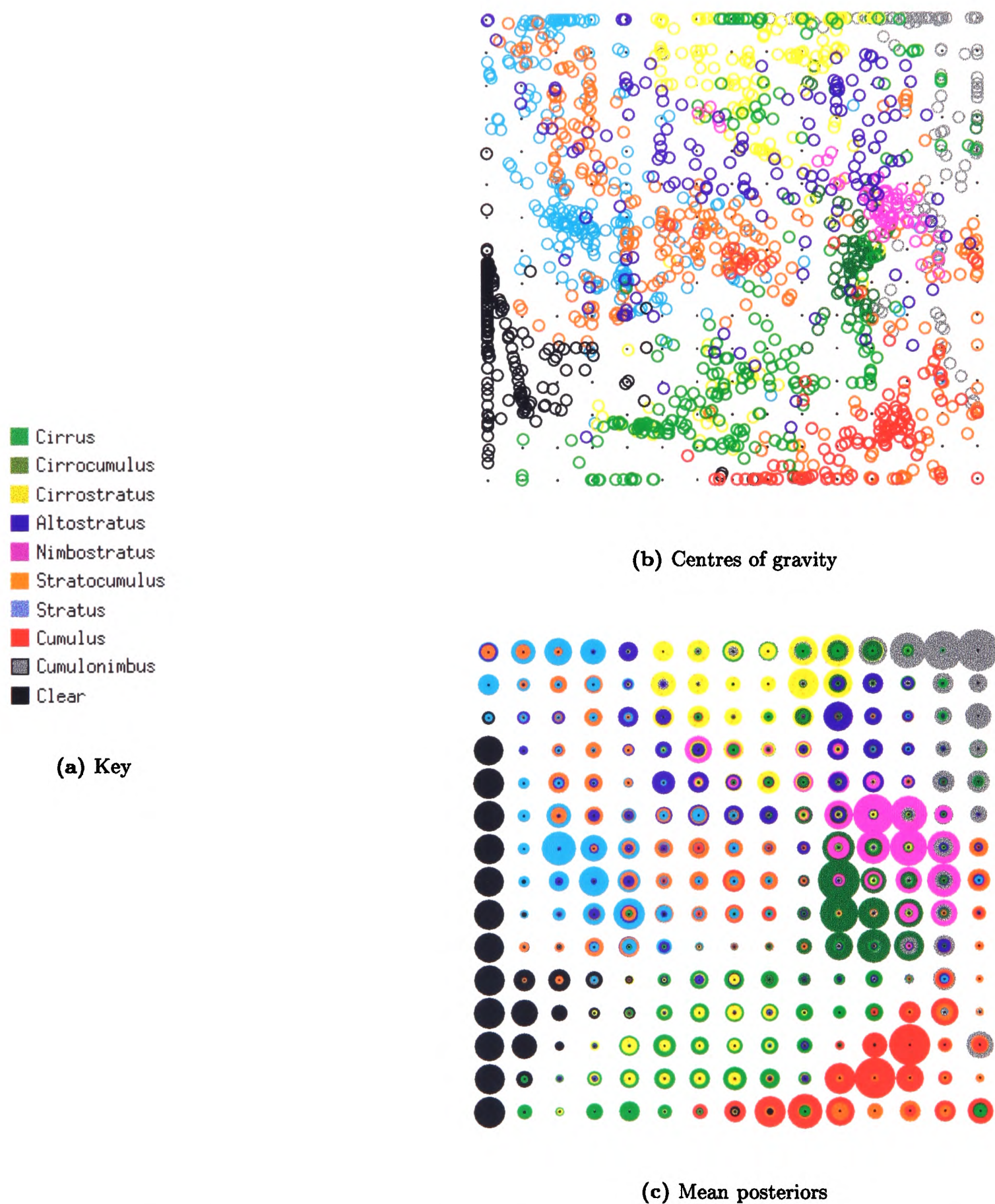
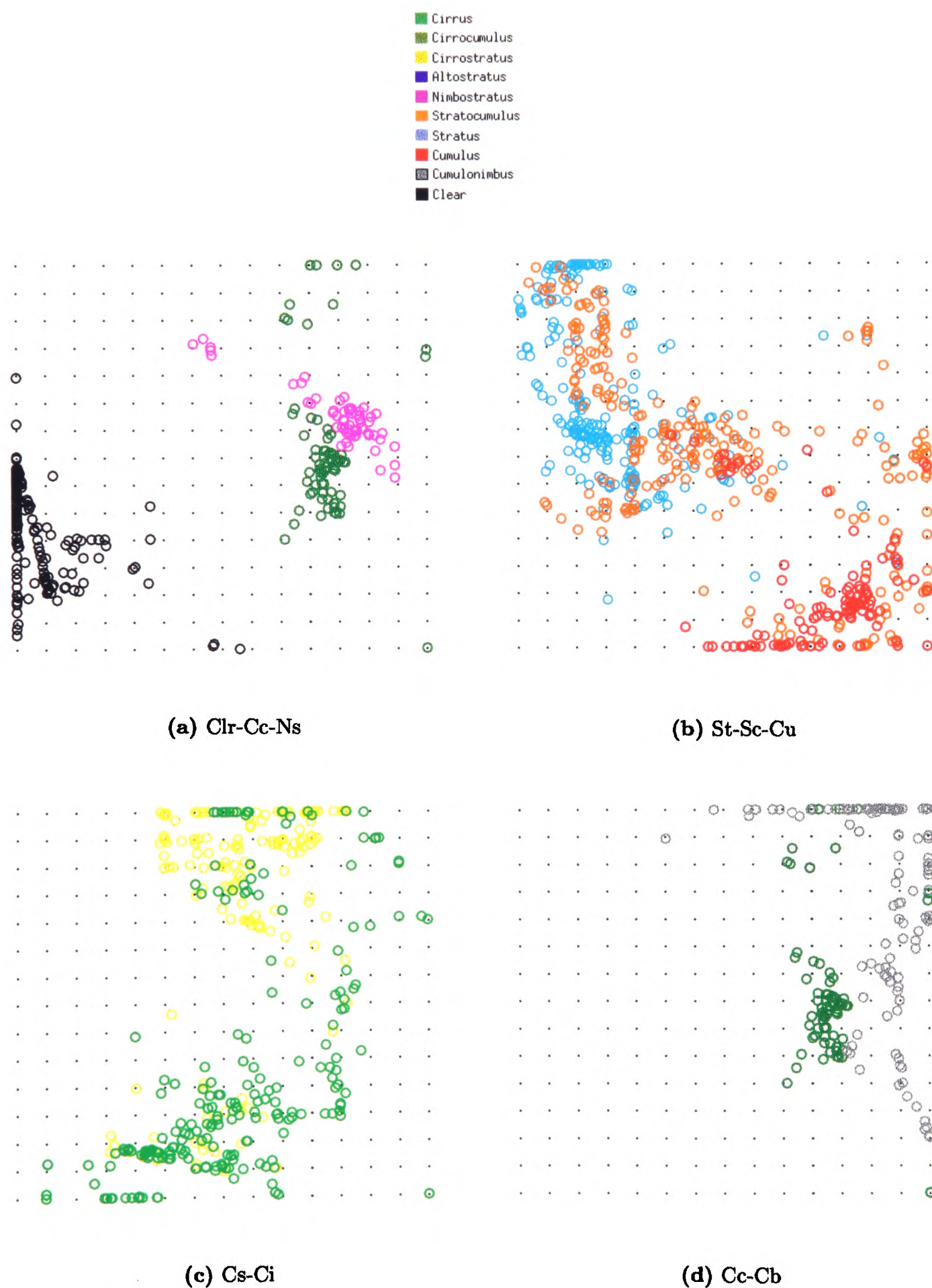


Figure 4.9: GTM output for a 15×15 latent space grid with 6 input dimensions for the Bankert database. A 5×5 grid of basis functions was used, each with standard deviation equal to the spacing between the basis functions.



This is more easily seen in Figures 4.10(a) and 4.11(c). A high accuracy is expected for the classification tests in Chapter 5 for this class. Figure 4.10(a) also shows that Cirrocumulus and Nimbostratus are quite close together, but reasonably well separated.

The Stratocumulus samples tend to be well spread in feature space, covering the area ranging from the reasonably well-defined Stratus cluster to that of Cumulus, as shown in Figure 4.10(b). This is a manifestation of the fact that stratocumulus shares characteristics of both stratus and cumulus, to varying degrees. All three are low-level cloud types, so the main distinguishing feature is the texture apparent in the visible image. The ordering is approximately Cumulus-Stratocumulus-Stratus with increasing smoothness of texture, and this is shown in the ordering on the GTM plot. Since Stratocumulus is spread across the map, it is expected that there will be some confusion between it and other classes.

Closer inspection of the Cirrus distribution reveals the effect described previously with regard to the thin nature of the cloud. There is a main cluster in the lower half of Figure 4.10(c), but a significant proportion of Cirrus samples are also mapped to a separate cluster associated more with the Cirrostratus class, in an area associated with high cloud types. There is significant mixing of these two classes, which may be difficult to separate in classification tests. Figure 4.10(d) shows that Cirrocumulus and Cumulonimbus should be easily separated.

The Cirrostratus-Nimbostratus-Cumulonimbus triplet of Figure 4.11(a) shows the three classes close in feature space, with reasonable separation but with some mixing between all three at the boundaries. Hence some misclassifications are expected between these three classes. Although there is some mixing of the Cirrus and Cumulonimbus classes, shown in Figure 4.11(b), most of the Cirrus lies in an area well away from that of Cumulonimbus.

Like the Clear class, Cumulus forms quite a well-defined cluster, shown in Figure 4.11(c), although there is a small separated cluster which will cause some misclassifications. In addition, there is significant contamination from the Stratocumulus class, shown in Figure 4.10(b); hence the Cumulus classification accuracy may not be as good as the cluster definition sug-

gests. Finally, the Altostratus class is seen to be very indistinct and activates a wide area of the GTM, shown in Figure 4.11(d), not forming a distinct cluster in feature space.

4.6 Conclusions

It is vital to understand the structure of the dataset before employing any classification system. Three visualization methods have been presented which project the high-dimensional feature vectors onto a two-dimensional plane, allowing a visual inspection of the structure of the data. Although each of the methods make different assumptions in order to create the mapping from the feature space to the visualization space, they produced very consistent two-dimensional orderings. This provides confidence in both the feature sets and the conclusions that may be drawn from the results.

The Generative Topographic Mapping is a very recently-developed technique, and hence there have been very few applications of it so far to real-world data. Some of the advantages of the GTM algorithm over the Kohonen map were discussed by Bishop *et al* [9]. The key advantage stated was that there is a well-defined objective function to be minimized in the GTM algorithm. The Kohonen algorithm requires arbitrary choices for the learning rate and neighbourhood parameters to control the smoothness of the mapping. The results of this chapter also show that there are a number of practical advantages in the way that the GTM latent space information can be displayed, compared with the Kohonen and Sammon mappings. For every data point $\mathbf{x}$ there is a posterior distribution $p(\mathbf{y} \mid \mathbf{x}, \mathbf{W}, \beta)$ defined on the latent space. This posterior distribution can be displayed in different ways, including the two used here.

Firstly, the position of the mean of the posterior distribution in latent space (the centre of gravity) can be calculated for each sample in the dataset. This allows a one-to-one mapping from a point in data space to a point in latent space, rather than a many-to-one mapping for the Kohonen and Sammon mappings. The centres of gravity are not restricted to lie on the grid points, and in this way the GTM is similar to the Sammon mapping where the

prototype vectors are free to occupy any point on the plane. An example of this was found when comparing the Sammon plot of Figure 4.6(d) with the GTM plot of Figure 4.8(b). This one-to-one mapping of the GTM algorithm allows each point in the dataset to be visualized separately, or all at once. In this way, the GTM method provides a much clearer view of the overlap of cloud classes in the Bankert database, as demonstrated by the plots of Figure 4.10.

Secondly, each data point produces a posterior probability for each GTM latent space node, and these posterior probabilities can be summed over all data points. This produces results like those of Figure 4.9(c) and those in Figure 4.11. This method of display is very similar to that produced by the Kohonen map. An even closer link could be made if, for each presented sample, a winning node were selected as it is for the Kohonen map; the winning node would be the latent space point with the highest posterior probability for that sample (the mode of the distribution).

Therefore the recently-developed GTM technique has been shown to have several advantages over the Kohonen and Sammon maps, although all three techniques have provided consistent mappings allowing a good visual analysis of how the the Bankert and Meteosat samples are arranged in feature space. For the Meteosat database, it is clear that the classes are reasonably distinct and separate in feature space, while for the Bankert database there is some overlap between classes. This is probably due to the fact that the Bankert classes are more narrowly-defined than the Meteosat classes. For the GTM results, only the best six features from the RBF feature selection method of Chapter 3 were used, compared with fifteen for the Kohonen and Sammon results. These six features appeared to contain sufficient information to produce good results, which provides validation of the feature selection method employed.

The visualization results presented in this chapter will be invaluable for the full interpretation of the classification results to be presented in Chapter 5, and will be considered again in the light of the results of that chapter.

Chapter 5

Classification tests

Section 1.4 introduced the image databases used for this study, a collection of 458 visible and infra-red pairs. It also described the expertly-labelled samples in each database. Chapter 2 set out the theory of the neural network classifiers employed for this study, and Chapter 3 gave a thorough description of the pre-processing and feature extraction that must occur in any classification system. Finally, Chapter 4 presented three methods for visualizing the available data in multi-dimensional feature space, and these provided a degree of understanding of the amount of overlap between the different classes of the Meteosat and Bankert databases.

Classification techniques can now be applied to both databases. The method of linear discriminants, described in Section 2.1.1.1, is firstly used to establish benchmark classification results against which the performance of the MLP and RBF neural networks can be evaluated. The main discussion of the results will be deferred until Section 5.5 in which the results from all three methods are compared and contrasted, and assessed in the light of the visualization results of Chapter 4. This chapter starts, however, with an overview of previous work in automated cloud classification, using both classical and neural techniques.

5.1 Overview of previous work

A number of studies that have used satellite imagery to detect and classify clouds have been reported in the literature and it is the aim of this section to provide an overview of past research in this area. For the purpose of this survey, the research has been divided into two broad categories: general methods, described in Section 5.1.1 and neural methods, described in Section 5.1.2. An independent review of the pattern recognition techniques that had been applied to satellite image interpretation and cloud identification until around 1994 was conducted by Pankiewicz [71, 73].

5.1.1 General classification work

Much of the early work in this field is described in Parikh [79], which covers the period from about 1970 to 1976. The author's aim was to assess the value of different spectral and textural features for classifying clouds using coarse cells. Comparisons were also made between single-step classifiers and hierarchical "decision-tree" classifiers. The study concluded that the best results were obtained using the spectral features only, in a single-stage maximum likelihood classifier, and that textural features provided little benefit. More recent evidence, described later, tends to the opposite conclusion. Harris and Barrett [45] used the crude textural features of the standard deviation of a 5×5 box of pixels and a measurement of vector dispersion (a measure of surface roughness) to achieve accuracies of 73% for a four-class problem (Stratiform, Cumuliform, Strato-cumuliform and Clear).

An important classification task is that of segmenting the image into cloudy and clear pixels. Rossow *et al* [93] summarized early work in this area, and compare a number of the methods which include straight thresholding, with or without radiative transfer theory, and cluster analysis.

In a 1982 paper, Desbois *et al* [29] used an unsupervised clustering technique called the "dynamic cluster method". It is similar to the batch K -means algorithm described in Section 4.2.1.1, with the added step of removing a cluster should its population fall below

a certain number. Significantly, no textural information was used, the sole inputs being the infra-red and visible spectral values and the water-vapour channel value on a pixel-by-pixel basis. The method generally led to 5 or 6 clusters, which corresponded to broad cloud classes such as “convective” and “low”.

Related clustering studies include that by Pairman and Kittler [70], in which a similar method to Desbois *et al* [29] was used, but which included the covariance of each cluster in the calculations. This improved the separability for clusters with widely-differing covariances, and was found to improve the detection rate for water with sun-glint and sub-pixel cumulus. Seddon *et al* [100] used a clustering algorithm that allowed both addition *and* removal of clusters, and the clustered image gave good visual correlation with the original image.

Parallel to this research, the use of context as an aid to classification was being investigated. The papers by Kittler and Föglein [51] and Kittler and Pairman [52] provide a good summary of this work. The latter study made use of a Bayesian maximum likelihood classifier, with additional models needed to estimate the class-conditional probabilities, most notably the Gaussian model and a Markov Random Field (MRF) model [37] (see Section 6.2.1). For each pixel, spectral values *and* class labels of neighbouring pixels were incorporated into the feature vector. This use of context avoids the noise generally associated with single-pixel classification and is considered further in Chapters 6 and 7.

Garand [35] used 13 physical features for his 1988 paper. These included “cloud fraction”, “streakiness factor” and “cloud height”. Hence some morphological (rather than strictly textural) and physical information was incorporated into the features. A Gaussian maximum likelihood classifier, 20 cloud classes, and a coarse grid method were used. In order to evaluate the results, three human experts were used independently to classify the samples used. All three experts agreed with the automated classification for 55% of the samples, but this figure rose to 79% for a “consensus” opinion with at least two agreeing. This shows the subjective nature of cloud classification and how it is a non-trivial problem even for human experts.

By the mid to late 1980’s, textural features were being seen as having an important

part to play in cloud classification algorithms. In 1987, Ebert [33] considered 36 textural features (including GLDV statistics and Fourier descriptors) as well as 30 spectral features calculated for each coarse grid sample. After feature reduction, eight features were chosen for the Bayesian maximum likelihood classifier (Gaussian model), three of which were textural features. Accuracies of 84% were reported for an 18-class dataset. As for Garand [35], this was the consensus opinion of three experts. Again the subjective nature of the problem is demonstrated by the fact that of the samples classified by the three experts, only 45% had all three experts in agreement on its class.

After the Desbois *et al* paper [29] of 1982, two of the authors continued the study [105] by adding a crude textural measure to improve cluster definition, this being the standard deviation of the nine pixels in a 3×3 box around each pixel.

Welch *et al* [112] and Chen *et al* [21] investigated the use of GLCM [81, 107], GLDV and SADH features for texture description, using a range of pixel separations in the algorithm. Landsat images (see Appendix A) were used with three classified cloud classes (Cumulus, Stratocumulus and Cirrus). Using a Gaussian maximum likelihood classifier with the GLCM descriptors, they achieved classification accuracies of 83–85% using random sub-region hold-out, whereby individual samples are held out for testing, and 92–95% for random scene hold-out, for which a whole image is held out for testing. The GLDV and SADH features were found to give similar accuracies to those for the GLCM features, but with substantial savings in computing time and storage.

Textural features were also an important part of the 1990 study by Key [49]. A combination of grid and pixel-by-pixel methods was used, whereby a 16×16 pixel grid was moved across the image two pixels at a time and an average value of each feature assigned to each pixel. Again a Gaussian maximum likelihood classifier was used, and an accuracy of 68% was claimed for one of the test images.

5.1.2 Neural network research

At around the same time as the Key study [49] at the end of Section 5.1.1, neural network techniques began to be applied to cloud classification problems. The 1989 study of Key *et al* [50] used an MLP trained on the spectral values of AVHRR data, as well as the values of two channels of a satellite-borne microwave radiometer. An accuracy of only 53% was recorded, for eight cloud and four land types. This was probably due to a very small dataset consisting only of 1% of one image. When 9% of the image was used with a maximum likelihood classifier, an 85% accuracy was achieved, but this dataset was not applied to the MLP, the latter being used mainly for weight interpretation in order to gain some insight as to which spectral channels were used to distinguish between classes.

Lee *et al* [57] also used an MLP to classify 512×512 pixel (29×29 km) boxes into one of only three cloud types with an accuracy of 93%. Considering that only the visible channel was used, this represents a very good accuracy. In addition, the fact that *only* visible textural features were used added to the weight of evidence in favour of the use of textural features. This study used the same Landsat images and classes as Welch *et al* [112] and Chen *et al* [21] described above, so an interesting comparison can be made.

In 1992, Welch *et al* [113] used both an MLP trained with the back-propagation algorithm and Specht's probabilistic neural network [108, 113], ten cloud/ice classes and 64×64 pixel (70×70 km) regions. The probabilistic neural network is essentially a non-parametric density estimator similar to that described in Section 2.1.1.2. Twenty spectral and textural features were selected from a total of over 200, giving overall accuracies of 88% and 87% respectively for the two neural approaches.

In 1993, Tovinkere *et al* [109] applied a variety of artificial intelligence techniques to the problem of distinguishing a number of cloud types and ice in polar regions. The techniques used were (1) an MLP; (2) Specht's probabilistic neural network; (3) a hybrid neural network; (4) a "don't care" perceptron model; (4) a "don't care" MLP model; and (6) a fuzzy logic based expert system. The hybrid model consisted of an unsupervised Kohonen-like layer

preceding an MLP layer. For the “don’t care” models, each output was designed to distinguish between two specific classes, rather than between one class and all the rest, so that there were $m(m-1)/2$ outputs for m classes. The decision surface (see Section 2.2.3) from each output therefore separates those two classes but may cut through others. The poorest performer was the hybrid network, with 86% accuracy, while the “don’t care” MLP model was best with 98.7% accuracy. The excellent performance was partly attributed to the use of very pure training and testing samples. However, for a comparison, further tests were carried out using the *same* dataset as was used for the study by Welch *et al* [113]. An accuracy of 92% was reported for the “don’t care” MLP model, 4% higher than for a standard MLP.

A similar study to that by Welch *et al* [113] was carried out by Bankert [5], who also used a probabilistic neural network, but the ten output classes were nine cloud types plus “clear”, all samples being taken over maritime regions. Fifteen spectral and textural features gave an accuracy of 79.8% using the leave-one-out method. A 2-level 4-network system was designed in an attempt to improve accuracy. Eight of the original ten classes were grouped into three broad categories, the final two classes remaining distinct. The first layer classified samples into one of the five classes, and the three networks in the second layer sub-classified each of the three grouped categories back into their original classes. Accuracy was found to be 81%, very similar to the single-layer method, but the accuracy produced by the first layer (broad classification) was 91%. Note that this is one of datasets investigated in this thesis, which is described in Section 1.4.1. A further study by Bankert and Aha [6] considered a different selection of features, reporting a leave-one-out accuracy of 86.8%.

Other applications using neural networks include work by Peak and Tag [82, 83] whose ultimate aim is the recognition of synoptic-scale features such as cyclones, jet streams and frontal systems, as opposed to the recognition of the basic cloud types making up such synoptic features. The 1992 paper [82] required a human to identify the synoptic shapes, and an MLP was used to classify the shapes using input features such as the spatial extent and amount of high and low cloud. An accuracy of 83% was reported. They proposed

a method for automatic segmentation based on a bottom-up approach whereby individual pixels are first classified into cloud types, and then a segmentation technique is used to identify specific synoptic features. The authors of the 1994 paper [83] suggested essentially a top-down approach where the image is segmented first according to large-scale features and then subsequently each region is analysed in more detail. Only the segmentation process was described, and it was based on a neural network used in a hierarchical threshold method. These papers are considered again in Chapter 7.

Segmentation is also dealt with by Yhann *et al* [117] who used an MLP for essentially a two-class problem, that of determining whether a pixel is cloud or ocean. The study included both day-time and night-time (*i.e.* no visible data) investigations, leading to accuracies of 84% and 92% respectively.

Pankiewicz [72] used an MLP and nine simple spectral features, using the same dataset as was used in the Bankert study [5]. The aim was to train the MLP on AVHRR data and to subsequently apply this trained network to Meteosat data in order to look at the degradation in classification accuracy resulting from using slightly different spectral channels and lower resolution. The ten classes were grouped into four broader categories: Clear, Shallow and Deep Convective, and Dynamic. Accuracy on the AVHRR dataset was 76%, and the same network weights achieved an accuracy of 68% on the Meteosat samples.

A further study by Pankiewicz [74] used the same Meteosat database as the one used in this thesis, previously described in Section 1.4.2. The aim of the study was to identify regions of convective activity for use as part of the flood forecasting system designed by the UK Meteorological Office (see Section 1.3). The two clear categories were combined into one, and seven spectral and two temporal features were used as inputs to an MLP. No textural features were used in order to save processing time. An accuracy of 80.8% was achieved. In a related study [76], Pankiewicz used a two-stage process in order to introduce a scale-dependent aspect. The class output of a fast, simple spectral-feature MLP was used to choose the size of the textural mask for the calculation of just two textural features for a fast, second

MLP. In this way the accuracy was improved to 83.5%. The emphasis of these studies was to produce robust classifiers able to perform throughout the year and during local daylight in an operational environment; the test results reflected this requirement.

5.1.3 Overview

It is impossible to make a direct comparison between all the studies presented in this section. Many different databases have been used, each with different numbers of classes and different labelling schemes, resulting in a wide range of reported accuracies. A common conclusion, with the notable exception of Parikh [79], is that textural features are very important in combination with the spectral features. Also apparent is the increasing use of neural techniques in recent years applied to cloud-type identification.

The set of studies that can be used as a benchmark against which to compare the results of this thesis are those by Bankert [5, 6] and Pankiewicz [72, 76, 77], since they use the same databases and similar features to those used for this thesis. They will be considered again in Section 5.6.

5.2 Linear discriminant tests

The linear discriminant method set out in Section 2.1.1.1 will be used as a benchmark against which the non-linear neural network methods can be compared. The class-conditional probability densities $p(\mathbf{x} \mid C_i)$ for each class are modelled by a multivariate Gaussian function, with Σ_i and μ_i being the covariance matrix and mean vector for the data points in class i . In the case of *linear* discriminants, the average covariance matrix Σ for all classes is used, that is:

$$\Sigma = \frac{1}{C} \sum_i \Sigma_i \quad (5.1)$$

where C is the number of classes. Therefore it is assumed that the correlation between features is the same for all classes. A test feature vector $\mathbf{x}$ is then assigned to the class with the greatest $g_i(\mathbf{x})$ given by Equation 2.9 [8].

All samples in the Meteosat and Bankert datasets were used, 70% being used to calculate the model parameters, and the remaining 30% set aside for testing. The feature vectors were composed of the first fifteen features from the RBF feature selection method shown in Tables 3.6 and 3.7, and all the output categories were included. Using fifteen random allocations of data points to the training and test sets, the average accuracy achieved for the Meteosat test sets was 85.0%, with a standard deviation of 0.8%, while for the Bankert test sets it was 64% with a standard deviation of 1%. The average confusion matrices are shown in Tables 5.1 and 5.4 (on Pages 101 and 102).

5.3 MLP classification tests

In order to train and then evaluate the performance of the MLP, the Meteosat and Bankert datasets were divided into three parts: the training set, the validation set, and the test set. The validation set is used not for weight modification but to decide when to stop training, as explained below.

Each feature vector must be scaled so that every component of the vector lies within a similar range. In theory, an MLP can adjust its weights to cope with whatever range is used, but widely-varying ranges for different inputs can lead to slow training. If the inputs (or equivalently the initial weights) are too large, the units in the first layer are likely to saturate, producing small error gradients. For this reason, each input in the training set is scaled linearly to zero-mean and unit-variance, and the validation and test set vectors scaled using the same scaling parameters as for the training set.

The mean square output error (MSE) on the validation set is used to monitor the progress of training. During the training procedure, the MSE on the validation set falls as the weights are adjusted, but after a certain point the MSE begins to rise as the MLP's input-output

1. Initialize the MLP's weights.
2. Set n , m and $p_{\max}$ to 0.
3. Cycle through all the vectors in the training set once, presenting them one at a time to the MLP.
4. Increment n .
5. If n is a multiple of ten, goto 6, else goto 3.
6. Measure the classification accuracy, p , and MSE, e , on the validation set.
7. If p is greater than $(p_{\max} - 1.0)$, save the weights as the current optimal set $\mathbf{W}_{\text{opt}}$, and set $p_{\max}$ to p .
8. If the change of e from its previous value is less than 0.0001, increment m .
9. If $m = 2$, goto 10, else set m to zero and goto 3.
10. Load back the optimal weights $\mathbf{W}_{\text{opt}}$ into the MLP.

Figure 5.1: The scheme used to decide when to stop the training procedure for the MLP. The counter n holds the number of cycles through the training set and $p_{\max}$ is used to store the largest classification accuracy achieved so far on the validation set. The counter m is used to monitor the changes in the mean square error (MSE).

mapping becomes too specific to the training set being used [8]. Therefore training is stopped when the MSE on the validation set begins to rise in order to avoid over-fitting; this is known as *early stopping* [8].

Since the MLP is being used as a classifier, an alternative stopping strategy might be to stop at the point when the minimum classification error rate is reached on the validation set. However, the classification error curve is more noisy than the MSE curve and hence it is more difficult to find the optimal stopping point. Hence a combination of the MSE and of the classification error rate is used in this thesis to decide on the optimal weights set, described algorithmically in Figure 5.1. The scheme uses the classification accuracy to determine which weights to use, whilst still using the MSE to decide when to stop training.

The feature vectors were composed of the same fifteen features that were used for the linear discriminant benchmark experiments of the previous section. Fifteen hidden units were used, with all possible output classes (five for the Meteosat dataset and ten for the Bankert

dataset *i.e.* 15:15:5 and 15:15:10 MLPs respectively). Results presented elsewhere [13] indicate that altering the number of hidden nodes around fifteen has little effect on the resulting classification accuracy. A learning rate of 0.01 and momentum coefficient of 0.2 were used, with initial random weights in the range -0.2 to $+0.2$.

All samples in the Meteosat and Bankert datasets were used, 60% being used for training, and 20% each for validation and testing. Three different random sample allocations were made to create three different sets of training, validation and test sets. For each of these partitions, five separate training runs were performed using different random initial weights. In this way, an average over fifteen runs could be calculated. Each sample was classified according to the Bayes decision rule of Section 2.1 as the class that had the highest estimated posterior probability *i.e.* the class that represented the output that had the largest value.

The average accuracy achieved for the Meteosat test sets was $93.9\% \pm 0.6\%$, after an average of 1000 ± 200 cycles. The best accuracy achieved was 94.7%, and the confusion matrix for this network is shown in Table 5.2 on Page 101. For the Bankert test sets, the average was $78\% \pm 3\%$, after an average of 1400 ± 90 cycles. The best accuracy achieved was 81.9%, and the confusion matrix for this network is shown in Table 5.5 on Page 102.

Further trials were carried out using a reduced feature set consisting instead of the first five features from Tables 3.6 and 3.7 and a 70:10:20 percentage division of all the samples between training, validation and test sets respectively. The average accuracy achieved for the Meteosat test sets was $91.7\% \pm 0.4\%$, and the best accuracy was 92.5%. For the Bankert test sets, the average was $74\% \pm 2\%$, with the best accuracy being 76.8%. These represent slightly reduced accuracies, but there would be a significant saving in the time taken to calculate the feature vectors for each pixel. The confusion matrices are similar to those for the full set of fifteen input features and are not shown here.

5.4 RBF classification tests

A similar procedure to that followed in Section 3.3.2 was used here. All the samples in each dataset were divided into training and test sets in the ratio 3:1. Three different random allocations of the feature vectors to the training and test sets were made, to create three different sets of data. Subsequently, each of these sets were given five different random orderings, creating a total of fifteen training and testing set pairs. As before, the feature vectors were composed of the first fifteen features from the RBF feature selection method shown in Tables 3.6 and 3.7, and all possible output classes were used (five for the Meteosat dataset and ten for the Bankert dataset). The input vectors were scaled to zero-mean and unit-variance. This is essential for the RBF method since the hidden layer computes the Euclidean distance of each vector to the basis function centres. Vector elements having a wider range than others would dominate the distance value.

An RBF network with 200 kernel functions was employed. For each training run, using the per-class clustering procedure of Section 2.3.2.1, the 200 RBF centres were allocated equally to each class *i.e.* 40 per class for the Meteosat dataset and 20 per class for the Bankert dataset. The positions of the n kernels for each class were set to be those of the first n feature vectors belonging to that class in the training set, as described in Section 3.3.2. The widths of the kernels were then set using Equation 2.40, and the output layer weights set using singular value decomposition.

The average accuracy achieved on the test sets for the Meteosat dataset was $94\% \pm 1\%$, with the best accuracy over the fifteen trials being 95.4%. The confusion matrix is shown in Table 5.3. For the Bankert dataset, the average accuracy on the test sets was $80\% \pm 2\%$. The best accuracy achieved was 83.0%, and the confusion matrix for this trial is shown in Table 5.6.

Table 5.1: Average confusion matrix for Linear Discriminants for the Meteosat database, averaged over fifteen trials. The dimension of data space is fifteen. Overall classification accuracy is 85.0% (standard deviation of 0.8%). Rows represent the actual class, columns represent the predicted class. For class codes, see Table 1.3 on Page 21.

	Land	Sea	Dyn	Shal	Deep
Land	94.0	6.0	0.0	0.0	0.0
Sea	21.0	79.0	0.0	0.0	0.0
Dyn	9.2	0.3	79.0	5.8	5.8
Shal	0.5	0.0	3.1	92.3	4.1
Deep	2.8	0.0	8.3	0.0	88.9

Table 5.2: Confusion matrix for 15:15:5 MLP giving the best accuracy for the Meteosat database. Overall classification accuracy is 94.7%. The RMS output error is 0.13. Rows represent the actual class, columns represent the predicted class. For class codes, see Table 1.3 on Page 21.

	Land	Sea	Dyn	Shal	Deep
Land	97.8	0.0	2.2	0.0	0.0
Sea	14.6	85.4	0.0	0.0	0.0
Dyn	0.5	1.0	97.0	1.5	0.0
Shal	0.0	0.0	2.3	93.8	3.8
Deep	0.0	0.0	8.3	0.0	91.7

Table 5.3: Confusion matrix for RBF with 200 15-D centres giving the best accuracy for the Meteosat database. Overall classification accuracy is 95.4%. Rows represent the actual class, columns represent the predicted class. For class codes, see Table 1.3 on Page 21.

	Land	Sea	Dyn	Shal	Deep
Land	98.3	0.0	1.7	0.0	0.0
Sea	3.6	94.5	1.8	0.0	0.0
Dyn	0.4	0.0	98.4	1.2	0.0
Shal	0.6	0.0	6.3	93.0	0.0
Deep	0.0	0.0	22.7	4.5	72.7

Table 5.4: Average confusion matrix for Linear Discriminants for the Bankert database, averaged over fifteen trials. The dimension of data space is fifteen. Overall classification accuracy is 64% (standard deviation of 1%). Rows represent the actual class, columns represent the predicted class. For class codes, see Table 1.2 on Page 19.

	Ci	Cc	Cs	As	Ns	Sc	St	Cu	Cb	Clr
Ci	61.9	12.7	7.9	6.3	6.3	0.0	0.0	1.6	3.2	0.0
Cc	9.5	66.7	19.0	0.0	4.8	0.0	0.0	0.0	0.0	0.0
Cs	39.1	8.7	34.8	2.2	13.0	0.0	2.2	0.0	0.0	0.0
As	4.3	4.3	6.5	54.3	13.0	2.2	13.0	0.0	2.2	0.0
Ns	0.0	29.4	0.0	0.0	70.6	0.0	0.0	0.0	0.0	0.0
Sc	1.4	0.0	0.0	17.6	1.4	37.8	25.7	16.2	0.0	0.0
St	1.5	0.0	0.0	9.0	0.0	0.0	88.1	1.5	0.0	0.0
Cu	0.0	0.0	0.0	0.0	0.0	7.9	0.0	92.1	0.0	0.0
Cb	0.0	0.0	15.8	2.6	23.7	0.0	0.0	13.2	44.7	0.0
Clr	9.1	0.0	0.0	0.0	0.0	0.0	0.0	1.5	0.0	89.4

Table 5.5: Confusion matrix for 15:15:10 MLP giving the best accuracy for the Bankert database. Overall classification accuracy is 81.9%. The RMS output error is 0.37. Rows represent the actual class, columns represent the predicted class. For class codes, see Table 1.2 on Page 19.

	Ci	Cc	Cs	As	Ns	Sc	St	Cu	Cb	Clr
Ci	73.8	2.4	9.5	7.1	7.1	0.0	0.0	0.0	0.0	0.0
Cc	7.1	78.6	7.1	0.0	0.0	0.0	0.0	0.0	7.1	0.0
Cs	38.7	0.0	58.1	0.0	0.0	0.0	0.0	0.0	0.0	3.2
As	3.3	0.0	0.0	86.7	0.0	3.3	6.7	0.0	0.0	0.0
Ns	0.0	0.0	9.1	0.0	81.8	0.0	0.0	0.0	9.1	0.0
Sc	0.0	0.0	0.0	2.0	0.0	85.7	4.1	6.1	0.0	2.0
St	0.0	0.0	0.0	6.7	0.0	4.4	88.9	0.0	0.0	0.0
Cu	0.0	0.0	0.0	0.0	0.0	17.4	0.0	82.6	0.0	0.0
Cb	0.0	0.0	8.0	4.0	16.0	0.0	0.0	0.0	72.0	0.0
Clr	2.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.7

Table 5.6: Confusion matrix for RBF with 200 15-D centres giving the best accuracy for the Bankert database. Overall classification accuracy is 83.0%. Rows represent the actual class, columns represent the predicted class. For class codes, see Table 1.2 on Page 19.

	Ci	Cc	Cs	As	Ns	Sc	St	Cu	Cb	Clr
Ci	77.4	3.8	13.2	5.7	0.0	0.0	0.0	0.0	0.0	0.0
Cc	11.1	77.8	0.0	0.0	0.0	0.0	0.0	0.0	11.1	0.0
Cs	14.7	0.0	76.5	5.9	2.9	0.0	0.0	0.0	0.0	0.0
As	2.2	0.0	2.2	77.8	4.4	4.4	8.9	0.0	0.0	0.0
Ns	0.0	0.0	7.1	0.0	85.7	0.0	0.0	0.0	7.1	0.0
Sc	1.5	0.0	0.0	4.5	0.0	75.8	10.6	4.5	1.5	1.5
St	1.7	0.0	0.0	3.4	0.0	8.5	84.7	0.0	0.0	1.7
Cu	3.7	0.0	0.0	0.0	0.0	7.4	0.0	88.9	0.0	0.0
Cb	0.0	0.0	3.0	0.0	0.0	9.1	0.0	0.0	87.9	0.0
Clr	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.8	0.0	98.2

Table 5.7: A comparison of the average accuracy achieved for the three methods, Linear Discriminants, MLP and RBF.

	Meteosat database	Bankert database
Linear Discriminants	85.0% $\pm$ 0.8%	64% $\pm$ 1%
MLP	93.9% $\pm$ 0.6%	78% $\pm$ 3%
RBF	94% $\pm$ 1%	80% $\pm$ 2%

5.5 Discussion

A summary of the average classification accuracies achieved for the three methods, Linear Discriminants, MLP and RBF, is shown in Figure 5.7. For each dataset, the MLP and RBF accuracies are in agreement to within the stated limits, with the Linear Discriminant (LD) results being approximately 10% to 15% lower. This demonstrates the improved classification obtained by non-linear techniques with respect to a linear method. It was shown in Chapter 4 that the Meteosat classes are more distinct in feature space than the Bankert classes, and this explains why the linear method performs comparatively better on the Meteosat database.

There are a number of interesting points that arise when considering the confusion matrices on Pages 101 and 102, and these are discussed below in conjunction with the visualization results of Chapter 4. It should be noted that the confusion matrix for the LD is the *average* confusion matrix, whereas those for the MLP and RBF are the confusion matrices corresponding to the *best* accuracy achieved. However, there was little variation in the LD results so a valid comparison can be made.

5.5.1 Bankert database

There is significant misclassification of Cirrostratus as Cirrus, especially for the LD and MLP. The difficulty of separating these two classes was discussed in Chapter 4, and the GTM plot of Figure 4.10(c) shows how the two classes overlap significantly in feature space. The RBF network achieves the best accuracy for Cirrostratus of 76.5%, while the accuracy is only

34.8% for the LD, which demonstrates that linear discriminants are insufficient to separate the classes.

The percentage of Stratocumulus feature vectors correctly classified is also very low for the LD (37.8%). This is explained by the GTM plot of Figure 4.10(b), which shows the Stratocumulus class spread across a wide area of the map. However, the benefit of the MLP and RBF techniques is shown by the large increase in the number of Stratocumulus feature vectors correctly classified, up to 85.7% and 75.8% respectively. The main confusion is with Stratus and Cumulus, which form the triplet discussed in Section 4.5.2 and shown in the GTM plot of Figure 4.10(b). Confusion also occurs between Stratocumulus and Altostratus, especially for the LD. The latter class is also spread out in feature space (see Figure 4.11(d)) which will cause it to be confused with the similarly diffuse Stratocumulus. Interestingly Altostratus and Stratocumulus, the two classes that were shown in Chapter 4 to be more widely spread across feature space, are both dealt with less well by the RBF network than by the MLP. This may be a result of the former network's localized hidden-layer representation (as discussed in Chapter 2) which is not as good at representing the diffuse classes as the MLP.

The low cloud Stratus-Stratocumulus-Cumulus sub-matrices reveal misclassifications between the three classes. This is a result of their relative location in feature space, demonstrated by Figure 4.10(b). The same is true of the high cloud Cirrus-Cirrocumulus-Cirrostratus sub-matrices, with additional confusion between these three and Altostratus and Nimbostratus. Stratus also tends often to be confused with Altostratus.

The number of Nimbostratus samples incorrectly classified as Cirrocumulus is very high (29.4%) for the LD but 0.0% for both the MLP and RBF network. Figure 4.10(a) shows that these two are reasonably distinct clusters, but are very close to each other, with a small amount of overlap. The MLP and RBF are better at separating these closely-spaced classes.

The Cirrus-Cumulonimbus confusion is extremely low for the MLP and RBF. These are both high clouds, although Cirrus can sometimes appear as low cloud as explained before.

The low confusion is probably attributable to the fact that the two classes are reasonably far apart in feature space, as shown in Figure 4.11(b), although there still appears to be some contamination.

All methods are good at separating Clear from other classes, especially the RBF network and the MLP. This is not surprising given that the Clear cluster is shown to be well separated from other classes in Figure 4.9. The Cumulus class also appears reasonably distinct in Figures 4.10(b) and 4.11(c), although there is a lower accuracy for the MLP and RBF than for the Clear class. The main confusion is with Stratocumulus, which contaminates the Cumulus class rather than vice-versa. The Cumulus accuracy is approximately 10% better for the LD than for the MLP, and it is possible that the mapping of the MLP was optimized to distinguish other classes at the expense of Cumulus.

There are small but significant misclassifications between the classes in the Cirrostratus-Nimbostratus-Cumulonimbus triplet. As Figure 4.11(a) shows, the three classes seem to form a group in feature space with overlap occurring between all classes. The Cumulonimbus accuracy is very low for the LD, but the RBF network seems to be the best at distinguishing it. The Cirrocumulus-Cumulonimbus (but not vice-versa) confusion is high for the MLP and RBF network, but low for the LD; these two classes are shown in Figure 4.10(d). The low LD confusion, compared with that of the the RBF network and the MLP, is perhaps due to the effect mentioned above, but may also be due to a fortuitous positioning of the LD decision surfaces.

It was mentioned in Chapter 1 that cirrus is often hard to detect. These tests show that the classification accuracy for the Cirrus class compares favourably with the other accuracies, probably for two reasons. Firstly, all samples are taken over the dark ocean, enabling thin cirrus to be more easily detected. Secondly, it is *possible* that the dataset is biased: cirrus regions that are very difficult to detect may not have been detected and used by the database collators.

5.5.2 Meteosat database

The high overall classification accuracy of 85.0% for the LD method is further evidence of the good separation of the Meteosat classes in feature space, as demonstrated by the results of Chapter 4 and in particular by the GTM plot of Figure 4.8(b). Average accuracies of around 94% for the MLP and RBF network represent very high values, caused partly because there are fewer classes (more broadly defined than for the Bankert database), and partly because of the pureness of samples selected for the database.

Notable confusions are Sea for Land, especially for the LD and MLP, which may be explained by their close proximity in feature space. However, for many applications the classes may be grouped into one “clear” class [76], rendering this confusion irrelevant. Another major source of error is the classification of Deep Convective samples as Dynamic cloud, especially for the RBF. This is a common problem, since the tops of deep convective cloud are identical to cirrus at a local scale [4]. The two neural network models perform significantly better than the linear discriminant method when classifying the feature vectors from the Dynamic class.

5.6 Conclusions

The MLP and RBF networks have been trained to classify test feature vectors drawn from both the Meteosat and Bankert databases. They produce very similar accuracies, which are significantly better than those achieved using simple linear discrimination.

Analysis of the classification results in conjunction with the visualization results of Chapter 4 has provided an extremely detailed picture of the two datasets. Major sources of classification error have been identified and related to the distribution of the feature vectors in feature space. From the GTM plots, it is apparent that some classes, especially in the Bankert database, have a significant amount of overlap. The consistent results from the MLP and RBF network suggest that it will not be possible to separate these problem classes any better *using the current features*; the poor separation of the classes in feature space cannot be overcome. Either the feature set does not provide sufficient discriminatory information,

or some of the samples in the dataset are inappropriately labelled. A further reason for error is the small area that is being classified. The study by Garand [35] shows that even human experts have difficulty in identifying a cloud type from such a small region. The question of scale and context is of paramount importance and is dealt with further in Chapters 6 and 7.

Notwithstanding the above comments, it can be seen that the classification accuracies achieved are high, and would be sufficient for direct application. The study by Pankiewicz [74] used the same Meteosat database as the one used in this study, and resulted in an accuracy of 80.8% which was deemed to be sufficient for use as part of a flood forecasting system at the UK Meteorological Office. This compares with the best accuracy achieved in this thesis of 95.4% with fifteen features, and 92.5% with five features. However it must be stressed that no textural features were used for the Pankiewicz study in order to save processing time (only seven spectral and two temporal features were used), and the Deep and Shallow Convective classes were merged into one.

The Bankert study [5] achieved an accuracy of 79.8% (using what has been called the Bankert database throughout this thesis), while that of Bankert and Aha [6] achieved 86.8%, compared with the best accuracy achieved here of 83.0% for the RBF network. The classification accuracies reported for the Bankert studies were based on the leave-one-out method, whereby all but one of the samples are used for training, with the remaining sample being used for testing. Each sample is held out in turn, to get an overall classification accuracy for the dataset. In this way, effectively 99.9% of the dataset is used for training, whereas only 60% was used for the results of this chapter. It is therefore not surprising that the classification accuracy should be lower in this case, although it is only marginally so. Hence the results presented here compare favourably with related studies reviewed earlier in this chapter.

Chapter 6

Pixel-by-pixel image analysis

Chapter 5 dealt with the use of a neural network to classify individual labelled samples from both the Meteosat and Bankert databases. Features were first extracted for each sample, as set out in Chapter 3, and these features were used as the inputs to the trained MLP and RBF network which assigned the sample to one of the cloud classes. This class could then be compared against the class label attached to the sample in order to assess the performance of the classifier.

This chapter will apply the trained MLP classifier from Chapter 5 to data outside of the labelled sample sets for each database by producing class maps for entire satellite images. Each pixel is classified in turn and coloured according to the output class allocated by the neural network. The class label at a point in an image is highly dependent on that of neighbouring points unless the image is composed of random noise. A precise model for this dependence, the Markov Random Field (MRF), is presented as an alternative to the MLP pixel-by-pixel method, and is described in Section 6.2.

The pixel-by-pixel approach to image analysis is a natural by-product of the feature extraction and classification approach described earlier, and it has previously been described by a number of authors [49, 52, 59, 93, 117]. The limitations of such an approach will be highlighted and its performance will be compared with that of the MRF method. Nevertheless,

pixel-by-pixel analysis is an important precursor to the segmentation results to be presented in Chapter 7.

A three-dimensional Kohonen map will also be used on a pixel-by-pixel basis to colour each pixel according to cloud type. However, as explained in Chapter 4, the Kohonen map relies on an unsupervised learning algorithm, which disregards the class labels during training. The use of a three-dimensional Kohonen map in this way will therefore be termed “unsupervised classification”.

6.1 Pixel-by-pixel analysis with a trained MLP

The results of Chapter 5 show that the MLP and RBF network give similar classification results within the stated experimental limits. The MLP will be used for the results of this section since it is faster than the RBF network for classifying test data.

The trained MLPs from Section 5.3 that gave the highest classification accuracy are used here. The confusion matrices for these MLPs are shown in Tables 5.2 and 5.5 (on Pages 101 and 102) for the Meteosat and Bankert databases respectively. The MLPs had fifteen inputs and fifteen hidden-layer nodes, with all output classes represented (five for the Meteosat database and ten for the Bankert database).

6.1.1 MLP results and discussion

The feature vectors were composed of the first fifteen features chosen as a result of the RBF feature selection method shown in Tables 3.6 and 3.7. Feature vectors were calculated for each pixel in the image and used as inputs to the neural network. As in Chapter 5, the pixel is assigned to the class corresponding to the output node with the highest value (which estimates the posterior probability of that class), according to the Bayes decision rule of Section 2.1. The pixel is then coloured using a different colour for each class. It is not possible to make an objective analysis of the success of the resulting colour images since the “correct” classification for each pixel is not known, but many observations can be made.

Figure 6.1 shows the result of applying pixel-by-pixel classification to an image from the Meteosat database, the same as the example image used in Chapter 1 on Page 10; this image is discussed more fully there in terms of the cloud types that are present. The MLP has classified correctly the large swathe of convective cloud covering the western half of the image, with a band of deep convection showing up in white. It has also picked out the area of deep convection associated with the low in the centre. There is a small patch of clear sea that has been classified incorrectly as Land in the north-west corner, as well as some incorrectly-classified areas of Deep Convection in the south-east corner. The area of clear sea to the west of centre has been correctly classified, as well as a pocket of deep convection to the north-east. The high, smooth cirrus comma cloud just south of centre has also been identified, as has the large swathe of dynamic cloud covering much of the eastern portion of the image.

From the class assigned to each pixel by the MLP, a labelled map has been produced. Each pixel is treated independently of its neighbours, and as a result the map tends to be quite noisy, with local regions or pixels given a particular class against the overwhelming evidence of neighbouring regions. Examples can be seen of regions of two or three pixels diameter being classified as Deep Convection within the swathe of dynamic cloud to the east. There is little evidence to indicate that these areas are significant. Other examples are seen spread all over the map, with noisy boundaries between cloud types. This problem has been identified by a number of authors [52, 71].

A further example is presented in Figure 6.2, for an image from the Bankert database. There are ten output classes, and the confusion matrix for the MLP for this database, shown in Table 5.5 on Page 102, indicates a lower overall classification accuracy than for the Meteosat database. Hence the classification map of Figure 6.2 is noisier than that of Figure 6.1. The image is one of the four NPS images described in Section 1.4.1, and the density of labelled samples is relatively high (see Figure 1.6 on Page 23). There is very good agreement between the classification map and the labelled database samples, but the map is still very broken up,

containing small regions of cloud a few pixels across which have been labelled differently to the surrounding regions. Small areas of sea in the south-east have been classified as Cirrus. The reasons for this possible confusion were discussed in previous chapters, and arise principally because thin cirrus is often hard to distinguish from the surface below it. Close inspection of the infra-red channel suggests that there is perhaps evidence of some thin cloud which does not show up on the visible channel.

Some of the problems associated with the use of a 16×16 pixel box are also evident. There is a ring to the south of the map (in the centre) which has been classified as Cirrus, which is unlikely to be the case. It is probably a manifestation of the fact that at the boundary between two cloud regions, a 16×16 pixel box will contain a mixture of the two classes, and hence the textural and spectral values will be influenced by the characteristics of both cloud types. A consequence may be that the joint characteristics are more like those of another cloud type, resulting in misclassification. A similar effect can occur at the boundary between clear sea and cloud. Clear sea has a very little texture and is uniformly dark. If there are some cloudy pixels at the edge of an otherwise clear 16×16 pixel box, the textural and spectral features will become unbalanced, and the MLP may no longer classify the central pixel as Clear Sea. However, the example of Figure 6.1 shows very good definition of the Clear Sea regions, matching well with the actual boundaries apparent from the satellite image.

A second example from the Meteosat database is shown in Figure 6.3. Unlike Figure 6.1, the entire 640×640 pixel image is considered. The swathe of dynamic cloud across the central portion of the map has been correctly classified (which includes both high and low cloud), with very little contamination of misclassified pixels; there *is* some evidence for the small areas that have been classified as Shallow Convection. The two small areas classified as Land to the north-east and south-east are indeed over land, although the actual regions of clear land are slightly larger. This is due to the boundary effect caused by the 16×16 pixel box, discussed previously. There are many small regions within the deep convective area to the north-west that have been classified as Dynamic cloud. This can be explained with reference

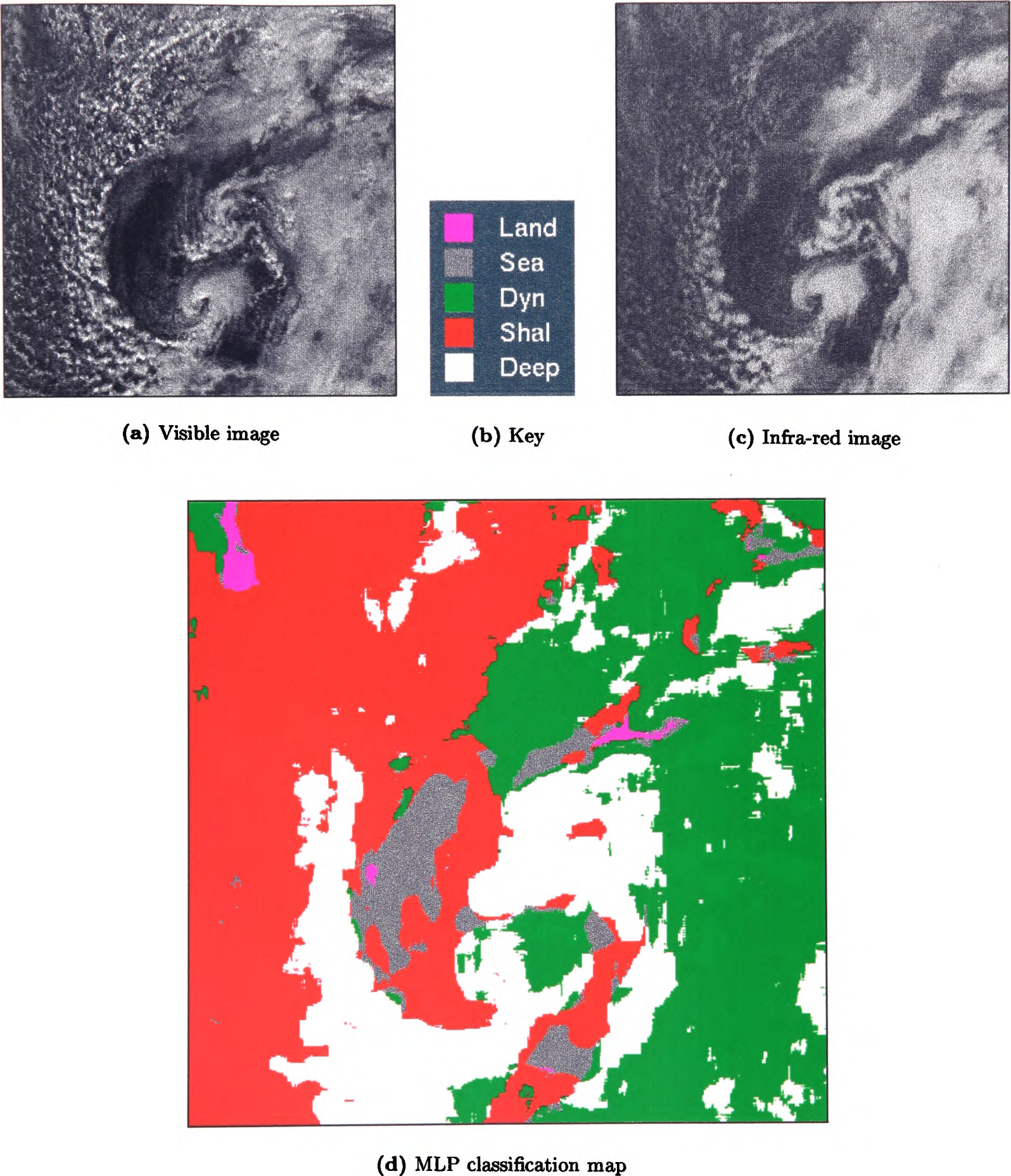


Figure 6.1: First pixel-by-pixel classification by the trained MLP of Chapter 5. Larger versions of the small visible and infra-red images in (a) and (c) are shown on Pages 206 and 207, as well as their details. The corresponding MRF classification is shown in Figure 6.4. The lower-left 512×512 portion of the main image was used.

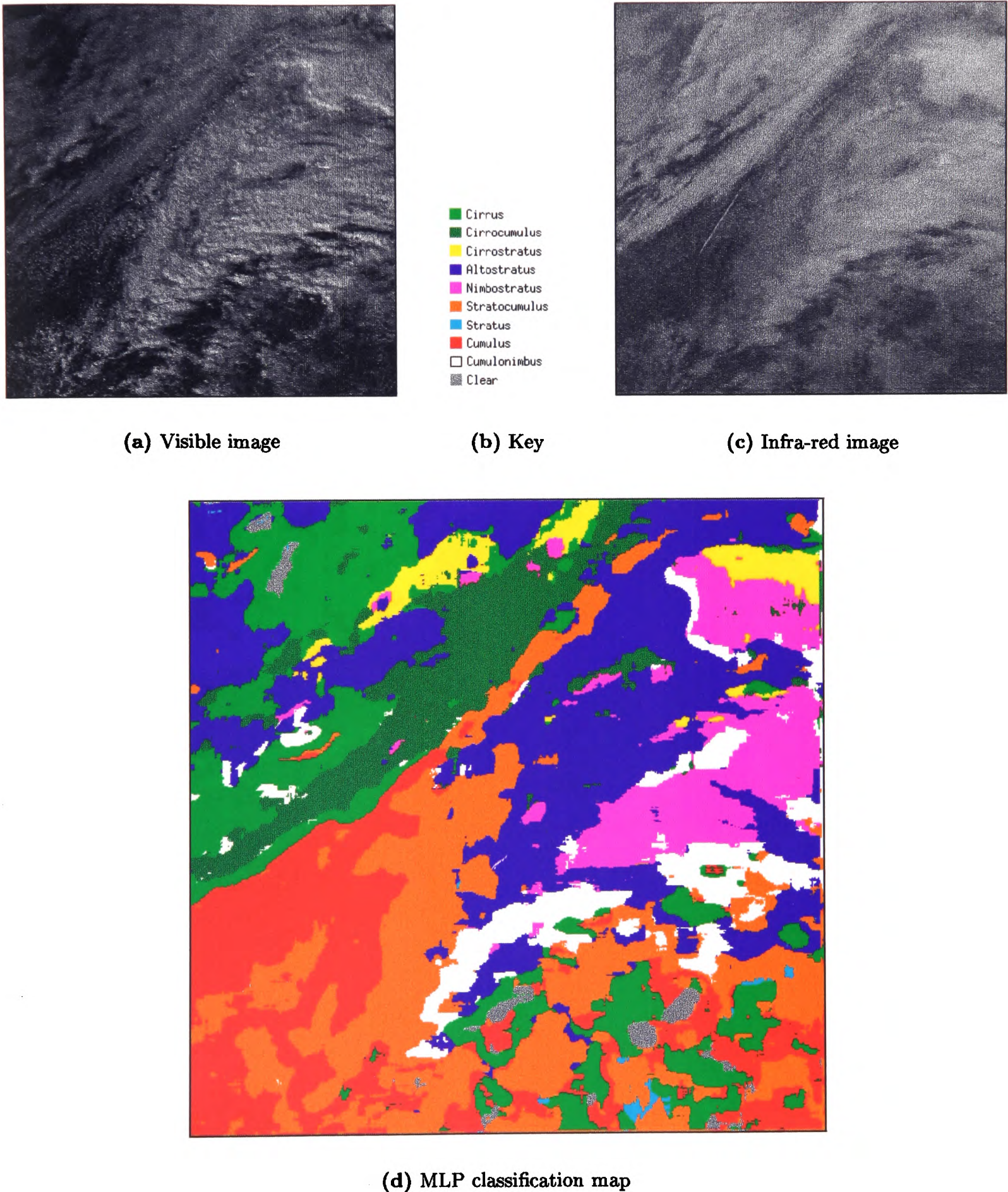


Figure 6.2: Second pixel-by-pixel classification by the trained MLP of Chapter 5. Larger versions of the small visible and infra-red images in (a) and (c) are shown on Pages 214 and 215, as well as their details. The corresponding MRF classification is shown in Figure 6.5.

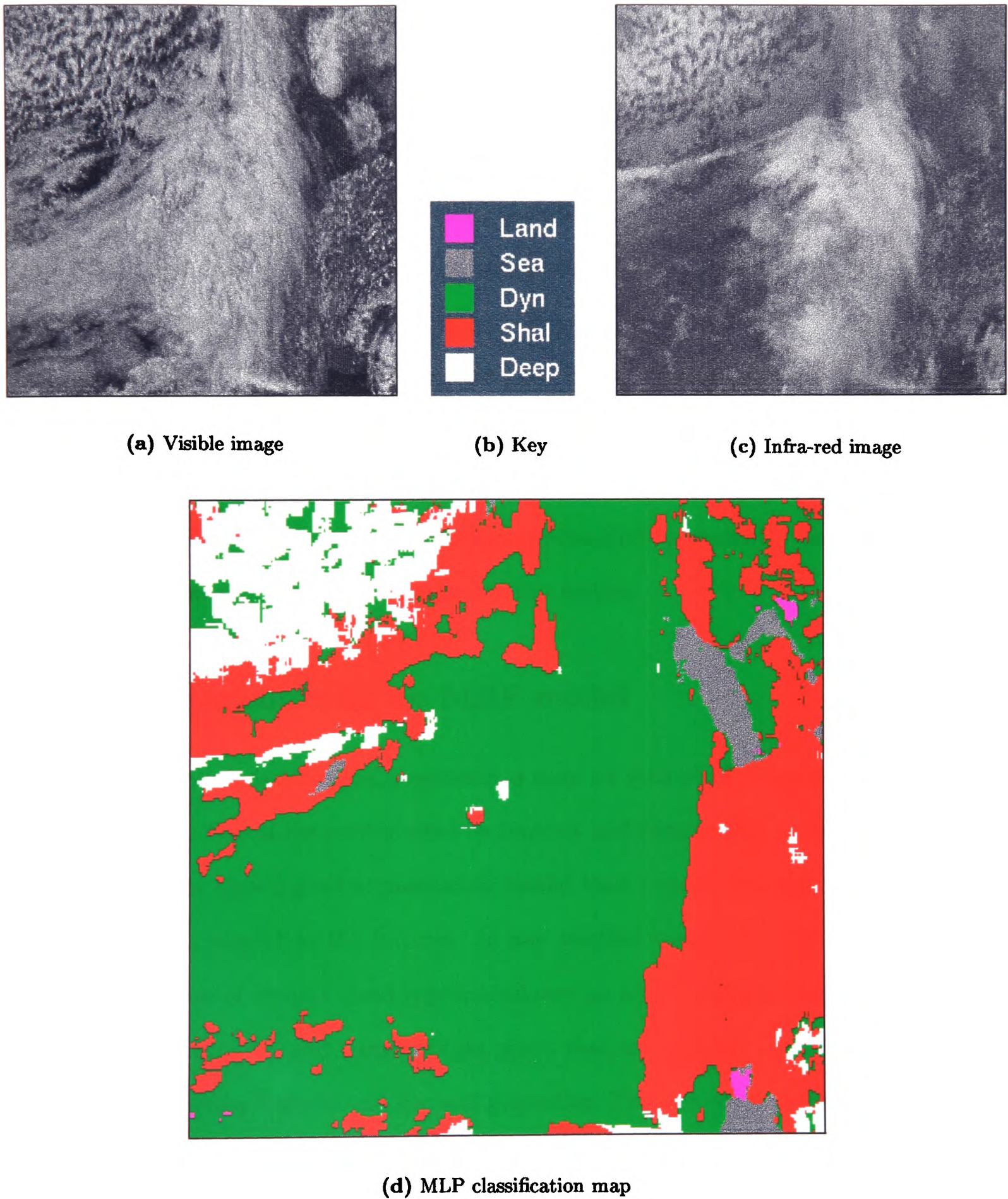


Figure 6.3: Third pixel-by-pixel classification by the trained MLP of Chapter 5. Larger versions of the small visible and infra-red images in (a) and (c) are shown on Pages 208 and 209, as well as their details. The corresponding MRF classification is shown in Figure 6.6.

to the confusion matrix of Figure 5.2 for this MLP, which shows a high misclassification of Deep Convection for Dynamic cloud (since the tops of deep convective cloud are identical to cirrus at a local scale [4]).

The three examples have shown that applying an MLP to each pixel in a satellite image independently can give a very good overview of the cloud types present within the image. The MLP was trained on reasonably pure cloud samples from the databases (see Sections 1.4.1 and 1.4.2), and has been applied to test images from different locations and times of year with good results. The main problem is that of noise, especially at cloud boundaries which are often difficult to locate. In summary, there remains a significant amount of interpretation required of the user, and this should ideally be reduced in an automated way.

The next section will consider a contextual framework for classification, the Markov Random Field, which is introduced as a comparison to the pixel-by-pixel MLP analysis and which will address some of problems encountered in this section.

6.2 Comparison with an MRF model

With pixel-by-pixel classification, processing is done for each pixel independently of all surrounding pixels. One of the criteria cited by Haralick and Shapiro [43], and discussed further in Chapter 7, is that a good segmentation should have regions that are uniform and homogeneous with respect to the features. In any satellite image it is reasonable to assume that these regions of distinct cloud type extend over an area ranging from tens to thousands of kilometres across [4], and therefore that pixels that are spatially close to each other are likely to have similar textural and spectral properties. This notion of context is not used for pixel-by-pixel classification but could provide useful information for image segmentation, to produce smoother, simpler regions.

The use of context is implicit in the theory of MRFs [58], where the label of one pixel is dependent on those of its neighbours via a defined probability distribution. This dependency provides contextual information, which can be controlled via the model parameters. The

theory of MRFs will be briefly summarized in the next section, with reference to relevant previous applications. The results of an MRF model applied to the Meteosat and Bankert image databases will then be presented in Section 6.2.2 and compared with the MLP results of Section 6.1.1.

6.2.1 Basic MRF theory and application

Let X be a lattice, and let the label of a point x_{ij} on X be denoted by l_{ij} where i and j are the coordinates of the point on X . A *colouring* of X is a particular state of *all* the labels of *all* points on X , and is denoted by $\mathbf{L}$.

The meaning of the term *neighbour* in the context of MRF theory is different from its usual sense, and is defined as follows. The point x_{ij} is a neighbour of the point x_{kl} if $p(l_{ij} \mid \text{labels of all points} \neq x_{ij})$ depends on l_{kl} , *i.e.* a point is a neighbour of another if its label is conditionally dependent on that of the other. Note that this does not imply that points x_{ij} and x_{kl} are spatially close to one another, although in practice they usually are.

An MRF is a joint probability density on the set of all possible colourings $\mathbf{L}$ of the lattice X which is subject to the following two conditions [58]:

1. *Positivity*: $p(\mathbf{L}) > 0$ for all $\mathbf{L}$.
2. *Markovianity*: $p(l_{ij} \mid \text{labels of all points} \neq x_{ij}) = p(l_{ij} \mid \text{neighbours of } x_{ij} \text{ only})$.

The Markovianity represents the local characteristics of the model, each pixel only having direct interactions on its neighbours. In order to generate a Markov Random Field, a method must be employed to maximize the joint probability of all the labels, $p(\mathbf{L})$. Use can be made of the MRF-Gibbs equivalence [37] which allows the joint probability to be expressed as a Gibbs energy function, which can then be minimized [37]. Note that the term “label” can apply either a class label in its usual sense, or to a raw pixel grey-level. Therefore the image itself can be modelled as an MRF, as can the class labels attached to each pixel.

The exact formulation of the conditional probability $p(l_{ij} \mid \text{neighbours of } x_{ij})$ affects the characteristics of the MRF that is produced. For instance, Cross and Jain [25] considered a

binomial distribution with parameter $\theta(T)$, where T is an expression dependent on the grey-levels of the pixel neighbours. By manually choosing the values of the model parameters, they were able to generate various textures and blob-like structures. Using a maximum likelihood estimation method, they were able to estimate the parameters of the model in order to simulate various natural textures realistically. Chellapa and Chatterjee [20] demonstrated the usefulness of MRF features to describe texture by using them as the input to a minimum distance classifier to classify seven natural textures (such as grass and wood) to an accuracy of 93% to 99%.

The MRF model has been applied to the analysis and synthesis of cloud images by Garand and Weinman [36], using a structural-stochastic approach. They proposed that a cloud field is best modelled by an overall structural model (using the Fourier transform) which encapsulates the large-scale organization of the cloud, combined with a stochastic model (using an MRF) which captures the local statistical variation of the grey-levels. The probability model used for the MRF was the same binomial distribution used by Cross and Jain [25]. Using this technique, Garand and Weinman [36] were able to generate textures that closely resembled cumuliform cloud fields using only a small number of model parameters. A similar study by Chin *et al* [22] using an autoregressive, moving average (ARMA) process instead of the MRF, but no Fourier structural model, produced similar results. Both methods were also able to capture the anisotropic nature of some cloud fields. The practical size of the neighbourhood of an MRF is typically quite small, usually only one or two pixels distance from another pixel [25]. Hence the larger-scale structure of clouds cannot be adequately modelled using MRFs alone [22].

Kittler and Pairman [52] investigated cloud classification using an MRF contextual model in order to reduce the noise resulting from a pixel-by-pixel classification. A Bayesian maximum likelihood classifier was used (with class-conditional probabilities modelled by single Gaussians) for the initial labelling of each pixel in the image according to cloud type. An iterative process was then applied to clean up the labelled image, using an MRF model to make

each pixel's label dependent on its neighbours. In this way a much less noisy classification was produced.

It is apparent from the studies described so far that an MRF can be used to model either the set of class labels on the image (to provide for context) or the texture of the image formed by the grey-levels of the pixels themselves. It is also possible to use a *doubly*-stochastic model, whereby both the texture *and* the labels are modelled by MRFs, or by autoregressive models. Bouman and Liu [11] used such a doubly-stochastic system for the segmentation of textured images. They also noted the large computational cost of many minimization methods for generating MRFs, and as a consequence used a multiple resolution technique for this purpose, starting the segmentation at a coarse resolution and proceeding to finer resolutions until individual pixels were classified. The results of the segmentation of similar textures were found to be very good. Further work into multi-resolution MRFs has been performed by Lakshmanan and Derin [56] and Krishnamachari and Chellappa [54].

Bouman further developed the multi-resolution approach of the previous paper [11] and develop^d a new multiscale random field model consisting of a series of random fields at varying scales [12]. The standard MRF model has limited ability to describe large-scale behaviour since the spatial interactions are very local. The new model allows the behaviour at various scales to be controlled, and this is of critical importance since scale variation occurs naturally in images. This is the model that is used for this thesis, and the theory is described in detail in Appendix B. The code was written by Bouman, the author of the original paper [12], and adapted to work with the *NetSat* system.

As explained in Section B.2, the model is restricted to having observed pixel grey-levels that are conditionally independent of each other, given all the class labels. This means that the texture itself is not being modelled, which might imply that texture is not used as a discriminating feature. However, the model is designed for multivariate pixel information, which means that feature vectors such as those used for the MLP can be used. In this way, textural information is included in the process. A multivariate Gaussian mixture [8] is used

to model the class-conditional probabilities, with the parameters of the mixture model being set using the Expectation Maximization (EM) algorithm [28]. See Appendix B for further details of the MRF model, and Appendix C for information on the Gaussian mixture model and the EM algorithm.

6.2.2 Results from the multiscale MRF model

The feature vectors were constructed using six features, the first six from the RBF feature selection method shown in Tables 3.6 and 3.7 for the Meteosat and Bankert databases respectively, using ten Gaussians in the mixture model for every class. The mixture parameters were estimated using all the samples in each dataset. Each of the three images used for evaluating pixel-by-pixel classification with the MLP is again used for this section so that a direct comparison can be made. The results are shown in Figures 6.4, 6.5 and 6.6.

There is quite good agreement between the MRF and MLP classifications, especially for the Meteosat examples but less so for the Bankert example. In general, the cloud type of most areas agree for the two methods although of course it is impossible to make an objective comparison. The main difference is apparent for the Bankert examples of Figures 6.2 and 6.5. The Cumulonimbus area on the MRF map is larger than that on the MLP map, while the Altostratus area is smaller. There is also disagreement in the area to the south-east; much of it has been classified as Stratus by the MRF model, while the MLP model assigned it to the Cirrus class. The MRF model has identified the small regions of Clear Sea that were missed by the MLP, although close inspection of the visible channel indicates that the patch of Clear Sea classified by the MRF model to the south-west is larger than it should be. The differences arise from the ways in which the two methods model the data. The MRF model uses a parametric approach to model the class-conditional densities, using a mixture of Gaussians, with parameters estimated from the data using the Expectation Maximization algorithm, as explained in Section B.2. The MLP uses a non-parametric approach to estimate the posterior probabilities directly, as described in Chapter 2.

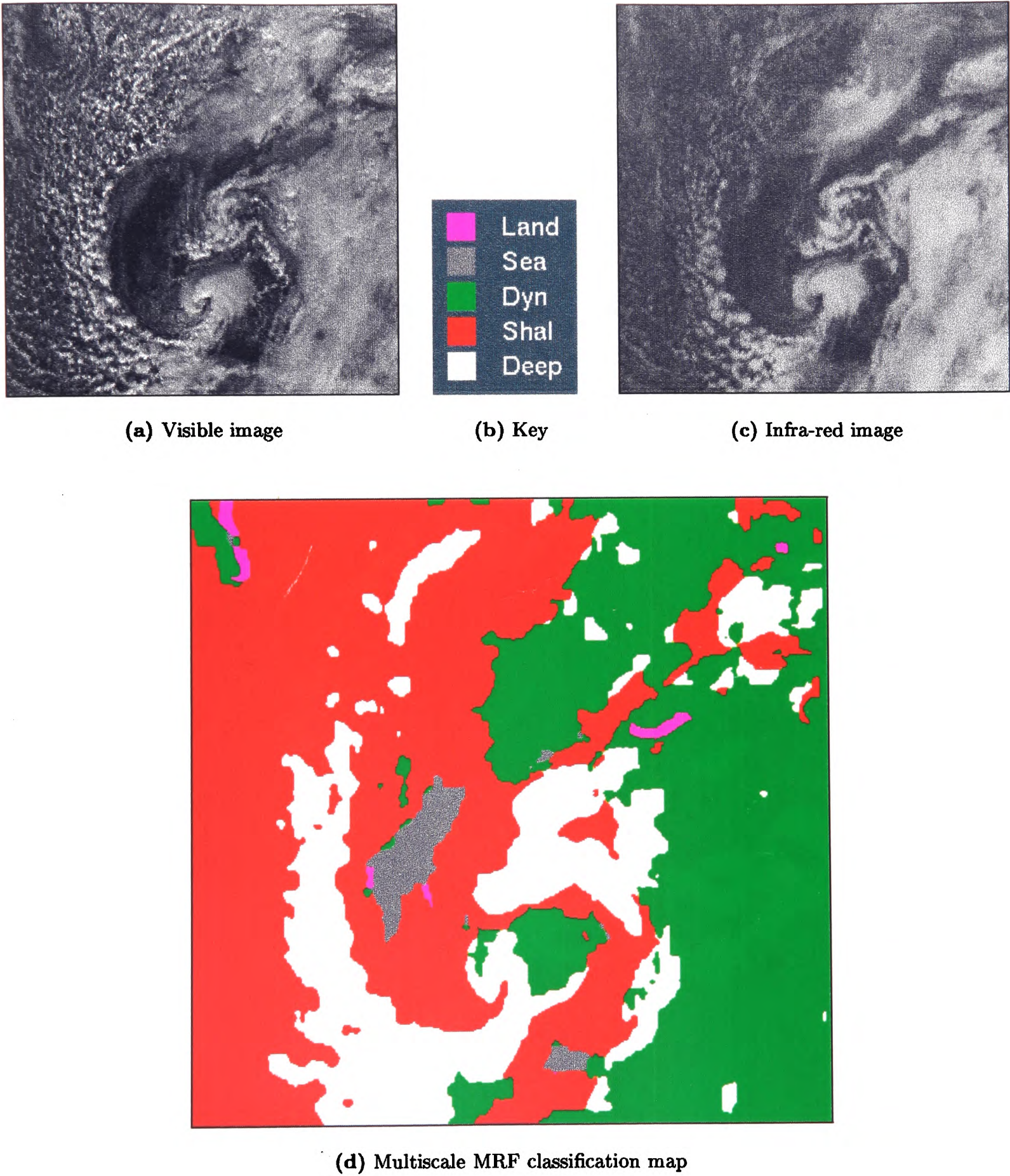


Figure 6.4: First classification by the multiscale MRF model. Larger versions of the small visible and infra-red images in (a) and (c) are shown on Pages 206 and 207, as well as their details. The corresponding MLP classification is shown in Figure 6.1. The lower-left 512×512 portion of the main image was used.

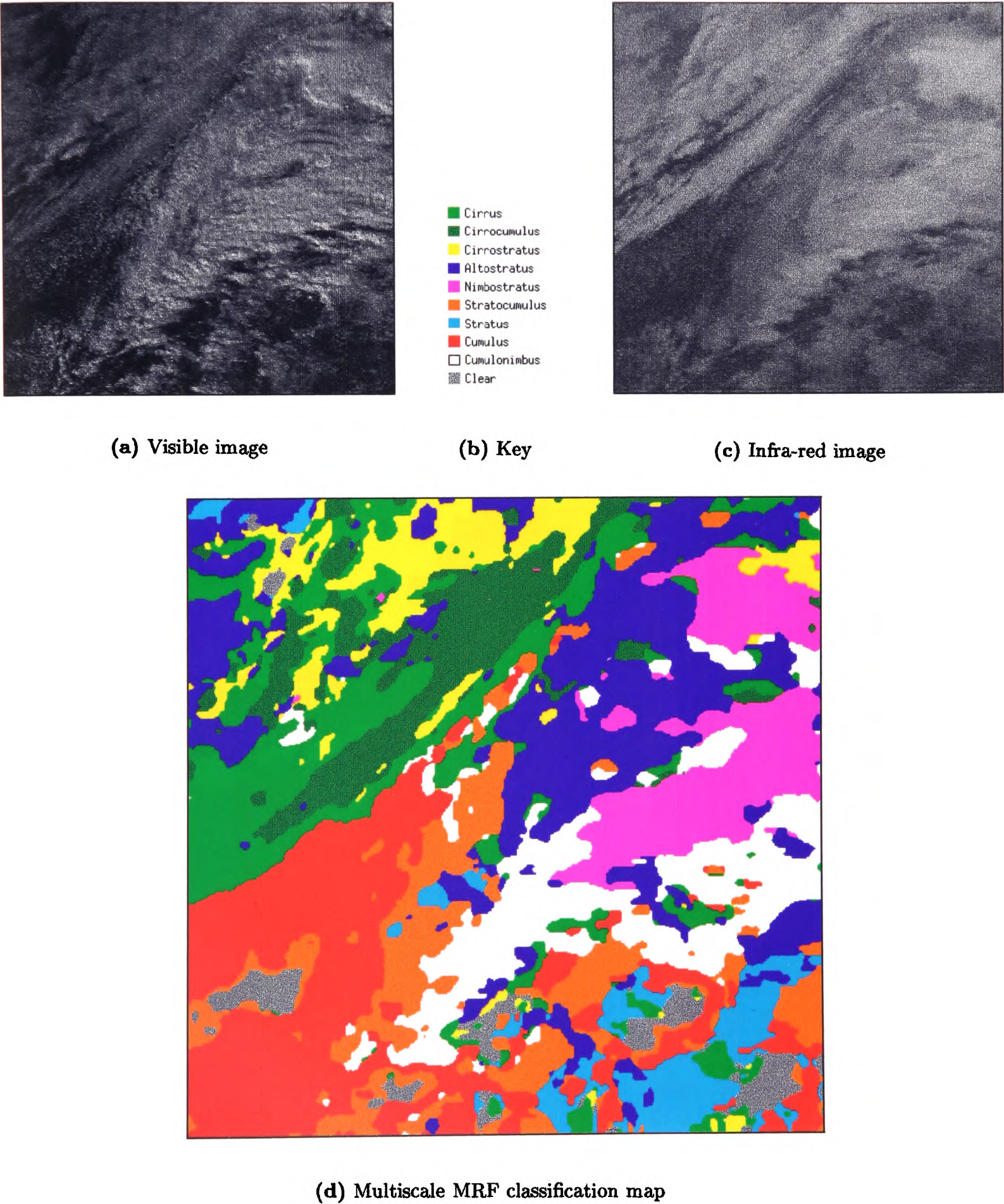


Figure 6.5: Second classification by the multiscale MRF model. Larger versions of the small visible and infra-red images in (a) and (c) are shown on Pages 214 and 215, as well as their details. The corresponding MLP classification is shown in Figure 6.2.

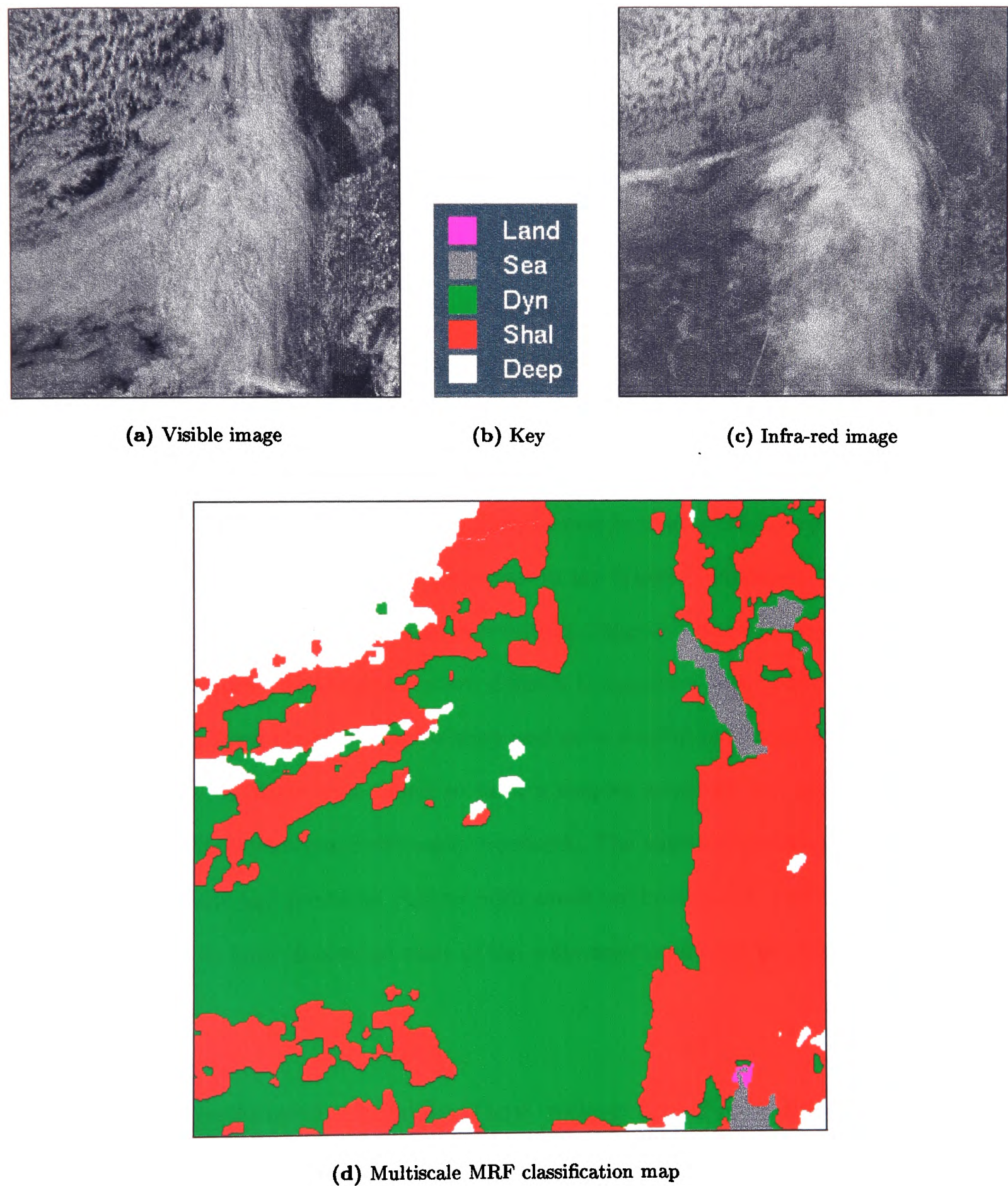


Figure 6.6: Third classification by the multiscale MRF model. Larger versions of the small visible and infra-red images in (a) and (c) are shown on Pages 208 and 209, as well as their details. The corresponding MLP classification is shown in Figure 6.3.

The boundaries between cloud types are smoother for the MRF model than for the MLP classifier. There seem to be fewer small, isolated regions for the two Meteosat examples in Figures 6.4 and 6.6. The area of Deep Convection in the north-west corner of Figure 6.6 shows almost no contamination with small regions classified as Dynamic, which occurred for the MLP analysis shown in Figure 6.3. Likewise, the swathe of Dynamic cloud to the east in Figure 6.4 is better defined than that of Figure 6.1. The MRF performance on the Bankert image is not as good, as shown in Figure 6.5. There are many small, isolated regions, and the overall map is perhaps more confused than for the pixel-by-pixel classification with the MLP. However, the boundaries of these small regions, like those of the larger ones, remain very smooth for the MRF model. This is consistent with results from other images from the Bankert database not shown here.

This suggests that the Gaussian mixture model may be inadequate to represent accurately the class-conditional probabilities for each class in the Bankert database. The visualization results of Chapter 4 and the classification results of Chapter 5 demonstrated that the cloud classes of the Bankert database are more difficult to separate than those of the Meteosat database, with a lower classification accuracy and more overlap in feature space. The classes of the Meteosat database were found to have a simpler structure in feature space, which the mixture of Gaussians can more easily represent. The addition of contextual constraints on the class labels has produced regions with smoother boundaries, and for the Meteosat database seems to have eliminated some of the noise associated with the MLP pixel-by-pixel classification.

6.3 Unsupervised classification using the Kohonen map

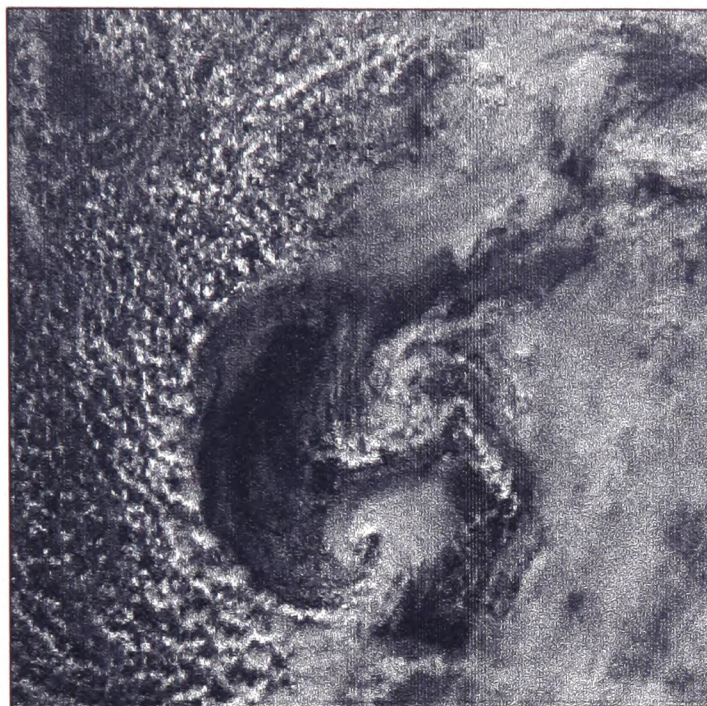
The theory of the two-dimensional Kohonen map presented in Section 4.1 is trivially extended to three dimensions. When a pixel's feature vector is applied to the trained Kohonen map, the winning unit in the three-dimensional structure can be selected. The coordinates of this unit within the three-dimensional structure can be used as an RGB triplet and plotted in

the resulting colour, each coordinate being scaled from 0 to 255. In this way, every pixel in the image can be coloured according to which unit in the Kohonen map its feature vector is closest. Similar cloud types, with similar spectral and textural properties, will be mapped to a similar colour (since they are close in the Kohonen map), and conversely different cloud types will be mapped to different colours. No human labels are used at any stage for the production of these classification maps, and there is no pre-determined colour scheme; here the colours are determined solely from the position of the feature vector in feature space.

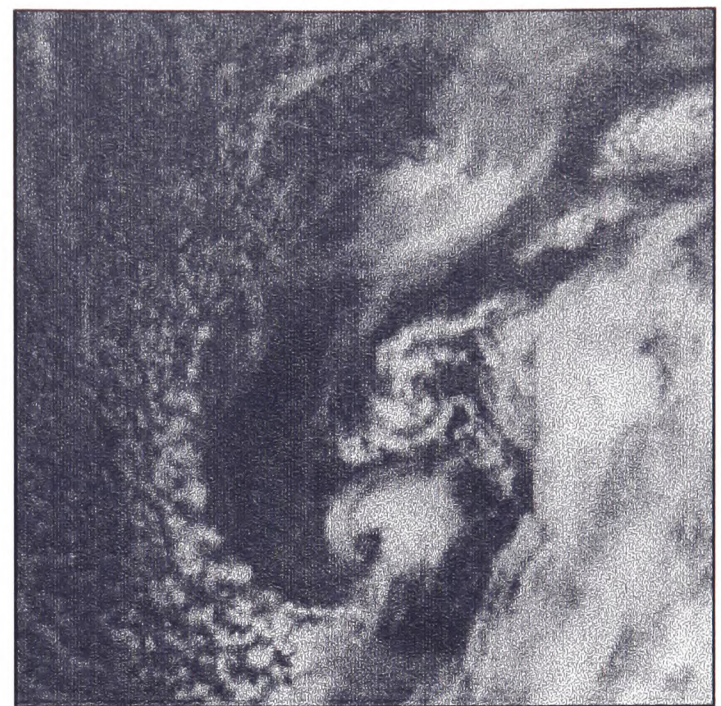
A $4 \times 4 \times 4$ Kohonen map was trained on all samples in the Meteosat database, using the same six features as for the GTM examples of Section 4.5, with 30 cycles through the entire dataset for training. The initial neighbourhood radius was given a value of two, the final radius had a value of one, and the initial learning rate was 0.8. Weights were initialized in the range ± 0.2 . The resulting colour map for an example image pair is shown in Figure 6.7. The same trained Kohonen network was then applied to a different image, with the result shown in Figure 6.8. Note that the colours bear no relation to those which were chosen for the visualization results shown in Chapter 4. The colour assignment process is totally unsupervised, as the colours are being determined by the mapping of the Kohonen network. It can be seen that different cloud types have been coloured differently, with shallow convective cloud distinct from dynamic cloud and clear sea.

Different Kohonen maps will lead to different colourings. An $8 \times 8 \times 8$ map was trained using 100 cycles through the same dataset as before, with a starting neighbourhood radius of seven, a final radius of one, and weights initialized in the range ± 0.2 . Two example images have been coloured using this network, shown in Figures 6.9 and 6.10. Since colour assignment is unsupervised, different colours are given to the same pixel positions in Figures 6.7 and 6.9. However, allowing for the different colour maps, the two are in good agreement. Since the $8 \times 8 \times 8$ lattice has more nodes available than the $4 \times 4 \times 4$ lattice (and hence more colours), there is a smoother graduation of colour changes in Figure 6.9.

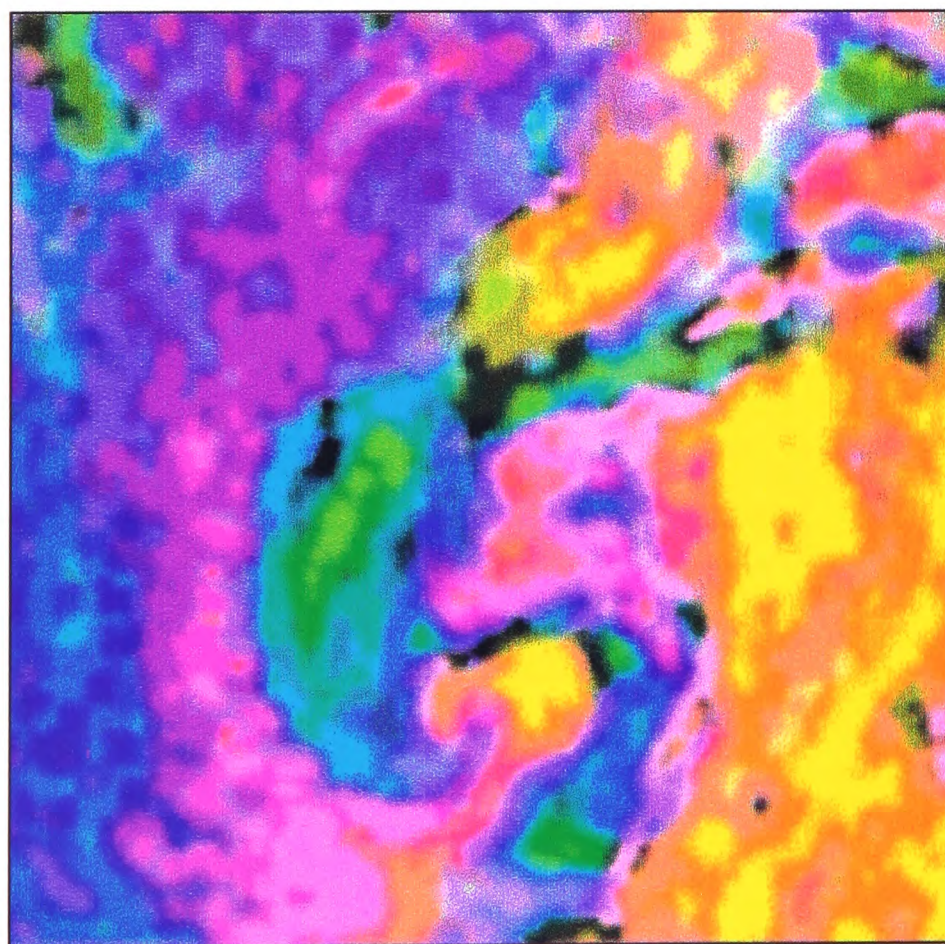
The unsupervised map of Figure 6.7 can also be compared with the MLP classification



(a) Visible image

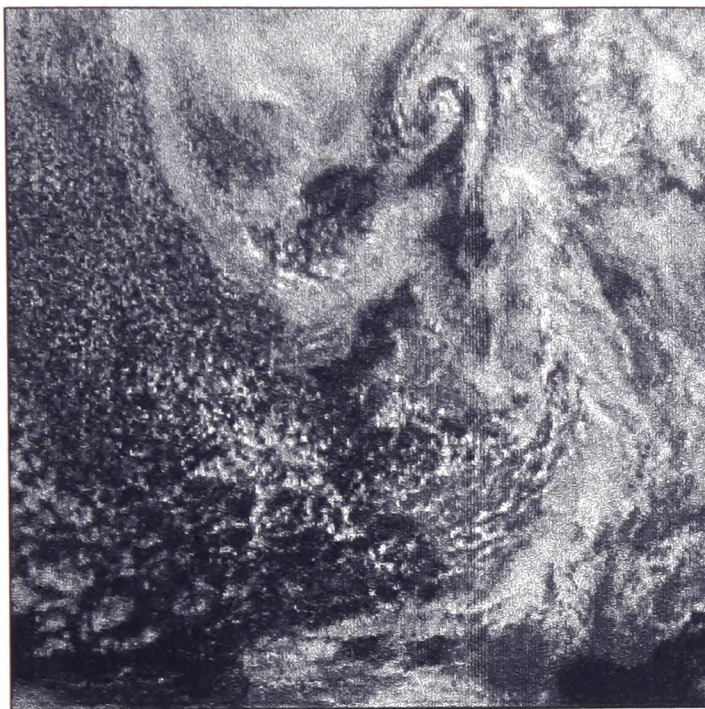


(b) Infra-red image

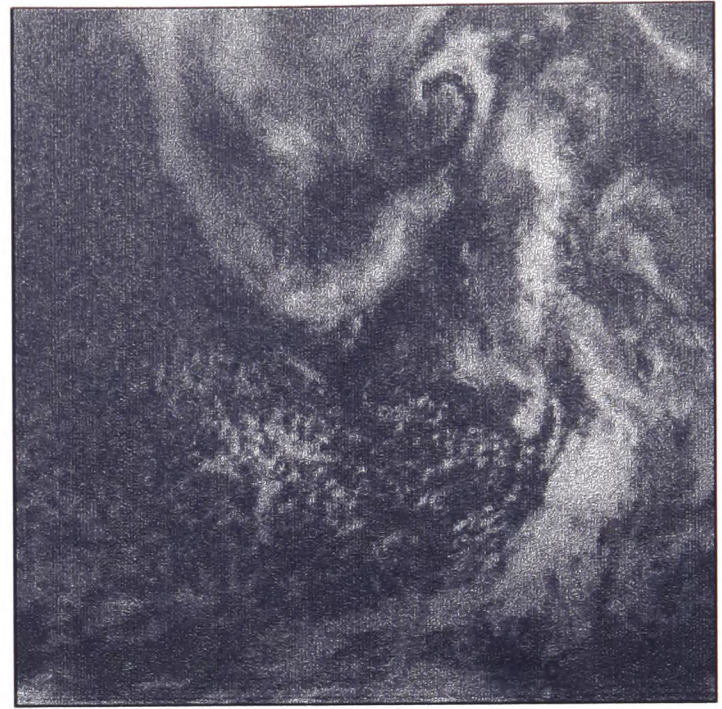


(c) Unsupervised classification map

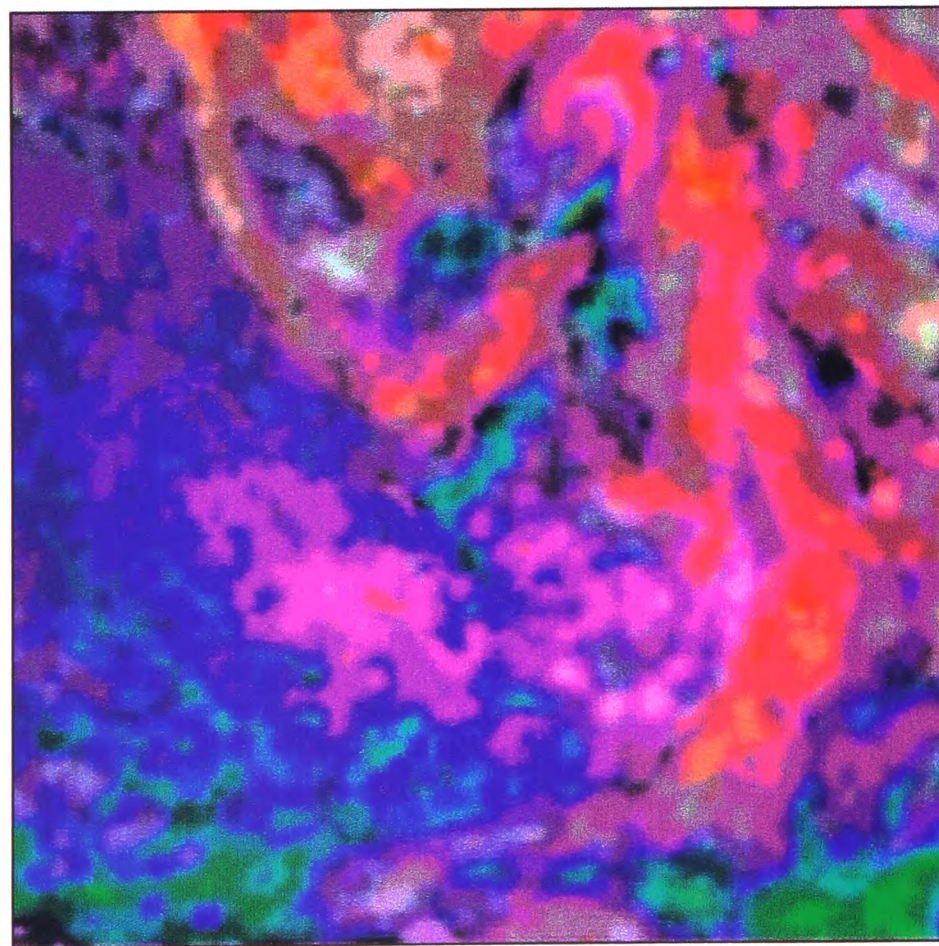
Figure 6.7: First unsupervised classification map from run one. The same trained $4 \times 4 \times 4$ Kohonen map as that for Figure 6.8 was used. Larger versions of the small visible and infra-red images in (a) and (b) are shown on Pages 206 and 207, as well as their details. The lower-left 512×512 pixel portion of the main image was used.



(a) Visible image



(b) Infra-red image



(c) Unsupervised classification map

Figure 6.8: Second unsupervised classification map from run one. The same trained $4 \times 4 \times 4$ Kohonen map as that for Figure 6.7 was used. Larger versions of the small visible and infra-red images in (a) and (b) are shown on Pages 212 and 213, as well as their details.

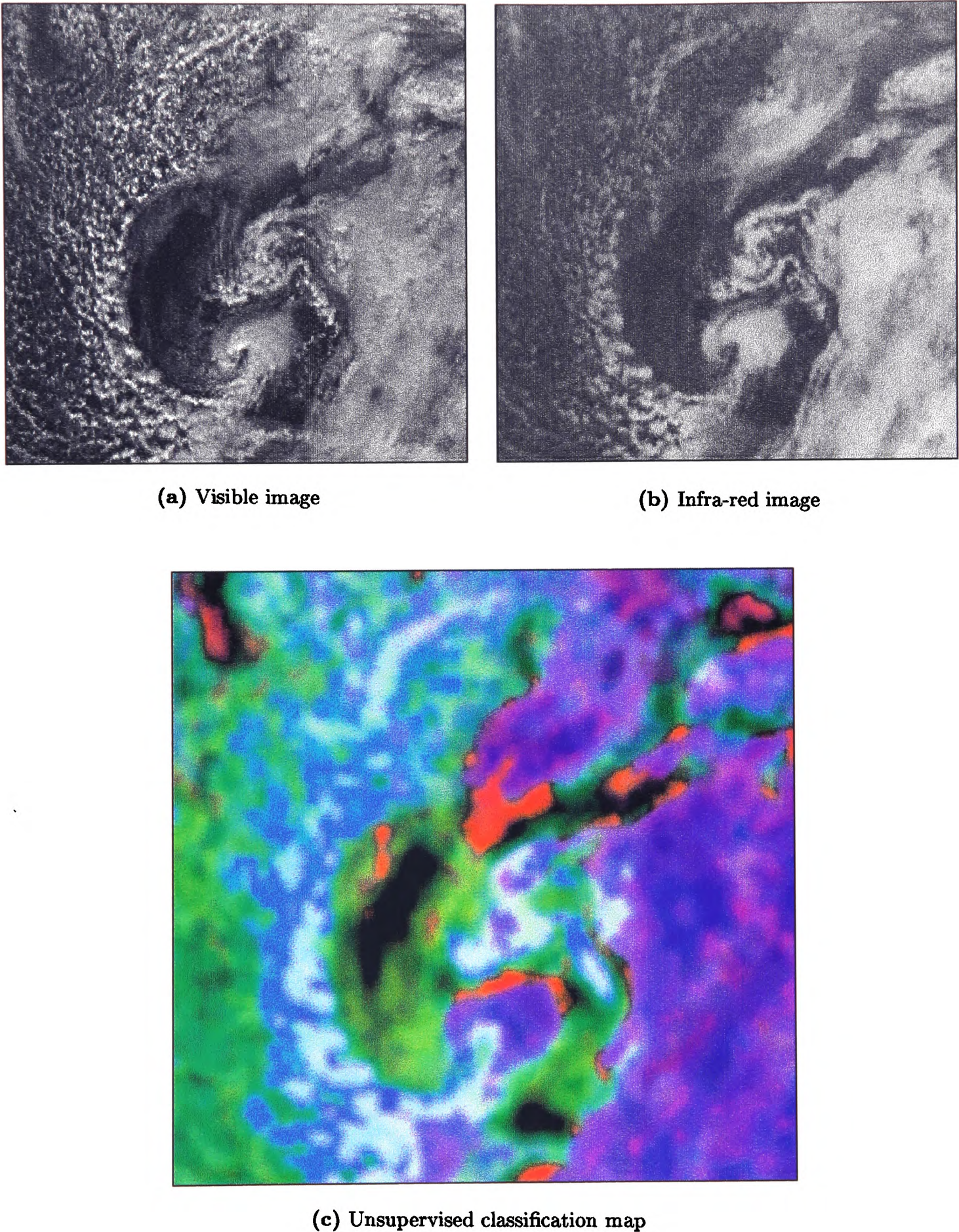
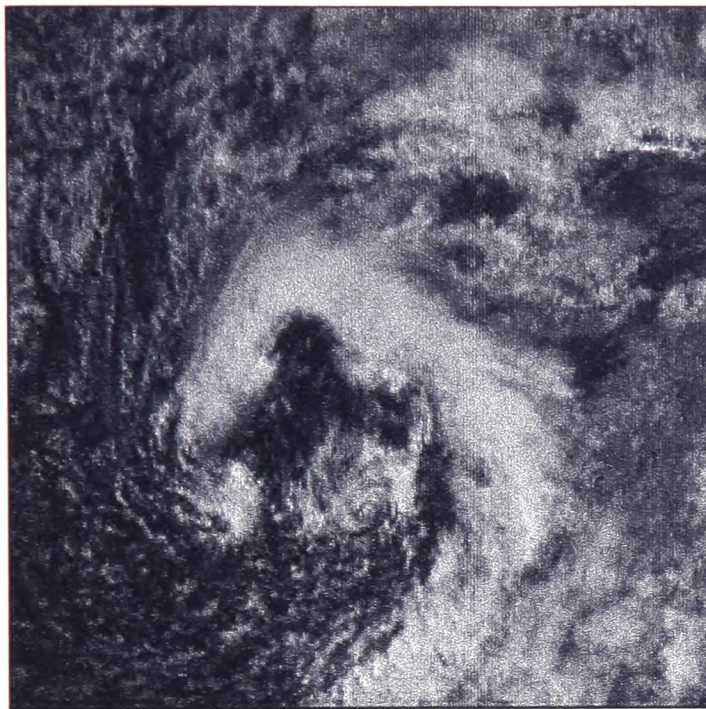
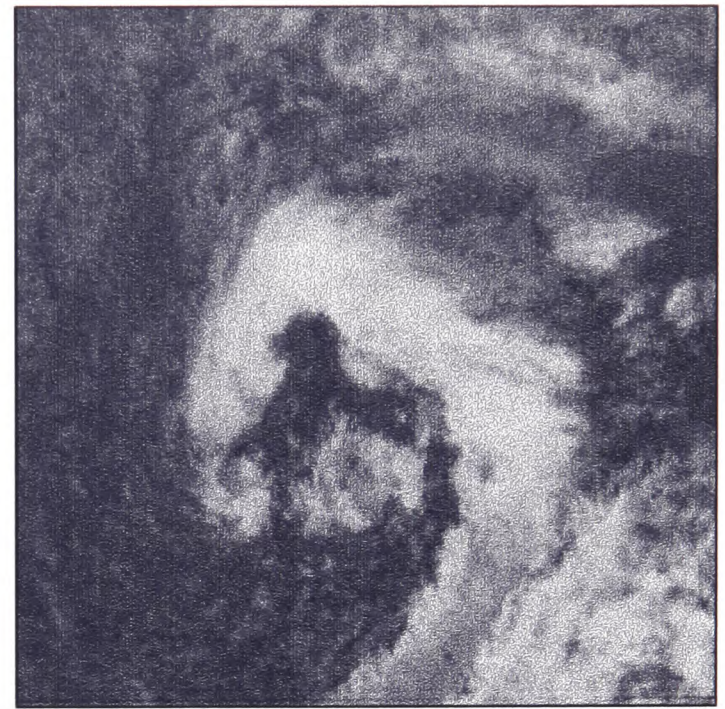


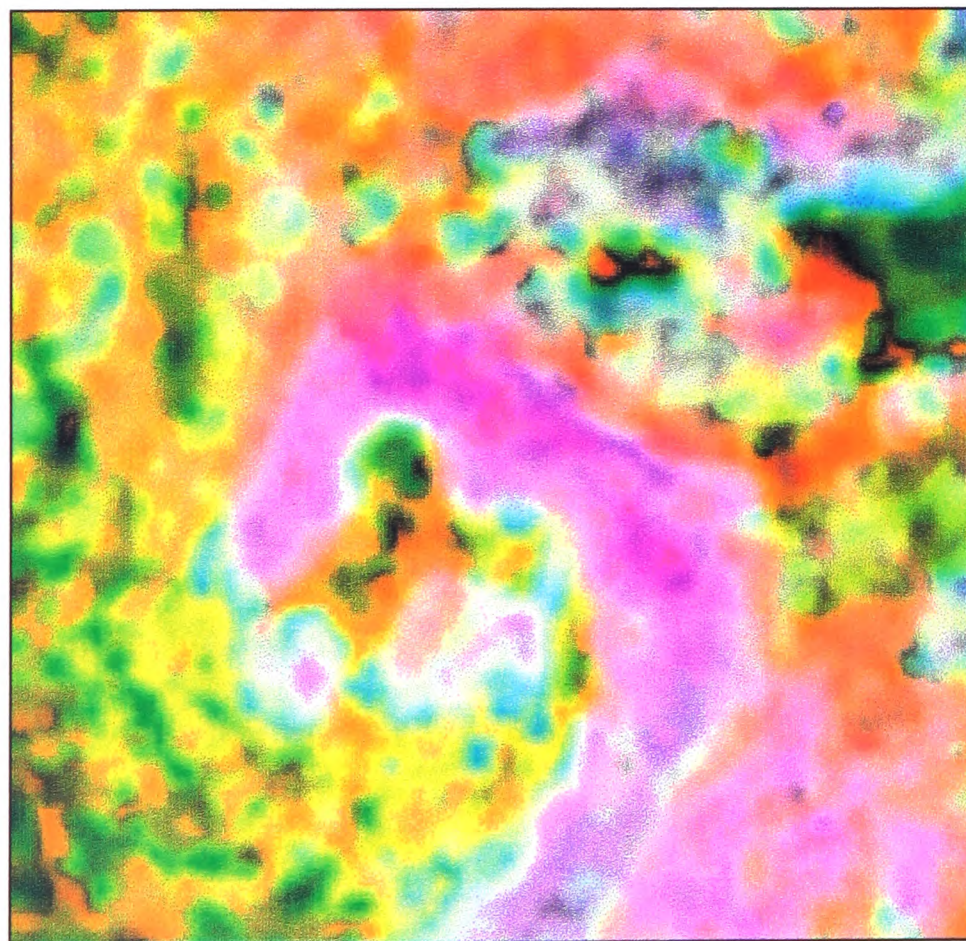
Figure 6.9: First unsupervised classification map from run two. The same trained $8 \times 8 \times 8$ Kohonen map as that for Figure 6.10 was used. Larger versions of the small visible and infra-red images in (a) and (b) are shown on Pages 206 and 207, as well as their details. The lower-left 512×512 pixel portion of the main image was used.



(a) Visible image



(b) Infra-red image



(c) Unsupervised classification map

Figure 6.10: Second unsupervised classification map from run two. The same trained $8 \times 8 \times 8$ Kohonen map as that for Figure 6.9 was used. Larger versions of the small visible and infra-red images in (a) and (b) are shown on Pages 210 and 211, as well as their details. The lower-left 512×512 pixel portion of the main image was used.

map of Figure 6.1. Generally, there is very good agreement between the two on the extent of the coloured regions in each map, indicating the validity of the unsupervised classification approach. No human labels are used at any stage for the production of these images. The colour is not chosen according to a pre-determined class colour scheme as it was for the MLP and MRF classifications; it is determined solely from the position of the feature vector in feature space. In this way, an entire image can be visualized instantly, incorporating both textural and spectral information from each spectral channel into one coloured image, with no subjective class labels (and potential errors) being introduced at any stage of the process.

6.4 Conclusions

This chapter has demonstrated the results of applying an MLP classifier independently to every pixel in a satellite image. General agreement has been found between the cloud types apparent in the visible and infra-red images which can be identified by a trained observer, and the output classifications produced by the MLP. Errors associated with any classification process, which were quantified for the MLP in Chapter 5, result in small regions of inappropriately-labelled pixels. The assignment of a cloud class on the basis of a small region removed from the context of the surrounding image is a difficult task even for human experts [24, 35].

Noisy or inappropriately-labelled pixels can be broadly attributed to three factors. Firstly, the MLP may classify the pixel wrongly (*i.e.* differently from the label given by the human expert). Secondly, the 16×16 pixel box may be located over the boundary between two cloud types, both of which are known with certainty, which results in an inappropriate feature vector for that box, and hence a misclassification. Finally, the precise cloud class may not be obvious, even to human experts. The cloud naming scheme used for the Bankert database is an artificial creation which attempts to group clouds according to how they appear. Very often, clouds do not display characteristics specific to any designated class, or may exhibit properties of two or more classes. The Meteosat database used classes related more to the physical

processes involved in the formation of the cloud and are therefore somewhat less arbitrary. However, dynamic and convective activity can occur together, so the same ambiguity occurs.

These points emphasize the benefits of the unsupervised classification maps produced in Section 6.3. Each pixel is coloured not according to some arbitrary classification scheme, but is controlled directly by the textural and spectral properties of the cloud itself. Traditional classification of a cloud image requires the introduction of subjective human labelling, and errors are introduced. The totally unsupervised colouring of a cloud image using the three-dimensional Kohonen map uses no such subjective labels at any stage. In this way an image is “classified” into different colours according to cloud type, without human labels.

Humans make use of context when attempting to interpret confusing information. An estimate of the cloud class in a local region may be supported or contradicted by considering information from the neighbouring area, or by looking at the synoptic-scale systems of which the region is a part [71]. The multiscale MRF model presented in this chapter has been shown to make use of local context in order to label each pixel in the image with spatial constraints. This leads to smoother regions with less noisy boundaries, one of the criteria set out by Haralick and Shapiro [43]. It would be possible to post-process the MLP classification maps to filter out very small regions. However this is not ideal because the post-processing takes place after all the primary information has been considered.

The multiscale MRF model was found to perform less well on the Bankert database images, producing many small, albeit smooth, regions. This is probably due to the inadequacy of the Gaussian mixture model to model the data in feature space accurately, as compared with the MLP. Results also show that the MRF models can only incorporate context at a relatively local level. However, for the Meteosat database the multiscale MRF model performed well, producing smoother regions than the MLP with less noise. The question of scale and context will be addressed again in Chapter 7, which will deal with large-scale segmentation of satellite images into distinct regions of cloud type, treating cloud fields on a global rather than local level.

Chapter 7

Large-scale segmentation

There has been much research in recent years that aims to make much fuller use of the meteorological information contained in satellite imagery for automated applications [73, 75]. As has been seen in previous chapters, many of the studies have been concerned with the identification of cloud type for a very localized region around each image pixel, typically a 16×16 pixel box. The information content in such a small region has been shown in Chapter 5 to be sufficient to achieve cloud-type classification accuracies ranging from 80% to 95%, depending on the cloud classes and training method used [5, 71]. If every pixel in an image is classified individually, as was done in Chapter 6, a coloured map of the estimated cloud classes can be produced.

There are two main limitations to such a pixel-by-pixel approach. Firstly, the classification image tends to be very noisy due to local regions of misclassified pixels. Secondly, the product has limited practical application since there is no concept implicit in the method that defines a *region* of cloud (cloud field) or that relates the information to current conceptual models based on synoptic cloud features such as commas or vortices. It is the aim of the work presented in this chapter to perform a large-scale segmentation of the image such that it can be described in terms of a small number of regions rather than a large number of independently-classified pixels.

7.1 Overview and previous work

As seen in Section 1.3, a major goal in meteorological satellite image pattern recognition research is the accurate identification of synoptic-scale cloud systems for incorporation into numerical weather prediction (NWP) models [78]. Peak and Tag [82] proposed a “bottom-up” approach to identifying cloud systems. Processing was done first on a pixel-by-pixel basis, identifying local cloud information across the image. This information was then used to calculate six properties over a synoptic-scale region: the amount of low-, high- and multi-layer cloud in the region, the longitudinal and latitudinal extent of the region, and its northernmost latitude. It should be emphasized that all of the properties, as well as the region itself, were manually defined. This information was then used as the input to an MLP classifier in order to assign the region to one of eight synoptic systems (including tropical cyclones, frontal bands and troughs). An accuracy of 83% was reported.

The main problem identified in the Peak and Tag study was the provision of an automated rather than manual method for defining meaningful and significant regions that are used in the procedure. As a feasibility study they considered a region-merging method [67] based on the grey-levels of the image. Serious difficulties were encountered in producing meaningful regions, which tended to have very ragged edges and contain large holes. The bottom-up approach is also proposed by Pankiewicz [71]; he considers a three-level system [72, 73] whereby simple cloud classes are first defined locally, followed by a broad segmentation of the image. Textural properties are then calculated over each region (avoiding the problem of calculating texture over boxes of a fixed size), and used as input to a second classifier together with the class from the first stage. A final classifier would then attempt to allocate each region to a specific synoptic-scale system. Only the first stage was dealt with in the paper [72] (see Section 1.3 for further information on the work by Pankiewicz).

A subsequent study by Peak and Tag [83] tried to address the problem of segmentation first and foremost, using a new technique called Hierarchical Threshold Segmentation. In this method, a grey-scale threshold is applied to the image at a number of levels, and a neural

network is used to combine these threshold images to produce regions of an appropriate size. In giving more importance to the segmentation stage, the overall synoptic feature identification process was different to that of the previous Peak and Tag paper. Processing is done first on a large scale (threshold segmentation), and it is the properties of these regions that are considered as inputs to the final classification MLP as before. Hence the approach is now top-down rather than bottom-up.

Region-growing techniques applied to cloud segmentation have also been considered by Chou *et al* [23], using a variant on the region-merging technique, termed *region-oriented* segmentation. In this method, regions are grown one at a time, rather than formed simultaneously through merging of adjacent regions. A region is grown into an unused area dependent on similarity criteria which include the textural properties of the local area (using GLDV statistics – see Chapter 3). The results were found to provide more uniform regions with smoother edges than the pixel-by-pixel techniques.

An extension of the Canny edge-detection algorithm [15] to multi-spectral images has been considered by Drewniok [31], and applied to Landsat images (see Appendix A). It can be used to find edges based on the spectral information in all bands. However it is better suited to land-type segmentation where the boundaries are more edge-like than for cloud images. Also, like the Canny edge-detector, it produces a set of edges that are not necessarily joined together to define enclosed regions.

The importance of context and scale has been identified by a number of authors [52, 71, 73], both in terms of improving pixel-by-pixel accuracy by incorporating more contextual information and as a means of identifying larger-scale structure in the image. This was discussed with regard to the MRF model described in Section 6.2. There is also a large body of research work on the segmentation of textured images, and methods for the accurate location of the boundary between different textures. The study by Yhann and Young [118] used a multi-resolution approach that combines localized and distributed features directly to address boundary localization in texture segmentation. The algorithm was applied to

AVHRR satellite imagery with some success.

The research presented in this chapter aims to use the information gained from local-level processing in order to perform a secondary segmentation process which will define significant regions of the image, for instance to demarcate regions of deep convection or dynamic cloud. In this way, regions of a certain cloud type with lateral extents of the order of 1000 km can be identified, rather than localized regions of the order of 10 km. It would then be possible for a further stage to analyse the shape and positioning of these regions in order to relate them to conceptual models [73, 82] (although this is not dealt with here). Haralick and Shapiro [43] suggest that a segmentation should have the following properties: (a) regions of a segmented image should be uniform and homogeneous with respect to the features; (b) adjacent regions should have significantly different features; (c) regions should be simple and without too many holes; and (d) the boundaries of each region should be simple, not ragged, and be spatially accurate. These four criteria must be taken into account when considering any segmentation algorithm.

7.2 Method of posterior images

The boundaries between different cloud types in a satellite image are often not very easy to distinguish. There are usually no obvious changes in spectral intensity in any single spectral band that can be used accurately to determine region boundaries. Rather it is a combination of spectral *and* textural changes in *all* bands that serve to mark changes between one cloud type and another. The MLP classifier has been shown in Chapter 5 to be capable of distilling all this information to produce classification accuracies between 80% and 95%, which is sufficiently accurate to be used as the input to a further stage. The output of the MLP will be used to produce a “posterior image” for each cloud type, and it is this image that is segmented.

Figure 7.1 gives an overview of the process that is common to both methods proposed in this chapter. A feature vector $\mathbf{x}$ is calculated for each pixel in the image (as explained in

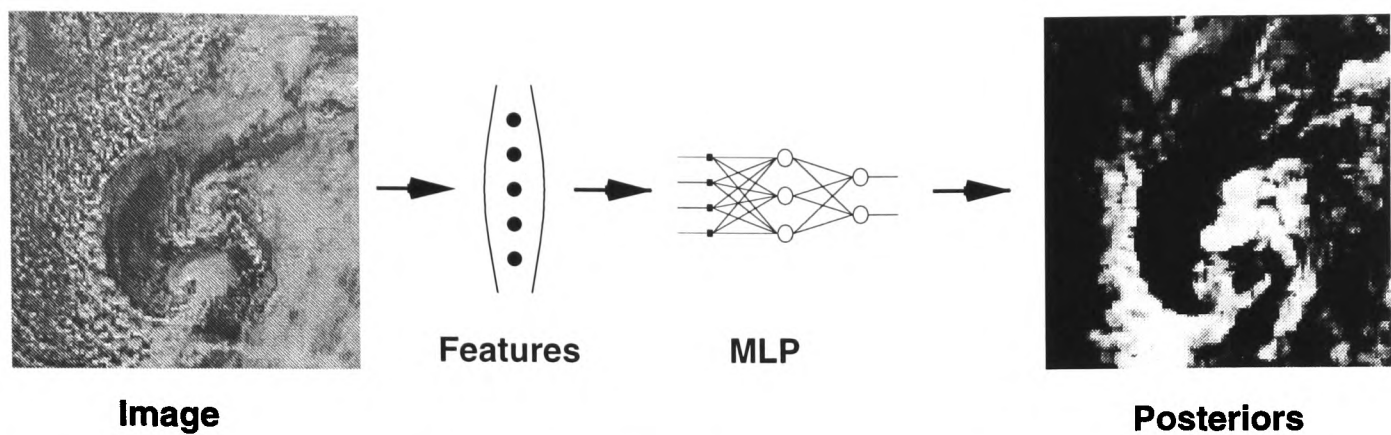


Figure 7.1: The process from original image to posterior image. Posterior probabilities are calculated for every pixel by an MLP, shown here for the Deep Convection class. See text for details. The image shown is from the visible image shown in Figure D.1 – see Page 206 for details.

Chapter 3), encoding as much spectral and textural information as possible. The vector $\mathbf{x}$ is the input to a trained MLP classifier which estimates the posterior probabilities $P(C_i | \mathbf{x})$, one for each cloud type under investigation. A plot of $P(C_i | \mathbf{x})$ for every pixel in the image gives a “posterior image” for each class, such as the one in Figure 7.1 which is for the Deep Convection class of the Meteosat database. It gives a map of the probability that the local area contains deep convective cloud. The next stage in the process is the demarcation of areas of significant probability within each posterior image, *i.e.* significant regions of each cloud type. This is done by locating the boundaries between the high and low posterior probability regions.

Two methods for finding salient contours in the posterior image will be used and compared: firstly the active contour model, or “snakes”, presented by Kass *et al* [48], and secondly the level set method developed by Malladi *et al* [64]. They are conceptually related to each other, producing a similar output but using different methods. Both consider an interface between two regions which is made to move across the image to locate the appropriate boundary, with smoothness constraints on the boundary in order to satisfy the smooth boundary criterion cited by Haralick and Shapiro [43].

7.3 The active contour model

This method uses a deformable spline which is under the influence of internal (bending and stretching) forces as well as external image forces that serve to push the spline towards salient image features such as edges. Representing the position of the snake parametrically by $\mathbf{v}(s) = \{x(s), y(s)\}$, the following energy function is defined:

$$E_{\text{snake}}^* = \int_0^1 E_{\text{snake}}(\mathbf{v}(s)) ds = \int_0^1 E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s)) ds \quad (7.1)$$

where E_{image} is the energy due to image forces acting on the snake at every point, and E_{int} is the internal energy due to bending and stretching. In practice the snake is composed of a number of nodes connected by straight lines. The smoothness of the snake shape is controlled by the bending energy – a high energy associated with bending will produce a smoother segmentation. By minimizing this energy function, the snake is attracted to salient image features. Details of the numerical procedure for this minimization are found in Section C.4 of Appendix C.

The external image force must be formulated so that the snake will seek out the borders of the posterior image regions. The following simple scheme is used: if a certain snake node associated with class i lies on a pixel that has a posterior probability of greater than 0.5 for class i , then act on that node with a force perpendicularly outwards from the snake. Otherwise act with a force inwards (see Figure 7.2). In this way the snake will move to surround the region of significant posterior probability.

An alternative image force was tried that uses the classification given by the neural network at the pixel on which the snake node lies. If the class given by the classifier is i , for a snake associated also with class i , then act on that node with a force perpendicularly outwards from the snake. Otherwise act with a force inwards. This method was found to give very similar final results to the one described above, and so will not be discussed further.

Minimization of the energy function given by Equation 7.1 is achieved by an iterative

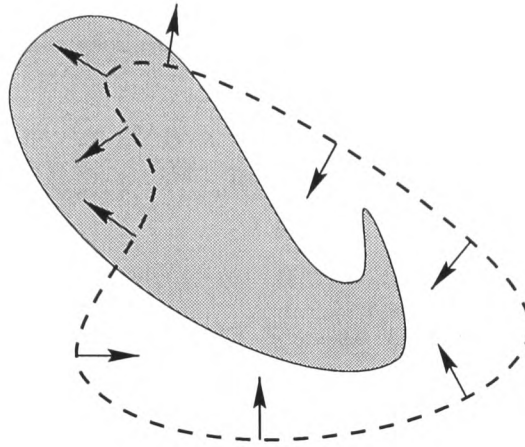


Figure 7.2: External image forces on the snake. The shaded region denotes the area with posterior probability greater than 0.5

scheme presented in Section C.4 of Appendix C, which causes the snake to move across the image at each step in the iteration. The initialization of the snakes is an important consideration. The active contour model method does not address this problem as it is left to the user to position manually the snakes in an appropriate place close to the desired features [48]. The following section describes a new Gaussian mixture model method which has been developed in this thesis to model the posterior probability maps, and which will provide a method for seeding the snakes.

7.3.1 Gaussian region method

Each pixel has associated with it a set of C posterior probabilities, where C is the number of cloud categories. The main concept behind the Gaussian region method is to replace a group of individual pixels, each having C posterior probabilities, by a single two-dimensional Gaussian with a single set of C probabilities (associated with that Gaussian) that give the best “fit” to the pixels within the range of the Gaussian. The theory is based on the Gaussian mixture model scheme presented in Appendix C.

As explained in Section C.1, a mixture of Gaussians can be used to model the probability density function that produces a given set of data points. The parameters of the Gaussians in the mixture are adapted using the Expectation Maximization (EM) [8, 28] algorithm such that the final mixture distribution maximizes the likelihood of the entire data set for that distribution. In this case, the probability density for one data point $\mathbf{r}^n$ for mixture component

j is given by:

$$p(\mathbf{r}^n | j) = (2\pi)^{-d/2} |\Sigma_j|^{-1/2} \exp \left[-\frac{1}{2} (\mathbf{r}^n - \mu_j)^T (\Sigma_j)^{-1} (\mathbf{r}^n - \mu_j) \right] \quad (7.2)$$

where d is the dimension of $\mathbf{r}^n$, and μ_j and Σ_j are the mean vector and covariance matrix respectively of Gaussian j . Note that this is the same as Equation C.5, with $\mathbf{x}$ replaced by $\mathbf{r}$.

Consider now the pixels in a satellite image. Assume that there are C distinct types of cloud possible (eg. Cumulus and Stratus). From the small neighbourhood around the pixel at $\mathbf{r}^n$, textural and spectral features are calculated as before (see Chapter 3), giving a feature vector $\mathbf{x}^n$ associated with pixel $\mathbf{r}^n$. Considered globally, the following can be written:

$$p(\mathbf{x}^n) = \sum_i^C p(\mathbf{x}^n | C_i) P(C_i) \quad (7.3)$$

where $P(C_i)$ is the global prior of class i . A set of two-dimensional Gaussians, again indexed by the variable j , is introduced and laid down across the image. Consider the joint probability $p(\mathbf{x}^n, \mathbf{r}^n | j)$, which is the joint probability of the position $\mathbf{r}^n$ and the features $\mathbf{x}^n$ given that the pixel comes from Gaussian j . The quantities $\mathbf{x}^n$ and $\mathbf{r}^n$ can be regarded as being independent, so that:

$$\begin{aligned} p(\mathbf{x}^n, \mathbf{r}^n | j) &= p(\mathbf{x}^n | j) p(\mathbf{r}^n | j) \\ &= p(\mathbf{r}^n | j) \sum_i^C p(\mathbf{x}^n | C_i) P(C_i | j). \end{aligned} \quad (7.4)$$

The term $P(C_i | j)$ can be regarded as a *local prior* for class i , that is local to the region influenced by Gaussian j . Expanding $p(\mathbf{r}^n | j)$, the following is obtained:

$$p(\mathbf{x}^n, \mathbf{r}^n | j) = (2\pi)^{-d/2} |\Sigma_j|^{-1/2} \exp \left[-\frac{1}{2} (\mathbf{r}^n - \mu_j)^T (\Sigma_j)^{-1} (\mathbf{r}^n - \mu_j) \right] \sum_i^C p(\mathbf{x}^n | C_i) P(C_i | j). \quad (7.5)$$

This is analogous to Equation 7.2. The class-conditional probabilities $p(\mathbf{x}^n | C_i)$ can be estimated using a parametric density estimation model or a neural network. The $P(C_i | j)$ become parameters of mixture component j which need to be estimated in the same way as the mean and covariance parameters.

During the course of fitting the Gaussians to the image, the shapes and positions of the Gaussians will therefore be updated at the same time as the local class priors for each Gaussian. Ideally, after fitting, the positions of the Gaussians and the values of the local class priors will maximize the likelihood that the image data was generated by that mixture of Gaussians. The process of “generating the image data” from the Gaussians can be regarded loosely in the following way. Points are sampled from the Gaussian in the same way as for the standard mixture model, and this generates a set of points. Now for each point a feature vector (and therefore textural and spectral properties) is generated according to the local class priors $P(C_i | j)$; if the prior is high for Cumulus, then textural and spectral properties like those of Cumulus will be generated for that Gaussian. Hence an image pair can be considered to have been generated, and if it is similar to the original image then the model fits well.

For the rest of the discussion, the following notation will be used:

$$p(\mathbf{x}^n | C_i) \equiv B_i^n \quad (7.6)$$

$$P(C_i | j) \equiv \mathcal{P}_i^j \quad (7.7)$$

$$p(\mathbf{x}^n, \mathbf{r}^n | j) \equiv p(\Delta^n | j). \quad (7.8)$$

7.3.2 Fitting using the EM algorithm

The EM algorithm is derived in Section C.2 and applied to the basic Gaussian mixture model in Section C.3. This derivation will be modified to fit with the new model set out in the previous section. The theory of Appendix C is applicable to this case (making appropriate

substitution of symbols) through to Equation C.22, which here becomes:

$$Q = \sum_n \sum_j P^0(j | \Delta^n) \log[p^1(\Delta^n | j) P^1(j)] \quad (7.9)$$

where the superscript 1 denotes new parameters and 0 denotes old parameters, and $p^1(\Delta^n | j)$ is given by Equation 7.5. For $\mathcal{P}$, use the notation $\tilde{\mathcal{P}}_i^j$ to denote new values, and $\mathcal{P}_i^j$ to denote old. The parameters $P^1(j)$ and $\tilde{\mathcal{P}}_i^j$ must sum to one, the latter one for all mixture components:

$$\sum_j P^1(j) = 1 \quad (7.10)$$

$$\sum_i^C \tilde{\mathcal{P}}_i^j = 1 \quad \text{for all } j. \quad (7.11)$$

This is achieved by introducing Lagrange multipliers:

$$\hat{Q} = Q + \lambda \left(\sum_j P^1(j) - 1 \right) + \sum_j \eta^j \left(\sum_i^C \tilde{\mathcal{P}}_i^j - 1 \right) \quad (7.12)$$

and maximizing this new $\hat{Q}$. Differentiating with respect to $P^1(j)$ leads to the result that $\lambda = -N$ as for the standard mixture model (N is the number of data points). Now maximize with respect to $\tilde{\mathcal{P}}_l^k$:

$$\begin{aligned} \frac{\partial \hat{Q}}{\partial \tilde{\mathcal{P}}_l^k} &= \frac{\partial Q}{\partial \tilde{\mathcal{P}}_l^k} + \eta^k \\ &= \sum_n \left[\frac{P^0(k | \Delta^n) B_l^n}{\sum_i^C \tilde{\mathcal{P}}_i^k B_i^n} \right] + \eta^k = 0. \end{aligned} \quad (7.13)$$

Multiply through by $\tilde{\mathcal{P}}_l^k$ and sum over l to give:

$$\sum_n P^0(k | \Delta^n) = -\eta^k \sum_l^C \tilde{\mathcal{P}}_l^k = -\eta^k. \quad (7.14)$$

Therefore $\eta^k = -\sum_n P^0(k | \Delta^n)$, giving:

$$\sum_n \left[\frac{P^0(k | \Delta^n) B_l^n}{\sum_i^C \tilde{\mathcal{P}}_i^k B_i^n} \right] = \sum_n P^0(k | \Delta^n). \quad (7.15)$$

Multiply both sides by $\tilde{\mathcal{P}}_l^k$ to give:

$$\sum_n [P^0(k | \Delta^n) P(C_l | \mathbf{x}^n, k)] = \tilde{\mathcal{P}}_l^k \sum_n P^0(k | \Delta^n) \quad (7.16)$$

where

$$P(C_l | \mathbf{x}^n, k) = \frac{\tilde{\mathcal{P}}_l^k B_l^n}{\sum_i^C \tilde{\mathcal{P}}_i^k B_i^n}. \quad (7.17)$$

Therefore

$$\tilde{\mathcal{P}}_l^k = \frac{\sum_n [P^0(k | \Delta^n) P(C_l | \mathbf{x}^n, k)]}{\sum_n P^0(k | \Delta^n)}. \quad (7.18)$$

Note that Equation 7.17, and hence the right-hand side of Equation 7.18, contains the parameter $\tilde{\mathcal{P}}$, which is the quantity to be determined on the left-hand side. Hence the approximation $\tilde{\mathcal{P}} \rightarrow \mathcal{P}$ is used on the right-hand side.

Thus an update equation for the EM algorithm has now been obtained for the $P(C_i | j)$ local priors of Equation 7.5. All the other update equations can be shown to be the same as for the standard mixture model EM algorithm set out in Section C.3, provided that the modified density is substituted from Equation 7.5.

7.3.3 A Gaussian region training run

An MLP was trained as described in Chapter 5 in order to provide estimates of $p(\mathbf{x}^n | C_i)$ for each data point. The MLP in fact gives an estimate of $P(C_i | \mathbf{x}^n)$, but this can be inverted using Bayes' theorem.

A simple training run is shown in Figure 7.3, where the one-sigma contours of the Gaus-

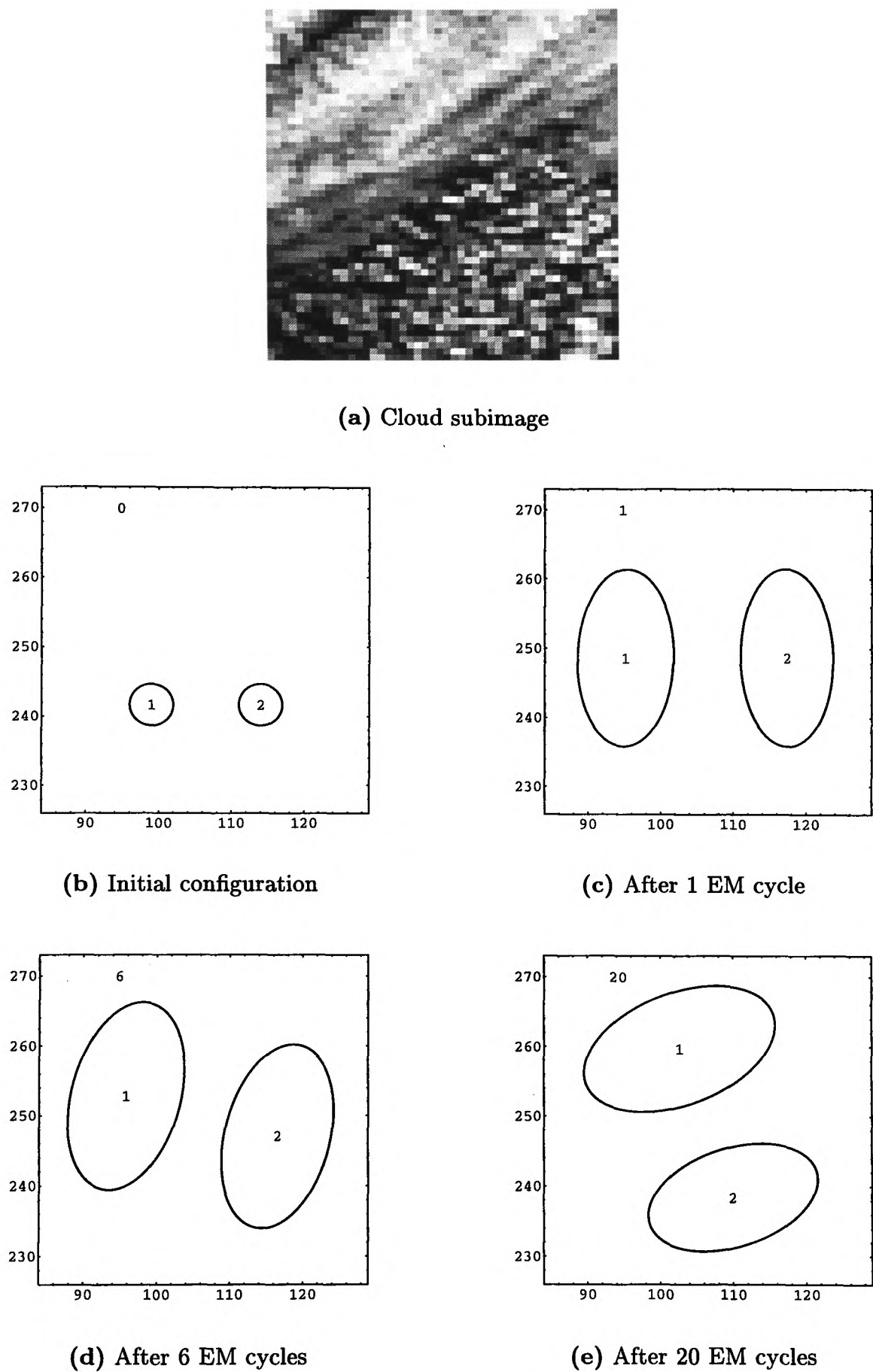


Figure 7.3: Example of an Expectation Maximization (EM) training run for the Gaussian region method. The one-sigma contours of the Gaussians are displayed. See text for further details. The cloud sub-image is taken from the visible image shown in Figure D.9 on Page 214.

Table 7.1: The final values of $P(C_i | j)$ for the two trial Gaussians in Figure 7.3.

C_i	$j = 1$	$j = 2$
Ci	0.4	4.8×10^{-23}
Cc	0.6	3.1×10^{-13}
Cs	2.3×10^{-42}	2.0×10^{-59}
As	3.9×10^{-23}	7.8×10^{-24}
Ns	2.2×10^{-32}	6.3×10^{-54}
Sc	2.8×10^{-12}	0.3
St	4.8×10^{-53}	5.5×10^{-52}
Cu	3.8×10^{-8}	0.7
Cb	1.0×10^{-12}	3.2×10^{-9}
Clr	7.3×10^{-47}	2.0×10^{-37}

sians are displayed. Two mixture components are used, equal to the expected number of distinct regions. The Gaussians quickly expand to fill the space, and then adjust to fit the class regions. At the end, the $P(C_i | j)$ values were as shown in Table 7.1. It can be seen that they do sum to one, and that they represent the likelihood that each region is of that cloud type; region one is most likely to be Cirrocumulus, and region two is most likely to be Cumulus.

Every one-sigma contour is taken as the starting curve for a separate snake, with each snake segmenting the cloud class having the largest $P(C_i | j)$ for that Gaussian. Snakes of the same class can merge into one as they move across the image. This Gaussian method could also be used in a different context to identify significant cloud regions whose exact shape is not needed. The central position and an estimate of the spread of the region can be obtained from the Gaussian component parameters.

7.4 The level set method

The level set method is proposed as an alternative to the snake method. Under certain circumstances during the evolution of the closed snake contours, it may be desirable for them to change connectivity and split, thereby undergoing a topological transformation. This is

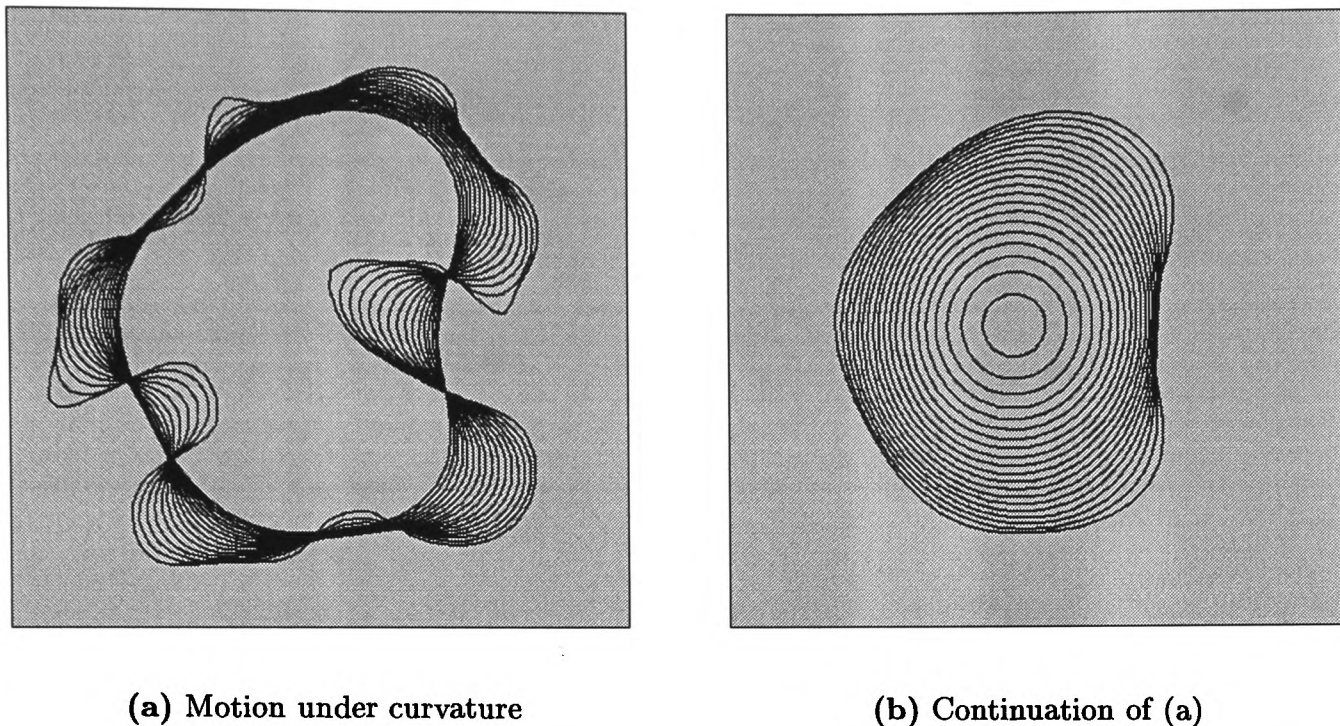


Figure 7.4: An illustration of curvature flow. The initial convoluted curve in (a) is made to move at each point along the curve with speed proportional to its curvature κ , which acts to smooth the curve. The influence of the curvature eventually causes it to become more circular, shrinking with time, as shown in (b). (From Malladi and Sethian [60]).

not easily handled using the snake method, but is dealt with transparently within the level set framework.

7.4.1 Level set theory

The method can be viewed as *curvature flow* [103]. A moving interface “flows” across the image, with the speed of flow at every point on the interface being a function of:

1. Curvature (increasing). This acts as a smoothing function; regions of high curvature are smoothed out.
2. Image gradient (decreasing). This acts as a stopping function, causing the curve to stop at salient image features.

Added to this is a constant speed inward or outward (the *advection* term). The curvature function is analogous to the bending and stretching forces of the snake method, and the image gradient function is similar to the image force that guides the snake. The effect of the curvature term is illustrated in Figure 7.4.

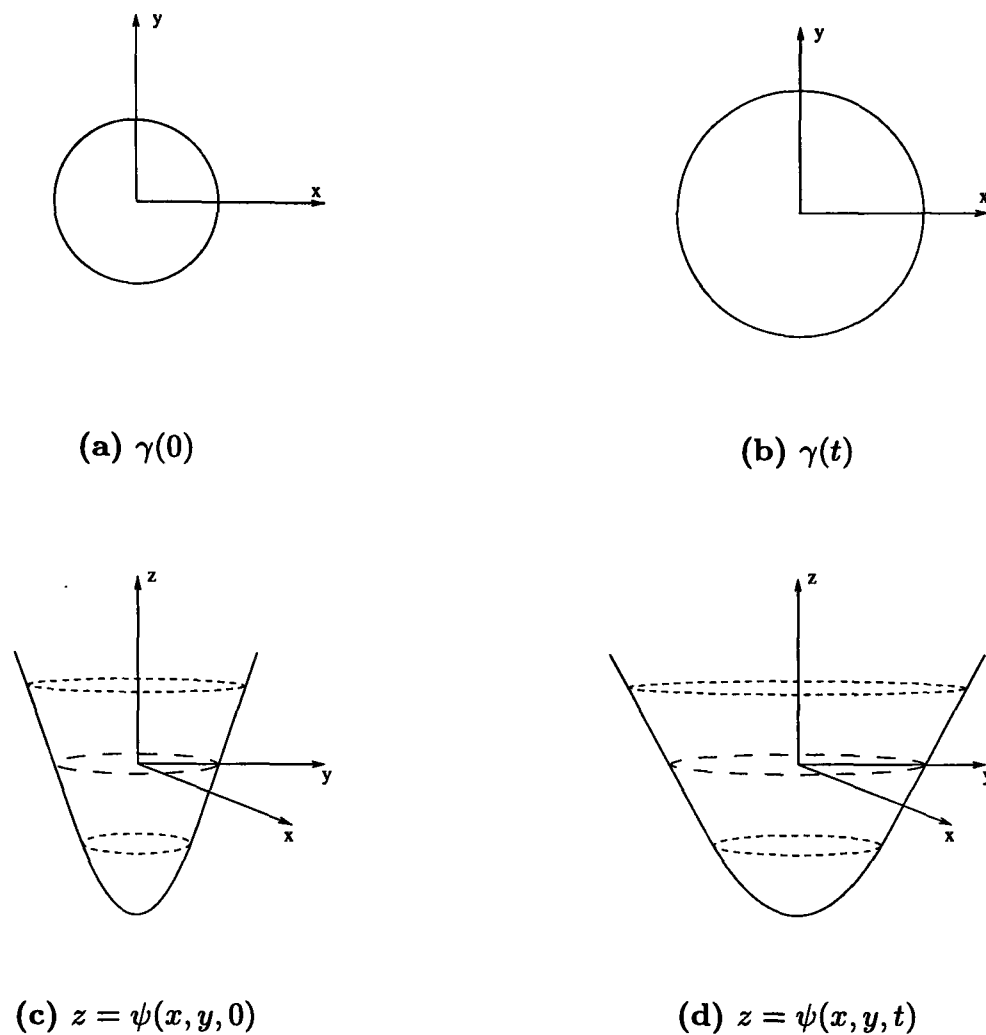


Figure 7.5: An illustration of the method of level sets. The propagating curve $\gamma(t)$ under consideration is embedded in a higher-dimensional function $\psi(x, y, t)$ as its zero level set $\{\psi(x, y, t) = 0\}$. Shown in (a) and (c) are γ and ψ at $t = 0$, which evolve into (b) and (d) at a later time t . The level set is an expanding circle in this case. See text for details.

Consider a closed propagating front $\gamma(t)$, at time t . The term *level set* derives from the fact that the front $\gamma(t)$ on the two-dimensional plane is the set of all points lying on a certain level of a function in three dimensions, $z = \psi(x, y, t)$, specifically the zero level set $\{\psi(x, y, t) = 0\}$. As an illustration, consider the example of Figure 7.5, showing $\gamma(t)$ as a propagating circle. The initial surface $\psi(x, y, 0)$ is constructed such that the zero level set $\{\psi(x, y, 0) = 0\}$ is the same as the initial curve $\gamma(0)$. Then ψ is made to evolve such that the zero level set $\{\psi(x, y, t) = 0\}$ always matches $\gamma(t)$. In this way, the motion of the curve has been embedded in the motion of the higher-dimensional surface. This approach was inspired by ideas introduced in Osher and Sethian [69], and applied mainly to problems in physics such as the modelling of crystal growth [104]. It was first applied to image shape recovery

by Malladi *et al* [63, 64] and Casselles *et al* [17, 18, 19]. The method has also been applied to image enhancement by Malladi and Sethian [60].

Suppose that $\psi(x, y, 0)$ is initialized such that

$$\psi(x, y, 0) = \pm \text{distance from } (x, y) \text{ to } \gamma(0) \quad (7.19)$$

where the minus (plus) sign is chosen if (x, y) is inside (outside) the initial curve $\gamma(0)$. The aim is to produce an equation for the evolving function $\psi(x, y, t)$ that contains the embedded motion of $\gamma(t)$ as the zero level set $\{\psi(x, y, t) = 0\}$. The derivation presented in Malladi *et al* [64] is followed here. Let $\mathbf{r}^\gamma(t) = (x^\gamma(t), y^\gamma(t))$ be a point on $\gamma(t)$. Since $\psi(x, y, t)$ is always zero on the propagating curve $\gamma(t)$,

$$\psi(x^\gamma(t), y^\gamma(t), t) = 0 \quad (7.20)$$

must hold for all points $\mathbf{r}^\gamma(t)$. Differentiating with respect to t and using the chain rule,

$$\frac{\partial \psi}{\partial t} + \frac{\partial \psi}{\partial x^\gamma} \frac{\partial x^\gamma}{\partial t} + \frac{\partial \psi}{\partial y^\gamma} \frac{\partial y^\gamma}{\partial t} = 0 \quad (7.21)$$

or

$$\frac{\partial \psi}{\partial t} + \nabla \psi \cdot \frac{\partial \mathbf{r}^\gamma}{\partial t} = 0. \quad (7.22)$$

Introducing the outward unit normal $\hat{\mathbf{n}} = \frac{\nabla \psi}{|\nabla \psi|}$, which is a unit vector normal to the curve in the xy plane, this becomes

$$\frac{\partial \psi}{\partial t} + |\nabla \psi| \hat{\mathbf{n}} \cdot \frac{\partial \mathbf{r}^\gamma}{\partial t} = 0 \quad (7.23)$$

or

$$\psi_t + F|\nabla\psi| = 0 \quad (7.24)$$

where F is the speed at which the curve travels in the direction of the normal to the curve. Although the theory has been formulated specifically for the *zero* level set of $\psi(x, y, t)$, it can be applied to *every* level set by considering a different level set $\psi(x, y, t) = k$ at the stage of Equation 7.20. Therefore Equation 7.24 applies over all space, and governs the evolution of the function $\psi(x, y, t)$, and hence the propagating curve $\gamma(t)$, through time.

This view on curve and surface motion offers many advantages. Firstly, although $\psi(x, y, t)$ remains a function, its zero level set can change topology, break, merge and form sharp corners, thereby providing enhanced topological properties over the snake method. Secondly, a discrete grid can be used together with finite differences to devise a numerical scheme to approximate the solution. Thirdly, intrinsic geometric quantities such as the normal or curvature of the curve can be easily extracted from the higher-dimensional function. Finally, the theory can readily be extended to moving surfaces in three dimensions (an issue that is not considered here).

7.4.2 Application to cloud segmentation

The level set framework set out in the previous section may be used for shape recovery from images if an appropriate speed function F is constructed. The speed function can be formulated such that it causes the propagating front to stop at the borders of the posterior image regions described in Section 7.2 in a manner similar to the snake method.

The speed function F is composed of three parts. Firstly, there is a constant *advection* term F_A which causes the front either to expand or to contract uniformly along its length, depending on the sign of F_A . Secondly, there is a geometry term F_G which depends on the local geometry of the front, usually the curvature κ , and which serves to constrain the front to smooth shapes. Thirdly, there is an image-based term F_I which must cause the overall speed

F to fall to zero near the boundaries that are being sought. The former two, F_A and F_G , are analogous to the internal forces of the snake method, while the latter, F_I , is analogous to the external image force of the snake method.

If the three terms are additive, that is $F = F_A + F_G + F_I$, then it is not possible [64] to construct an image term F_I that causes F fall to zero for all points on the front, unless $F_G = 0$. Therefore the additive term F_I term is replaced by a multiplicative image term g_I such that

$$F = g_I(F_A + F_G). \quad (7.25)$$

In the work described in this thesis, F_A will be positive, to propagate the front outwards, and F_G will be directly proportional to κ . Hence

$$F_A + F_G = (1 - \epsilon\kappa) \quad (7.26)$$

where ϵ is used to control the level of smoothness of the front. The curvature κ is calculated directly from ψ :

$$\kappa = \nabla \cdot \left(\frac{\nabla \psi}{|\nabla \psi|} \right) = - \frac{\psi_{xx}\psi_y^2 - 2\psi_x\psi_y\psi_{xy} + \psi_{yy}\psi_x^2}{(\psi_x^2 + \psi_y^2)^{3/2}}. \quad (7.27)$$

The sign is such that the curvature of a circle is positive when using the initialization scheme of Equation 7.19.

The image term g_I must cause the front to stop at salient boundaries in the posterior image $I(x, y)$. This is achieved by using a function that has values close to zero in regions of high image gradient and close to unity in regions with relatively constant intensity. The image term chosen for the work reported here is:

$$g_I(x, y) = \frac{1}{1 + |\nabla(G_\sigma * I(x, y))|} \quad (7.28)$$

where $G_\sigma * I(x, y)$ denotes the convolution of the posterior image with a narrow Gaussian smoothing filter (with a standard deviation of one or two) to remove noise. Hence Equation 7.24 becomes

$$\psi_t + g_I(1 - \epsilon\kappa)|\nabla\psi| = 0. \quad (7.29)$$

The formulation used for this thesis contains an additional term in Equation 7.29, as set out in Malladi and Sethian [61, 62], giving:

$$\psi_t + g_I(1 - \epsilon\kappa)|\nabla\psi| - \beta\nabla P \cdot \nabla\psi = 0 \quad (7.30)$$

where

$$P(x, y) = -|\nabla(G_\sigma * I(x, y))|. \quad (7.31)$$

The potential $P(x, y)$ attains a minimum near edges in the image, and therefore $\nabla P \cdot \nabla\psi$ is the projection of a force vector onto the normal to the front, which serves to attract the propagating front towards edges; the coefficient β controls the strength of this attraction.

The practical implementation of evolving the zero level set has not yet been addressed; the theory so far has been formulated for the continuous case. For the case of a discrete image and discrete time steps, finite difference approximations must be used. In addition, various computational procedures are implemented in order to speed up the segmentation process. Details of the finite difference and fast numerical schemes can be found in Section C.5 of Appendix C.

7.5 Results

The trained MLPs from Section 5.3 that gave the highest accuracy are used for generating the posterior images that are to be segmented in this section. The confusion matrices for these

MLPs are shown in Tables 5.2 and 5.5 for the Meteosat and Bankert databases respectively. Fifteen input features were used, with fifteen hidden-layer nodes, and all output classes. These are the same trained MLPs as were used for the results of Section 6.1 so that a direct comparison can be made between the pixel-by-pixel results of that section and the large-scale segmentation results of this section.

7.5.1 Snake method results

The Gaussian region method of Section 7.3.1 is initialized with radially-symmetric Gaussians. The n^2 Gaussians are arranged on an $n \times n$ grid such that neighbouring one-sigma circular contours touch tangentially, and the peripheral contours touch the edges of the image. Should there be $n^2 + m$ Gaussians, where $n^2 + m < (n + 1)^2$, then $n \times n$ Gaussians are arranged on a grid as before, and the remaining m Gaussians are given the same width as the others, and are placed at random positions across the image.

Using the Meteosat image pair from 6 January 1994, shown on Pages 206 and 207, the parameters of twenty-six Gaussians were set using the Gaussian region method. Each Gaussian, therefore, has a strong influence over approximately 10,000 pixels. This was found to provide a good compromise between preserving local detail (provided by narrower Gaussians) and capturing larger-scale properties (provided by wider Gaussians). The final positions of the one-sigma ellipses after training are shown in Figure 7.6, coloured according to the class with greatest $P(C_i | j)$ for each ellipse.

Snakes with 85 nodes were seeded on each of the Deep Convection ellipses. The stretching stiffness parameter, α , defined in Equation C.38 of Appendix C, was set to 1.5, and the bending stiffness parameter, β , was set to 0.5. The stabilized positions of the snakes are shown in Figure 7.7(a). It can be seen that the two Gaussians to the lower left have merged into one final region. This was achieved by checking at each iteration for overlap, and subsequently joining the two snakes at the crossing points when this occurred. Using 60 nodes and a larger value for β of 0.9, slightly smoother region boundaries are produced, as shown in Figure 7.7(b).

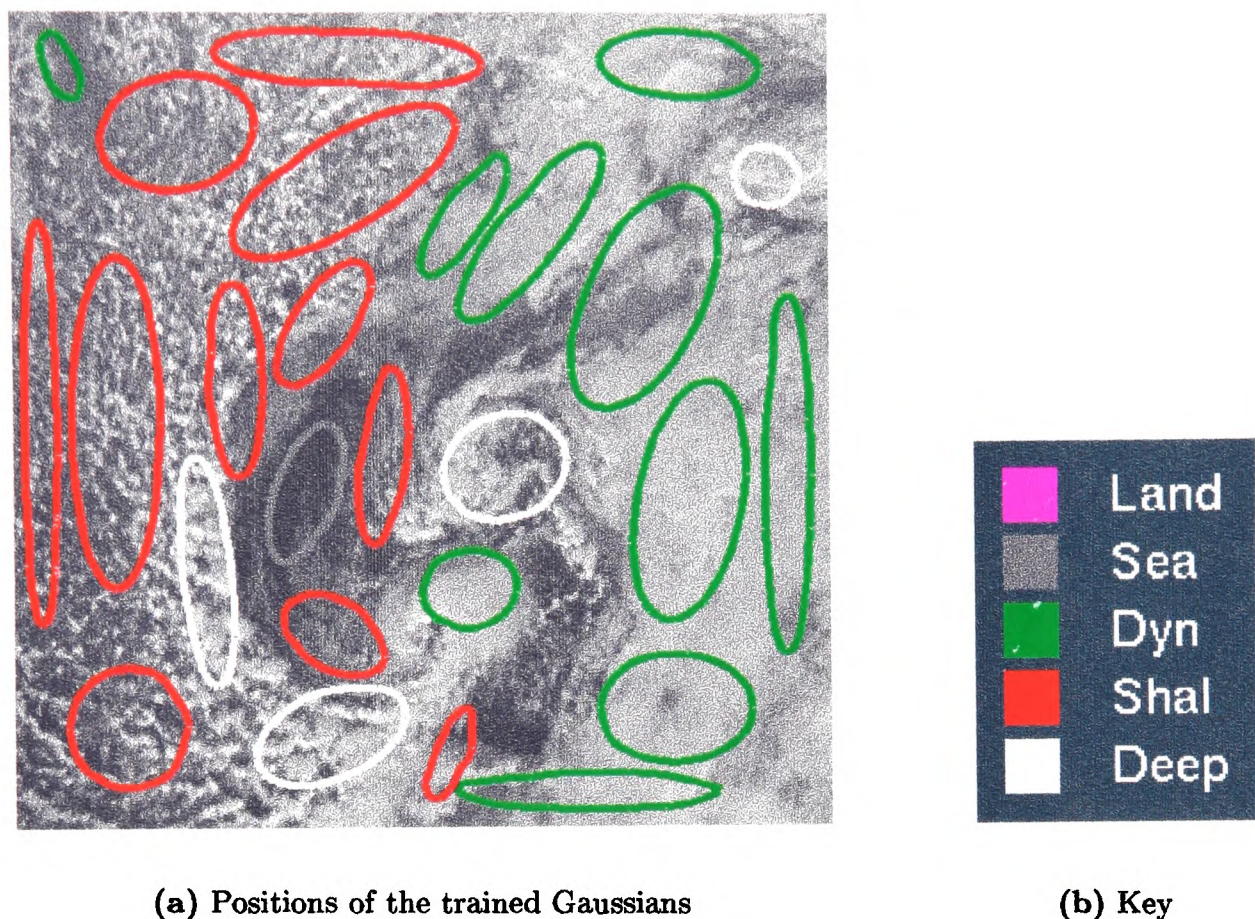
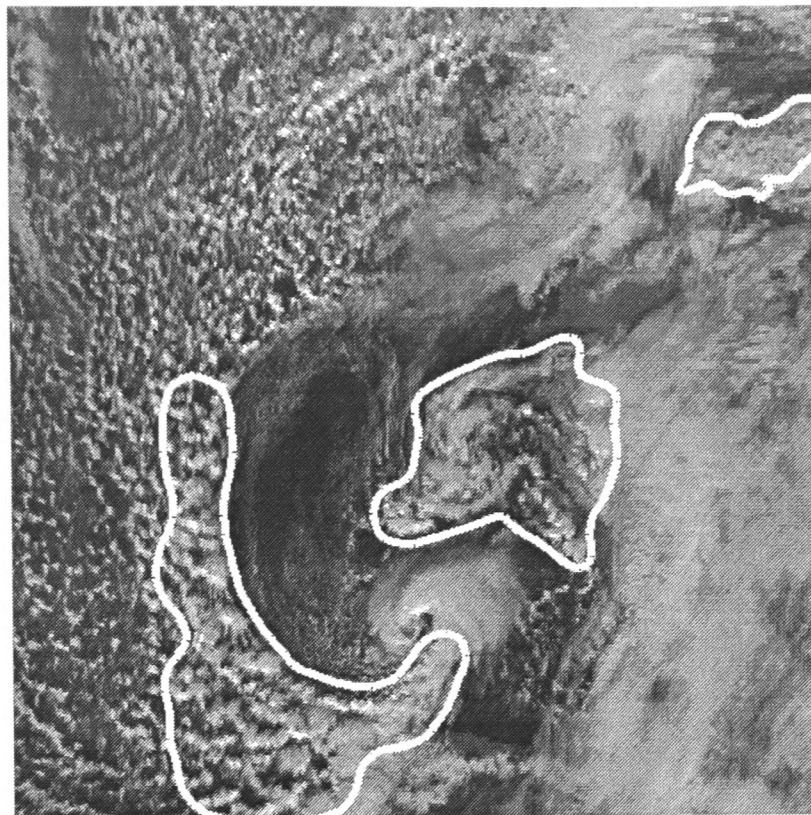


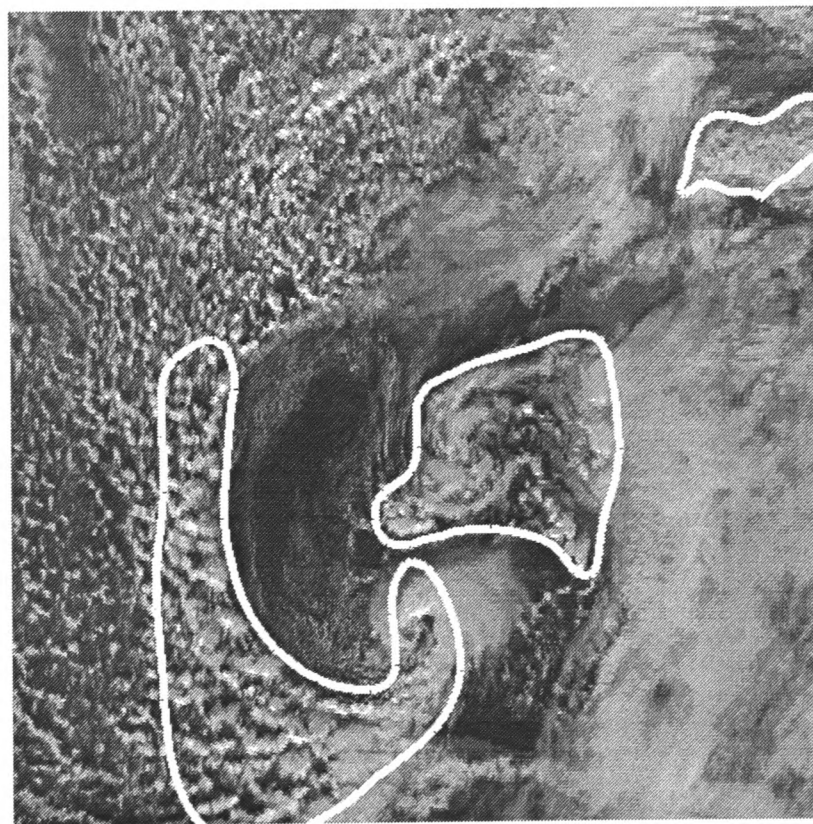
Figure 7.6: The final positions of the one-sigma contours of twenty-six Gaussians trained using the Gaussian region method. The Meteosat visible image is from 6 January 1994 at 1200 GMT. See Figure D.2 on Page 207 for the infra-red image, and Figure D.1 for a larger version of the visible image.

In both, the snakes can be seen to have altered shape to surround the significant regions of Deep Convection within the image.

The outward normal to the snake, which is the direction of the image force described in Section 7.3, was approximated at node i by rotating by -90° the normalized vector from node $(i - 1)$ to node $(i + 1)$. Nodes are initialized such that i increases anticlockwise around the closed snake contour. The effect of altering the strength of the image force is mainly to change the speed at which the snake moves to surround the regions within the posterior image. Too large a force, however, leads to instability which can cause the snake to develop a twist. In so doing, an inward force as defined above becomes an outward force, leading to an ever-expanding snake. It was found that a value of 1.0 for the magnitude of the image force vectors at each node (see Equations C.42 and C.43) provided reasonable stability. Occasionally, however, a twist would still develop.



(a) Deep Convection

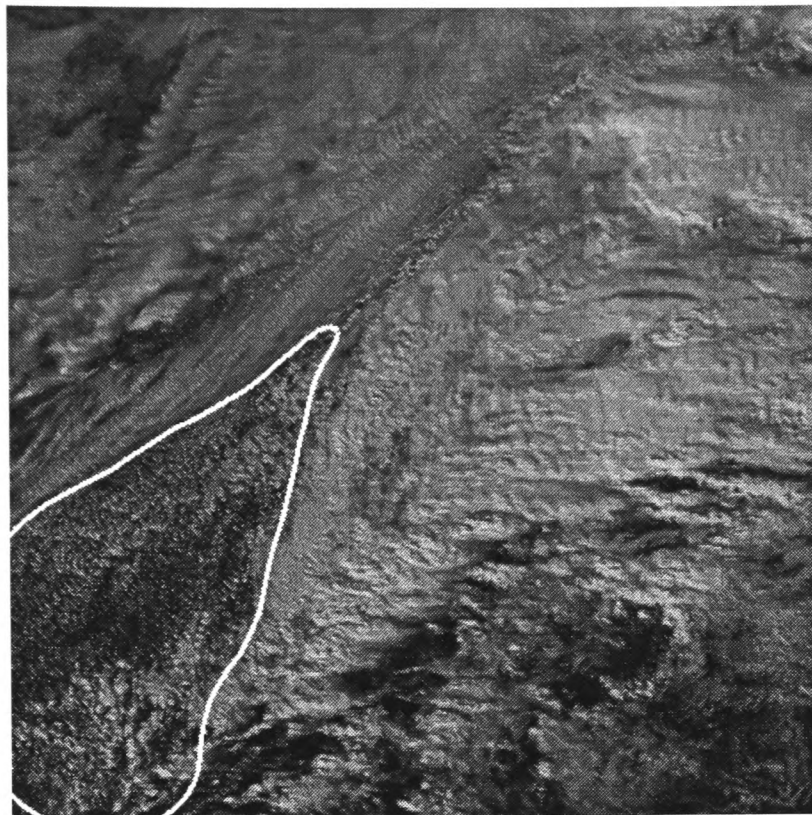


(b) Deep Convection

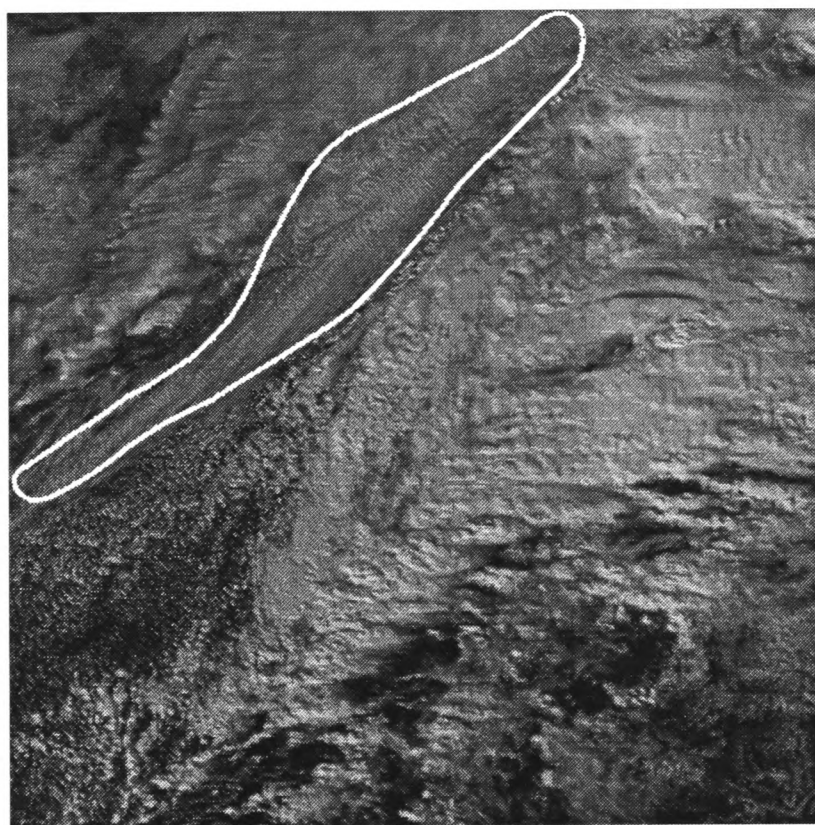
Figure 7.7: Snake segmentation of the Deep Convective areas, using the one-sigma contours of Figure 7.6 as the initial positions of the snakes. Each initial snake in Figure (a) had 85 nodes, with a stretching stiffness α (see Section C.4) of 1.5, and a bending stiffness β of 0.5. For Figure (b), 60 nodes were used, with α set to 1.5 and β to 0.9, producing a slightly smoother segmentation. The Meteosat visible image is from 6 January 1994 at 1200 GMT. See Figure D.2 on Page 207 for the infra-red image, and Figure D.1 for a larger version of the visible image.

Regions extracted from an example image pair from the Bankert database are shown in Figure 7.8. The same image force scheme as above was used, with 70 snake nodes and with α and β set to 1.5 and 0.5 respectively. The regions of Cumulus in Figure 7.8(a) and Cirrocumulus in Figure 7.8(b) have been accurately extracted, with constraints on the smoothness of the region boundaries. A comparison may be made to the pixel-by-pixel classification produced by the same MLP, shown in Figure 6.2 on Page 113. The Cirrocumulus region with noisy boundaries in Figure 6.2 has been extracted in Figure 7.8(b) with smooth boundaries. There are areas within the Cumulus region defined by the snake segmentation of Figure 7.8(a) that are classified as Stratocumulus by the MLP in Figure 6.2. These areas have significant posterior probabilities for *both* classes, indicating that they share characteristics of both Cumulus and Stratocumulus. When considered in the context of the surrounding area of Cumulus, the large-scale snake segmentation approach has grouped these areas into the Cumulus category.

A final example is shown in Figure 7.9, where the snake method has been used to define a region of Dynamic cloud. The corresponding pixel-by-pixel classification is shown in Figure 6.3 on Page 114; only the upper-left 512×512 pixel region is used here. A single snake was seeded manually on a circle with a twenty-pixel radius in the centre of the Dynamic region, with 70 nodes and the same stiffness parameters as the previous example. The image forces caused the snake to expand outwards, travelling over the small areas of cloud in Figure 6.3 that were not classified as Dynamic cloud, to locate the boundary of the main area of Dynamic cloud. Due to the smoothness constraint, the snake has strayed into the regions of clear sea and shallow convective cloud in the north-east corner. This highlights one of the problems associated with the snake method, which is that of choosing the bending stiffness parameter and number of nodes that allows sufficient flexibility whilst still allowing a stable evolution to occur. If the stiffness is set low enough to allow flexibility, then the coupling of image forces with a certain arrangement of the snake nodes can cause instability to develop. A related problem can be seen in the Cumulus segmentation of Figure 7.8(a) to



(a) Cumulus



(b) Cirrocumulus

Figure 7.8: Snake segmentation of (a) Cumulus and (b) Cirrocumulus cloud on an image from the Bankert database. Seventy nodes were used in each case, with α set to 1.5 and β to 0.5. The AVHRR visible image is from 13 December 1988. See Figure D.10 on Page 215 for the infra-red image, and Figure D.9 for a larger version of the visible image.

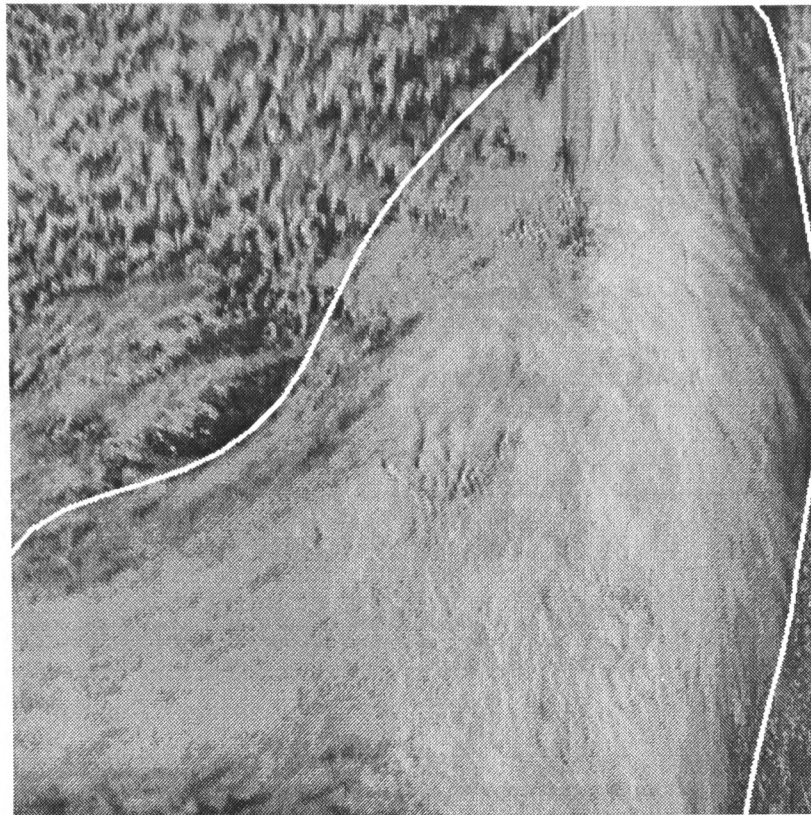


Figure 7.9: Snake segmentation of Dynamic cloud on an image from the Meteosat database. Seventy nodes were used in each case, with α set to 1.5 and β to 0.5. The Meteosat visible image is from 3 April 1994 at 1600 GMT. See Figure D.4 on Page 209 for the infra-red image, and Figure D.3 for a larger version of the visible image.

the far south-west. The image forces are insufficient in the corner to overcome the internal forces of the snake, which force the snake boundary to cut across the corner.

7.5.2 Level set method results

As explained in Section C.5.2, a combination of the fast marching technique and the narrow band method was used for segmentation using the level set approach. Manually-selected seed points on the posterior image were used as the $T = 0$ starting points for the fast marching method, which was employed for a certain time. Subsequently, 250 steps of the full level set methodology (using the narrow band technique) were carried out. The time step, Δt , was set to 5×10^{-6} for all results of this section and the narrow band width, d , to 20 pixels, motivated by the results of Malladi *et al* [62, 64]. Unless otherwise stated, the values of ϵ and β were both set to 0.08 for all the results. These values of the model parameters were found to give the best compromise between the smoothness and the accuracy of the region boundaries.

A demonstration of the evolution of the zero level set is shown in Figure 7.10 for the Cirrocumulus posterior image from the same Bankert image as used in the last section (see Figure 7.8). Three seed points were used, the positions of which are shown by the white dots in each of the images. The fast marching method was employed for a certain time t , followed by 250 iterations of the narrow band approach as described above. In regions where the posterior image is uniformly dark (low posterior probability), the front propagates with approximately constant speed in all directions, as demonstrated by the evolution of the lower front in (a) and (b). When high image gradients are encountered, the image term g_I causes the front to slow down, as is happening in (b). The middle front in (b) is beginning to wrap around small bright regions, but by (c) the front has passed through them. In this way the smoothness constraint of the technique has caused small (insignificant) areas to be ignored, as happened in the snake method. Also apparent is the merging of the lower and middle fronts, which is also possible in the snake method.

The change from (c) to (d) is evidence of the superior topological properties of the level set method compared with the snake method. The fronts have changed topology by splitting around the larger bright regions, to leave these regions defined behind them. Under the snake scheme, the fronts would not have been able to proceed further unless they were to move over the regions, thereby ignoring them. By (f), the front has passed over the entire image, demarcating all the significant regions of posterior probability.

The final zero level set is shown overlaid onto the visible image in Figure 7.11, and may be compared with the snake segmentation of Figure 7.8(b) and the pixel-by-pixel classification map of Figure 6.2. The level set approach has been able to provide boundaries that are more detailed and spatially accurate than the snake method, whilst still producing a degree of smoothness. A further stage could be incorporated to remove the smaller regions by setting a threshold on the area of the region. Alternatively, a measure such as the sum of the posterior probabilities of all pixels within the region could be calculated. This would provide

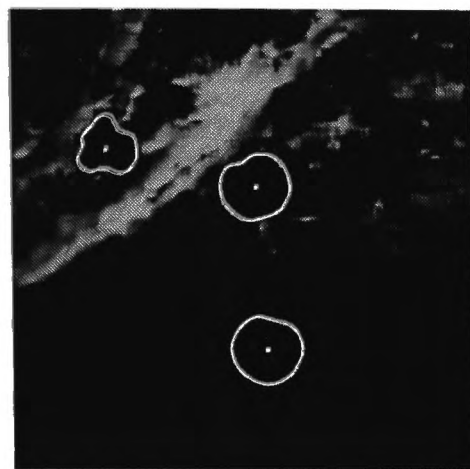
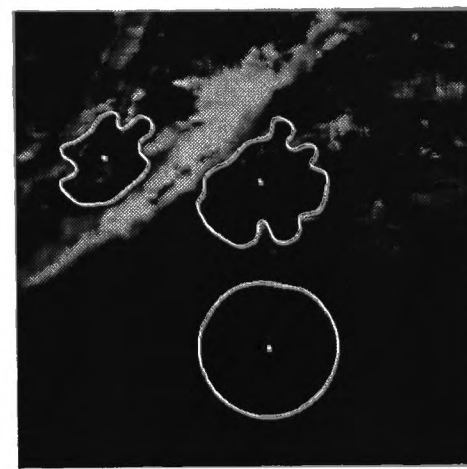
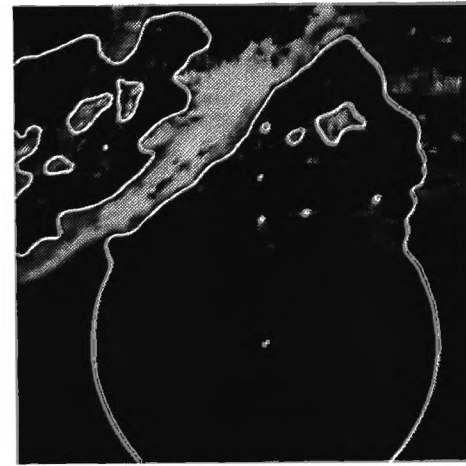
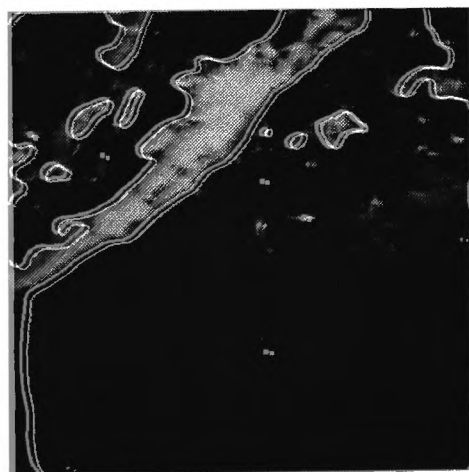
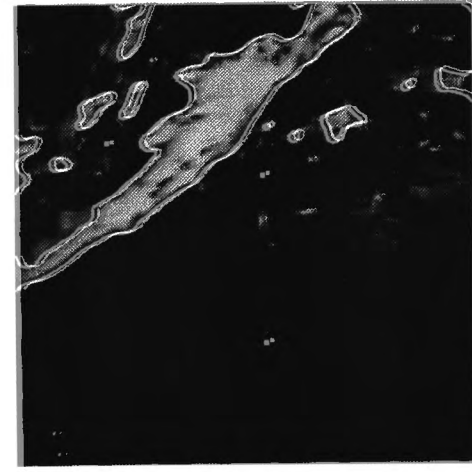
(a) $t = 0.1$ (b) $t = 0.2$ (c) $t = 0.3$ (d) $t = 0.5$ (e) $t = 0.7$ (f) $t = 5.0$

Figure 7.10: Evolution of the zero level set on the Cirrocumulus posterior image. Three seed points are used, and the level set evolved outwards from these. The fast marching method (see Section C.5.2.2) is used to propagate the front until the time shown under each figure. The final zero level set is shown overlaid onto the visible image in Figure 7.11.

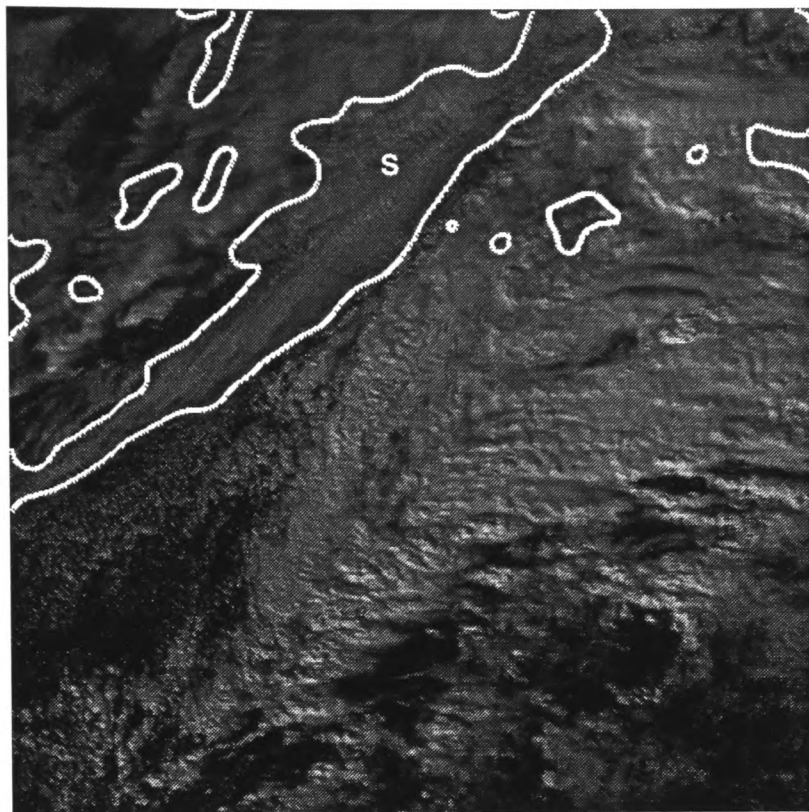
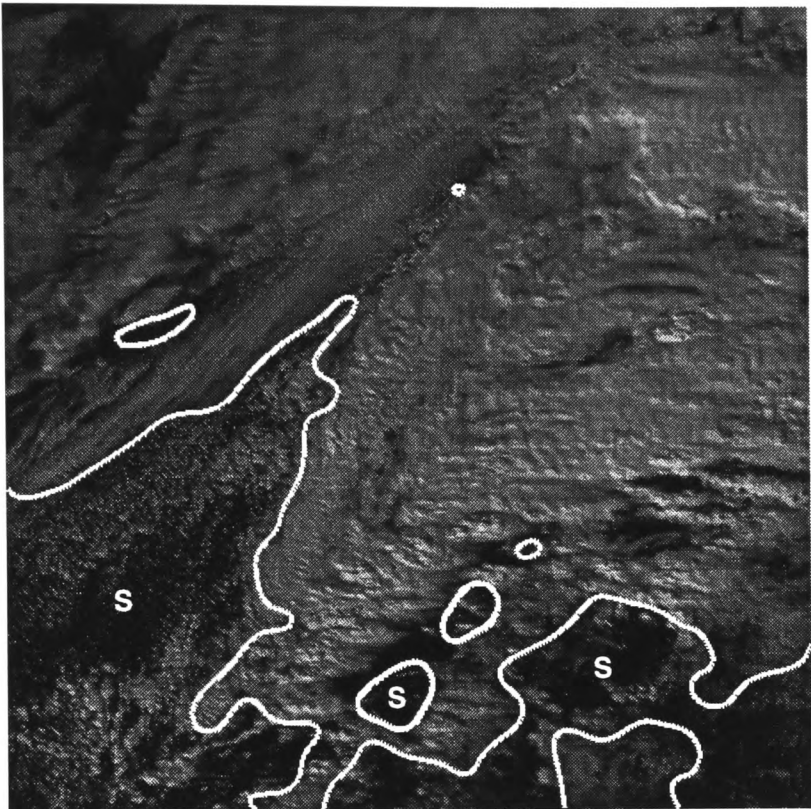


Figure 7.11: The final zero level set shown in Figure 7.10(f) is shown here overlaid onto the visible AVHRR image, from 13 December 1988, with the main Cirrocumulus region labelled *S*. See Figure D.10 on Page 215 for the infra-red image, and Figure D.9 for a larger version of the visible image.

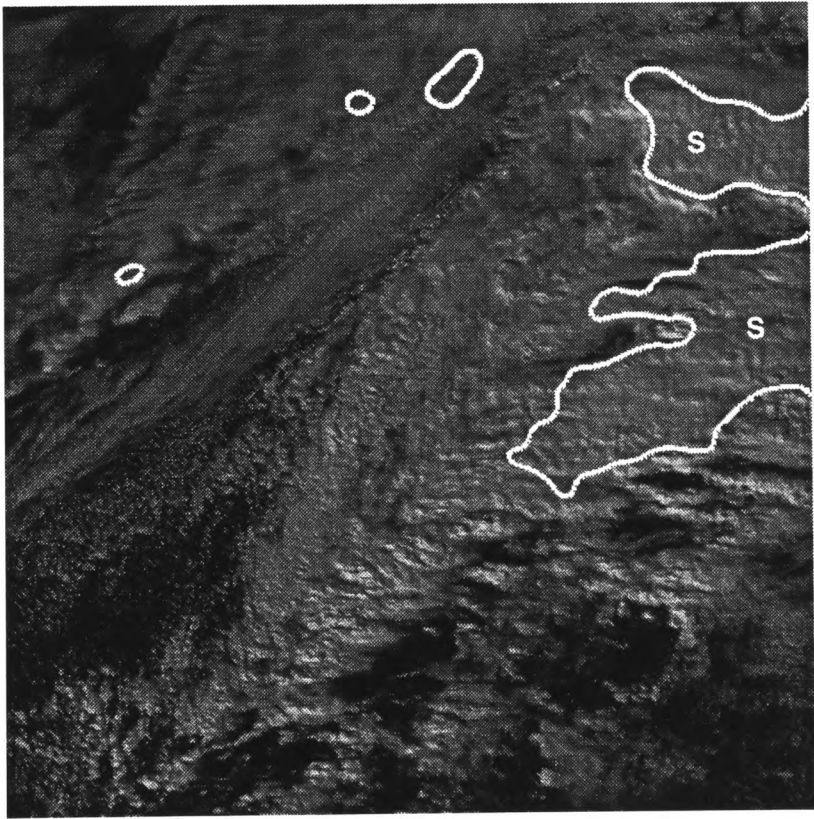
a measure of the significance of the region, and a threshold could be set on this to remove the insignificant regions.

The level set segmentation of Cumulus cloud, corresponding to that for the snake segmentation of Figure 7.8(a), is shown in Figure 7.12(a). The boundaries of the main region to the south-west of the image have been located close to those obtained with snake method, although with more detail. The level set segmentation has also demarcated a region further east, where there is some evidence of shallow convective activity, either cumulus or stratocumulus. Regions of Nimbostratus have been accurately segmented in Figure 7.12(b), and correspond closely to the labelled sample regions for this image which are displayed in Figure 1.6 on Page 23.

The level set method, like the snake method, performs the segmentation of each class independently of all other classes, since it operates on the posterior image of only one class. The boundaries defined for two cloud fields that are adjacent to each other are shown in Figure 7.13 for the Shallow and Deep Convective classes of the Meteosat databases. It is clear

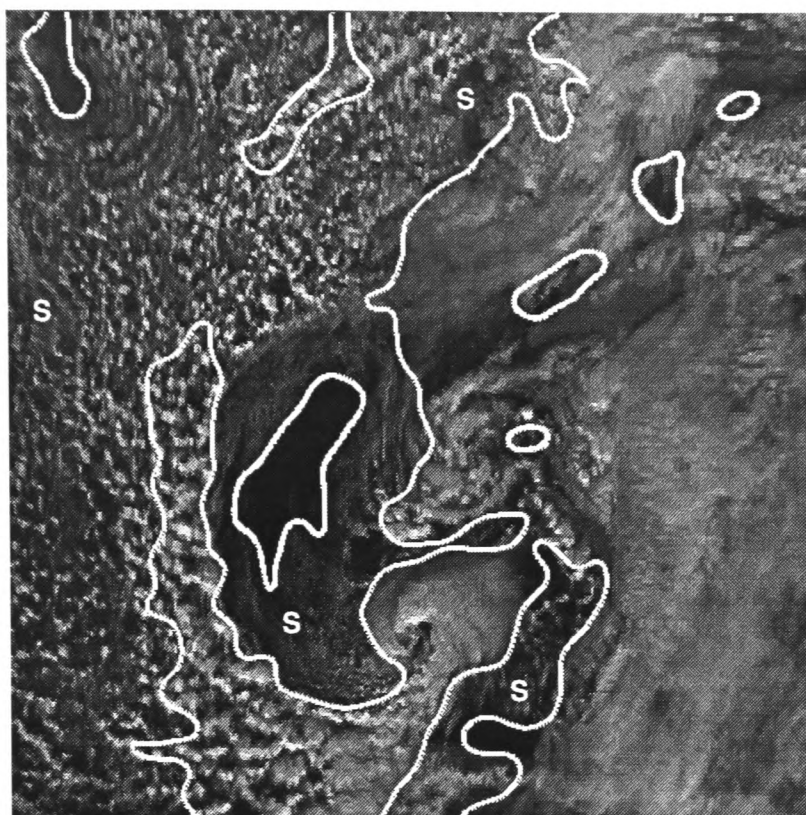


(a) Cumulus

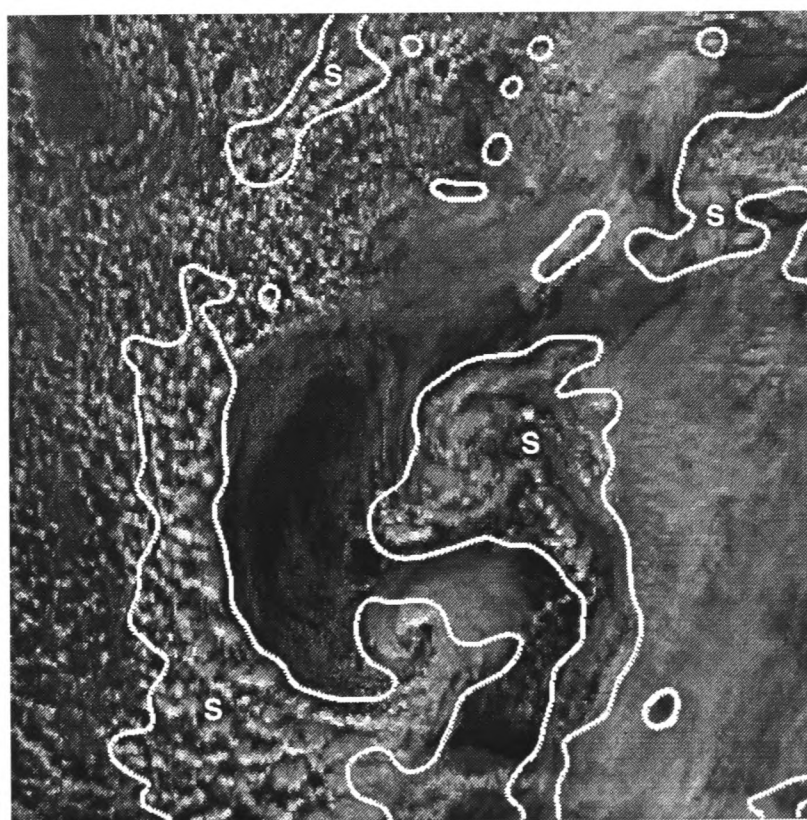


(b) Nimbostratus

Figure 7.12: Two further level set segmentations from the same image as shown in Figure 7.11. The main segmented regions are labelled S . See Figure D.10 on Page 215 for the infra-red image, and Figure D.9 for a larger version of the visible image.



(a) Shallow Convection



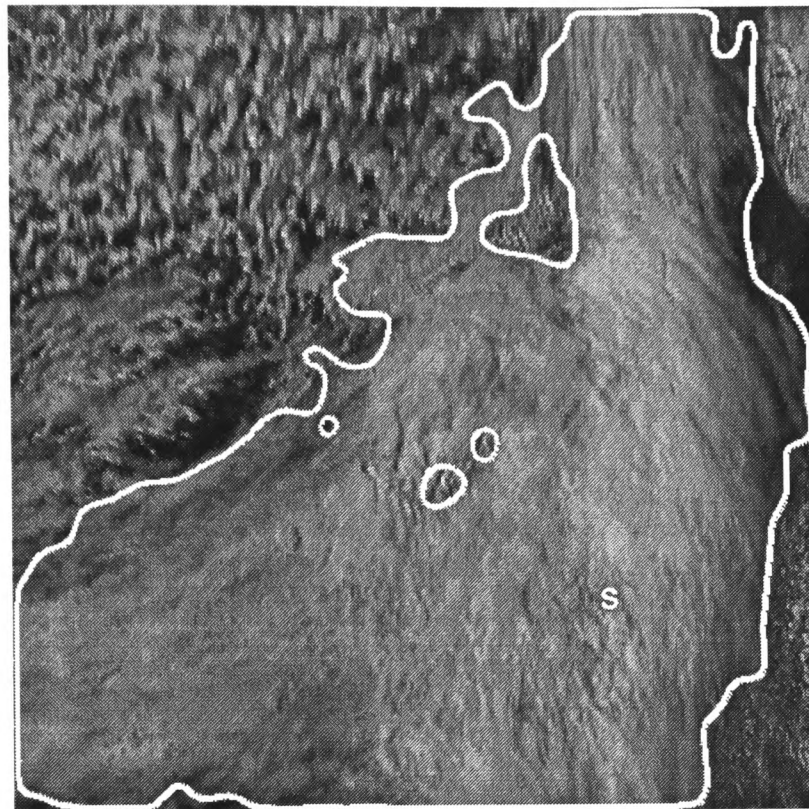
(b) Deep Convection

Figure 7.13: Level set segmentation of (a) the Shallow Convection class, and (b) the Deep Convection class. The main segmented regions are labelled *S*. The Meteosat image is from 6 January 1994 at 1200 GMT. See Figure D.2 on Page 207 for the infra-red image, and Figure D.1 for a larger version of the visible image.

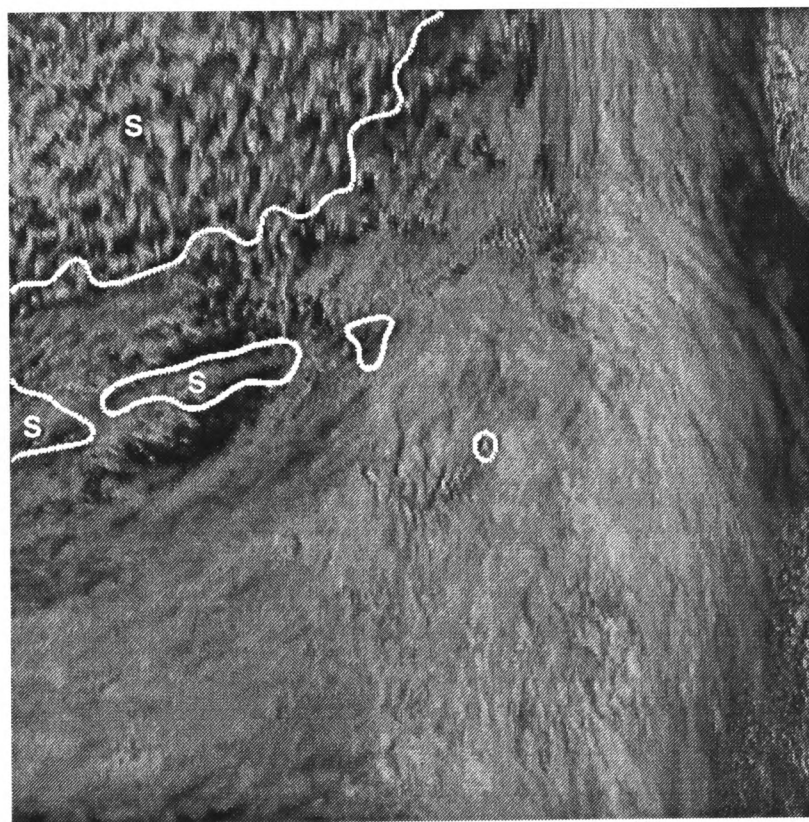
that there is a degree of overlap between the regions defined in each image. When segmenting the posterior image for class i , there is a tendency for the boundary between classes i and j to be located slightly further towards region j than the corresponding boundary using the posterior image for class j . This is probably due to the fact that the zero level set was seeded inside regions of low posterior probability and was made to propagate outwards to locate the boundaries of the regions of high posterior probability. The opposite was done for Figure 7.14(a), where the level set was seeded inside the large region of Dynamic cloud and propagated through this region to locate the boundaries. It can be seen that the boundary has the opposite tendency to stop just inside the boundary edge, as demonstrated by the small gap between the edge of the image and the final zero level set.

Comparing the level set segmentations of Figures 7.13(a) and 7.13(b) with the pixel-by-pixel classification of Figure 6.1, it can be seen that the regions have been extracted with a good compromise between spatial accuracy and smoothness of the boundary. The clear sea region just to the west of centre in Figure 7.13(a) has been demarcated with a high degree of accuracy, and the boundary is flexible enough to include the thin tongue of Shallow Convection protruding eastwards just south of centre. The topological properties of the boundaries displayed in these two images would not be easily achieved with the snake method.

Two final examples are displayed in Figure 7.14. Figure 7.14(a) was discussed above with reference to the positioning of the boundaries between regions. It is similar to the snake segmentation of Figure 7.9, with two main differences. Firstly, the boundary is more accurately defined in the level set segmentation, especially to the north-east corner. This was highlighted as a problem with the snake approach. Secondly, the small region of shallow convection within the main Dynamic region has been demarcated by the level set method, which is not possible using one snake due to the topological limitations mentioned above. The region of Deep Convection to the north-west of the image has been accurately segmented as one region in Figure 7.14(b).



(a) Dynamic



(b) Deep Convection

Figure 7.14: Level set segmentation of (a) the Dynamic class, and (b) the Deep Convection class. The main segmented regions are labelled *S*. See text for details. The Meteosat image is from 3 April 1994 at 1600 GMT. See Figure D.4 on Page 209 for the infra-red image, and Figure D.3 for a larger version of the visible image.

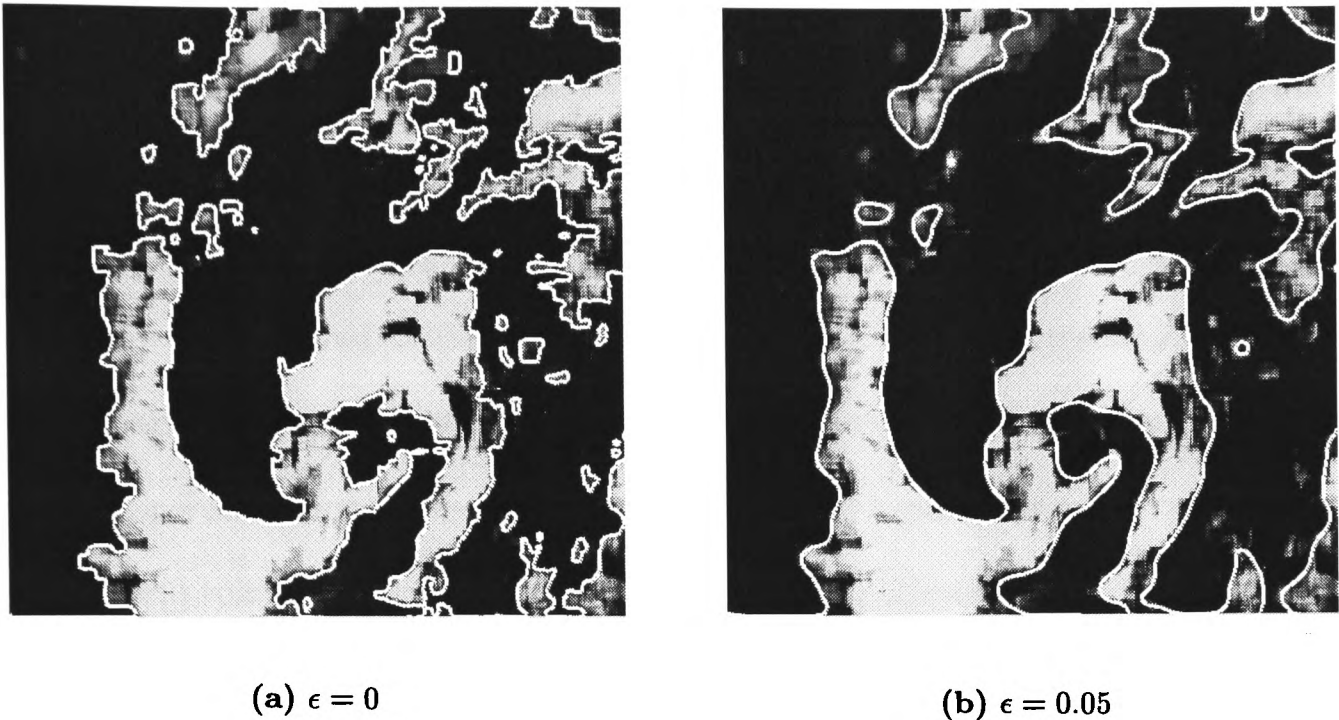


Figure 7.15: Level set segmentation of a posterior image for two values of the smoothness parameter ϵ from Equation 7.29. For (a) there is no constraint on the smoothness of the final curve. Using a higher value of ϵ produces a less noisy curve, as shown in (b).

The effect of the smoothness parameter, ϵ , is demonstrated in Figure 7.15. The posterior image shown is for the Deep Convection class, using a different MLP to that used for the other results of this section. When ϵ is set to zero, as done for Figure 7.15(a), there is no smoothness constraint on the zero level set. Therefore a very noisy boundary is formed. Raising the value of ϵ to 0.05 imposes the desired smoothness that is achieved in Figure 7.15(b).

7.6 Conclusions

A methodology has been presented in this chapter that combines low-level processing with analysis at the synoptic scale, which has enabled the definition of large-scale regions of uniform cloud type. The pattern recognition capabilities of the MLP were shown in Chapter 5 to be sufficient to achieve very good cloud-type classification accuracy at a local scale, and these capabilities are exploited directly with the posterior image approach developed in this chapter. For each cloud type, the spectral and textural properties are processed by the MLP to produce a single image in which significant areas of that cloud type are highlighted by

groups of pixels having a high posterior probability.

Two segmentation techniques have been presented to organize this array of individual pixels into meaningful and homogeneous regions. The first is the well-known active contour (or snake) model, using a novel Gaussian region method in order to initialize the positions of the contours. The second is the more recently-developed level set method, which was first applied to shape extraction in 1995 [64]. The two methods are conceptually similar, both using a moving front to locate the borders between cloud types, and produce similar results. The level set approach, however, has been shown to have a number of important properties that make it more suitable for the recovery of cloud regions from satellite images. These can be broadly divided into two areas, which are now discussed in turn.

Firstly, stability-related problems can occur when using the snake method. With certain combinations of the snake parameters and image forces, twists may occur where the snake crosses over itself. This can be prevented by imposing a higher energy penalty on bending in order to prevent such twists from occurring, or by using smaller image forces. This in turn can lead to regions with overly-smooth boundaries, or snakes that cut across image corners. The level set method was able to locate boundaries with more detail and greater spatial accuracy, while still imposing an appropriate degree of smoothness on them.

Secondly, the topological properties of the level set approach were found to provide important benefits. A snake contour is unable to change its topology by splitting without the introduction of cumbersome and time-consuming methods to detect when a split should occur (the implementation used in this thesis allows only for the merging of two snakes, for the reason given below). When a moving snake contour encounters an obstacle (in the form of a region of different cloud type), it is unable to split around it, but instead must either move over it or stop at that point. Since the snake method is minimizing an energy function, this is equivalent to being trapped in a local energy minimum.

This highlights the importance of seeding the snakes correctly, since two initial snakes either side of the obstacle would solve the problem highlighted above. Topological changes

occur transparently within the level set framework, and this in turn decreases the importance of the initialization procedure; a single seed region can undergo numerous topological changes as it evolves to extract any number of separate regions. The Gaussian region method developed for this thesis did, however, provide good a initialization for the snake method, seeding the snakes inside regions of distinct cloud type so that they expand outwards to locate the boundaries. For this reason, only snake merging functionality was needed in our implementation.

As previously mentioned in Section 7.1, Haralick and Shapiro [43] suggest that a segmentation should have the following properties:

- Regions should be uniform and homogeneous with respect to the features.
- Adjacent regions should have significantly different features.
- Regions should be simple and without too many holes.
- The boundaries of each region should be simple, not ragged, and be spatially accurate.

These four criteria are satisfied by both the active contour model and the level set method. The segmentation schemes presented in this chapter are an attempt to overcome the limitations of the purely pixel-based methods where the image is effectively a collection of independent pixels. The region-based approach treats the pixels as belonging to an area of cloud having associated constraints on the smoothness of its boundary.

The result is that the entire image can be defined in terms of a small number of large regions rather than a large number of pixels, which can be more easily related to meteorological conceptual models occurring at the synoptic scale. This large-scale treatment is a vital part of the scheme suggested by Pankiewicz [71, 73] and Peak and Tag [82] for the development of a pattern recognition system capable of automatically identifying synoptic-scale meteorological objects. Such a system is one of the goals of the project within the UK Meteorological Office described in detail in Section 1.3, with the eventual aim of assimilating this synoptic information into the numerical weather prediction (NWP) model.

Chapter 8

Conclusions and contributions

The aims of the work presented in this thesis were set out in detail in Section 1.3. They were put in context of the overall goals and current achievements of a parallel project within the UK Meteorological Office. Several of these goals were highlighted as being of relevance to this thesis:

- Determination of the best set of features for optimum classification.
- Investigation into the use of the Kohonen neural network.
- Development of a neural network to classify clouds.
- Development of a methodology for a pattern recognition system capable of identifying synoptic-scale meteorological objects.

This thesis has addressed a number of issues, including all of the above, relating to the use of statistical pattern recognition techniques for the interpretation of meteorological satellite images, which has complemented the research being carried out at the UK Meteorological Office.

Feature extraction and selection was investigated in Chapter 3, using both the Bhattacharyya distance measure and a new method based on a trained RBF network. It was argued that the RBF network, in contrast to the MLP, is particularly suited for the determination of an optimal feature set. The features chosen on the basis of the Bhattacharyya

distance were found to be very sensitive to whether or not the second covariance term was included in the distance measure. Sequentially removing features, so that all features are present at the start, was found to be a more reliable method for selecting features that provide discrimination in combination with each other. This is the first reported use, to the author's knowledge, of the RBF network for feature selection using this approach.

A major study of the structure of cloud feature vectors in high-dimensional feature space had not been conducted prior to this thesis. Three visualization methods were compared in Chapter 4: the Kohonen map, the Sammon map and the Generative Topographic Mapping (GTM). An evaluation of the strengths and weaknesses of each approach concluded that the GTM method had significant advantages over the other methods. The results also provided a validation of the RBF feature selection method, since the first six features were found to contain sufficient discriminatory information for good visual separation of the cloud classes on the two-dimensional plane. This work is one of the first major applications of the recently-developed GTM method to real-world data.

These visualization results were found to be invaluable for the interpretation of the classification results of Chapter 5. Sources of error could be explained in terms of the overlap of cloud classes in feature space apparent from the two-dimensional GTM plots, and suggested that better classification accuracy could not be achieved using the current features. Accuracies of 83.0% and 95.4% (using an RBF network) reported on two separate image databases showed that the same techniques could be applied to data from different satellites (having different spatial resolutions). The combined results of Chapters 4 and 5 demonstrated the complex distribution of data in feature space, and confirmed the need for the complex decision boundaries of the non-linear neural network methods for cloud-type classification; the linear discriminant clearly gave inferior results.

The limitations of a purely pixel-based approach were highlighted in Chapter 6. The classified image produced was found to be very noisy due to small regions of inappropriately-classified pixels, due in large measure to the fact that each pixel is processed independently

of its neighbours. As a contextual alternative, a multiscale Markov Random Field model was applied to cloud classification for the first time. This was shown to give a slight reduction in noise, especially for the Meteosat database. The novel use of a three-dimensional Kohonen map was found to provide an unsupervised alternative to the MLP classification maps, the colour of each pixel being controlled directly by the textural and spectral properties of the image.

An important component of the three-stage synoptic feature analysis scheme proposed by Pankiewicz [71, 73] is the broad segmentation of the image. A wide-ranging literature review revealed that there has been very little research conducted into the analysis of cloud fields at the synoptic scale. With standard pixel-based methods, there is no concept of a *region*; the pixels are independent of each other and are not organized into homogeneous groups. In Chapter 7, large-scale segmentation techniques were applied for the first time to meteorological satellite images, and it was shown that these techniques allow the definition of synoptic-scale regions of a given cloud type. The recently-developed level set technique has been shown to be very robust, and to have suitable characteristics for the accurate determination of boundaries between major cloud regions. In addition, a novel method has been developed for the reliable initialization of the snake method, which is of crucial importance. The region-based approach presented in this thesis has the potential to form part of the three-stage scheme, which might enable the assimilation of cloud system information into numerical weather prediction (NWP) models in the future.

In summary, this thesis represents a thorough and self-contained investigation into the use of neural networks and other current techniques for meteorological satellite image interpretation. A wide range of methods have been applied to each step in the process from raw spectral information to the large-scale definition of cloud regions, and the results represent a significant contribution to the field of research in general, and to the UK Meteorological Office project in particular.

8.1 Future work

The most promising avenue for future work would be the continuation of the research carried out on synoptic-scale analysis, with the incorporation of the large-scale regions into the bottom-up scheme proposed by Pankiewicz [71, 73] and Peak and Tag [82]. The eventual goal would be to assimilate this information into the NWP model.

There are several more immediate possibilities for improving the results of the level set segmentation method. Although the method did ignore most of the very small regions, there were still some that should be eliminated, for example by setting a threshold on the area of the region. Alternatively, a measure such as the sum of the posterior probabilities of all pixels within the region could be calculated. This would provide a measure of the significance of the region, and a threshold could be set on this to remove the insignificant regions.

At present, the level set method is seeded manually by selecting seed points on the posterior image; the automation of this process is an important area for future work. Due to the robust nature of the level set method, and the excellent topological properties it possesses, this should not be a difficult task. Only a single seed point is required inside each contiguous region of low posterior probability. That region may contain several high probability regions which are to be demarcated; the single level set will be able to locate the boundaries of all these regions.

The level set algorithm used in this thesis is designed for application to a single image (or posterior image). The MLP produces one posterior image for each class, and the level set algorithm has been applied to each posterior image independently of all the others. It may be possible to formulate an image term g_I that takes into account *all* of the posterior images at each pixel; for example a simple scheme would be to have g_I dependent in some way on the classification given by the MLP (which uses information from all of the posterior images).

Since each posterior image is treated independently at present, the cloud type regions may overlap to some degree. The basic level set algorithm has been extended to the treatment

of multiple interfaces by Sethian [103]. The interactions between different regions can be controlled by an *influence matrix*, so that they can be prevented from overlapping. In this way, simultaneous segmentation of all cloud types using non-overlapping interfaces could be performed.

Apart from the refinements to the level set scheme highlighted above, several other avenues for future exploration have been opened up. The dynamical properties of the snake method make it a candidate for cloud tracking. A snake surrounding a region of cloud defined in one time frame can form the initial guess for the same region in the next time frame. The image forces will cause it to surround the region in its new position; in this way the region is tracked through time. However, the topological limitations of the snake method would need to be overcome, and this may be a suitable task for the level set method instead.

It was mentioned in Section 6.2.1 that the multiscale MRF model used in this thesis allows the behaviour at various scales to be controlled, which is of critical importance since scale variation occurs naturally in images. By altering the behaviour at each scale, it may be possible to bias the model towards larger regions, or to create regions with smoother boundaries. It was also suggested in Chapter 6 that the Gaussian mixture model used may be insufficient to model the class-conditional probability densities needed in the algorithm. The use of a different density estimation technique could be investigated.

Finally, there is some scope for the use of a different colour map for the three-dimensional Kohonen unsupervised “classification” method of Chapter 6. At present, the coordinates of the winning unit are scaled linearly along all three axes to obtain the RGB triplet. Investigation could be carried out into the use of an alternative scaling method based on knowledge of human colour perception, so that regions of the three-dimensional colour cube where the eye is less sensitive can be assigned correspondingly fewer Kohonen units. It would also be possible to fix the position of a certain cloud type within the three-dimensional Kohonen map, thereby fixing its colour. In this way the process would be semi-supervised, with some degree of control over the eventual colour scheme.

Appendix A

The common weather satellites

The direct imaging satellites most commonly used for cloud investigations are the geostationary meteorological satellites (the European geostationary satellites are known as Meteosat) and the polar-orbiting National Oceanic and Atmospheric Administration (NOAA) meteorological satellites. The first operational meteorological satellite was TIROS-1¹, launched on 1 April 1960, which had on board a simple television camera. The current NOAA satellites are direct descendants of TIROS-1 and form part of the series termed TIROS-N/NOAA.

The investigations in this thesis make use of data from both the Meteosat satellites and the imagers on board the NOAA satellites, the so-called Advanced Very High Resolution Radiometers (AVHRR). This appendix describes the technical characteristics of these satellites. Brief mention will also be made of the Landsat satellite which is designed for imaging the Earth's surface but which has been used for several meteorological studies described in this thesis. Further detail on the satellites and scanners can be found in Pease [84] and Harris [44].

A.1 Meteosat satellites

The geostationary satellites are in orbit at approximately 35,800 km above a fixed point on the Earth's surface. Ideally there are five satellites positioned equidistant from each other

¹Television and Infra-Red Observation Satellite

Table A.1: Meteosat channel characteristics. The wavelengths stated are for the half-power points of the sensor response.

Channel	Wavelength band (μm)
Visible	0.5 – 0.9
Water Vapour	5.7 – 7.1
Infra-red	10.5 – 12.5

above the equator so that the entire surface of the Earth can be covered. In normal operation, the USA provides two of these satellites, and Japan, Russia and Europe one each.

The Meteosat satellite is cylindrically-shaped, with the axis parallel to the Earth's axis of rotation, and it is most easily stabilized by spinning it rapidly about its axis. With this arrangement, the images are built up by scanning the Earth's surface with a mirror that is tilted in steps from one pole to the other so that during each satellite rotation an adjacent strip of the Earth is scanned. An entire image can be produced once every 30 minutes. There are three imaging radiometers on board: visible, infra-red and water-vapour, summarized in Table A.1.

Although the temporal resolution is good, the main drawback with the images is the relatively low spatial resolution, in part due to the large distance from the Earth. The resolution at the sub-satellite point is 2.5 km for the visible channel and 5 km for the infra-red and water-vapour channels. A further drawback is that the Earth's curvature reduces resolution and distorts the image so that the image is only really useful between the equator and about 65° latitude.

This information is correct for the Meteosat-4 and Meteosat-5 satellites which were operational at the time of the capture of the images used for this thesis. Meteosat-5 took over from Meteosat-4 on 4 February 1994. The most recent Meteosat satellite is Meteosat-7, which was launched on 2 September 1997, although Meteosat-6 is still currently the operational satellite, with Meteosat-5 on standby.

Table A.2: AVHRR/2 channel characteristics. The wavelengths stated are for the half-power points of the sensor response.

Channel	Wavelength band (μm)
1	0.58 – 0.68
2	0.72 – 1.10
3	3.55 – 3.93
4	10.30 – 11.30
5	11.50 – 12.50

A.2 NOAA satellites

The polar-orbiting NOAA satellites travel in a low orbit passing over the poles at an altitude of about 850 km. Two are in operation at any one time. The satellite instruments build up an image by scanning from side to side normal to the direction of travel.

The orbit of the satellites is nearly sun-synchronous, *i.e.* almost fixed relative to the sun with the Earth spinning around beneath them so that the angle between the orbital plane and the sun-Earth line remains approximately constant. This means that the satellite keeps an almost constant local time, with a 12 hour shift between ascending and descending nodes. Therefore a particular point receives a pass every 12 hours, either on an ascending or descending node. The local time gets approximately 18 minutes earlier each day. The orbital period is about 102 minutes, equating to 14.2 orbits per day so that with the rotating Earth, each successive crossing of the equator is 25.4° , or 2850 km, further west.

Because of the low orbit, each swathe of the image is only about 3000 km wide. Successive swathes therefore just overlap at the equator, with the overlap increasing pole-ward so that effectively more than two passes a day are available at higher latitudes. Since the Earth rotates beneath the orbit, the satellite covers the entire surface twice a day. In practice, two polar-orbiting satellites are used with orbits at right angles to provide a pass at least every six hours.

Five spectral channels are available, summarized in Table A.2, and although the temporal resolution is not as good as for Meteosat, the spatial resolution is better, being 1.1 km at the

sub-satellite point for all channels. Channels 1 and 2 are both in or near the visible range of humans; channels 4 and 5 are in the infra-red; channel 3 lies in the near infra-red. The data in Table A.2 are for the second generation of the AVHRR instrument (hence AVHRR/2), which was first flown in 1981. The first generation AVHRR sensor had only four spectral channels. For simplicity, the term AVHRR in this thesis will imply AVHRR/2.

A.3 The Landsat satellite

The Landsat satellite is designed for mapping the surface of the Earth, rather than the clouds above it, and hence is not strictly a weather satellite. However the data are used for several studies reviewed in this thesis, and therefore the satellite is briefly described in this section. The information is correct for the Landsat satellites employed for those studies [55, 101].

The multi-spectral scanner (MSS) on board the Landsat satellite has a spatial resolution of 57 m, much higher than either the NOAA or the Meteosat satellites. This is a separate sensor to the Thematic Mapper also on board the later Landsat satellites. Like the NOAA satellite, the Landsat satellite is in sun-synchronous orbit. It obtains data for a north-south strip 185 km wide. The MSS has four narrow spectral bands at $(0.5 - 0.6) \mu\text{m}$, $(0.6 - 0.7) \mu\text{m}$, $(0.7 - 0.8) \mu\text{m}$ and $(0.8 - 1.1) \mu\text{m}$. It is apparent, therefore, that these bands cover only the visible or near-visible part of the spectrum.

Appendix B

The Multiscale Random Field model

This appendix describes the theory of the Multiscale Random Field (MSRF) model of Bouman and Shapiro [12], which is used in Chapter 6. Many approaches to Bayesian image segmentation have used maximum *a posteriori* (MAP) estimation in conjunction with Markov Random Fields (MRFs). There are a number of drawbacks with this approach. Exact MAP estimates cannot be computed, approximate MAP estimates are computationally expensive to evaluate, parameter estimation is difficult, and MRFs have a limited ability to describe large-scale behaviour. The MSRF model attempts to address these problems. All diagrams are reproduced from Bouman and Shapiro [12].

B.1 Multiscale segmentation approach

The random field X is the image that is to be segmented into distinct regions on a second random field Y , as depicted in Figure B.1. Realizations of these random fields are denoted by lower-case letters x and y . Individual pixels in X are denoted by X_s , where s is a member of a two-dimensional lattice of points S . Each label in Y can be one of M possible classes.

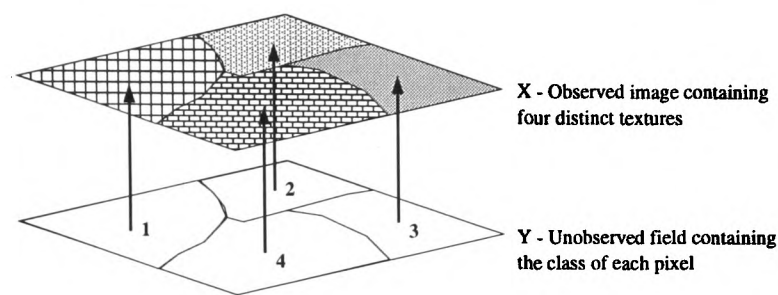


Figure B.1: The structure of a doubly-stochastic random field.

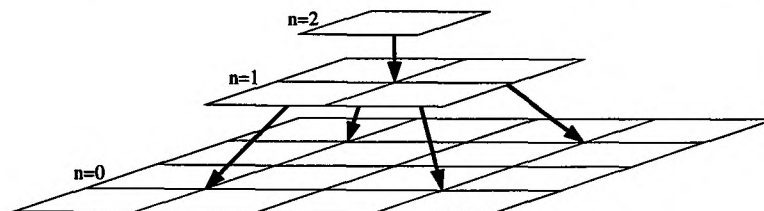


Figure B.2: Pyramid structure of the MSRF. The random field at each resolution is dependent on the coarser scale above it. The scale $n = 0$ is pixel resolution.

The dependence of the observed pixels on their labels is through the data model $p(x | y)$.

Prior knowledge of the shapes and sizes of regions is modelled by $p(y)$.

B.1.1 Multiscale Random Field model

Figure B.2 shows the basis of the structure for the Multiscale Random Field (MSRF) model. Note that the scale structure is for the segmentation Y , not the image itself, X . At each scale, n , the segmentation is denoted by the random field $Y^{(n)}$, and the set of lattice points by $S^{(n)}$. $Y^{(0)}$ is the finest scale, corresponding to the original image, and each point in the coarser-scale field corresponds to four points in the field below it.

The fundamental assumption of the MSRF is that the sequence of fields, from coarse to fine scale, form a Markov chain. Formally,

$$p(y^{(n)} | y^{(l)}, l > n) = p(y^{(n)} | y^{(n+1)}). \quad (\text{B.1})$$

Correspondingly, the exclusive dependence of X on $Y^{(0)}$ implies that

$$p(x | y^{(n)}, n > 0) = p(x | y^{(0)}). \quad (\text{B.2})$$

B.1.2 Sequential MAP estimation

Generally, Bayesian estimators attempt to minimize the average cost of an erroneous segmentation:

$$\hat{y} = \arg \min_y \mathcal{E}[C(Y, y) \mid X = x] \quad (\text{B.3})$$

where $C(Y, y)$ is the cost of estimating the true segmentation Y by the approximate segmentation y . The MAP estimate is the solution to Equation B.3 when the cost function is given by $C_{\text{MAP}}(Y, y) = 1 - \delta(Y - y)$, where $\delta(Y - y)$ is 1 when $Y = y$ and 0 otherwise. This means that the MAP estimate maximizes the probability that *all* pixels are correctly classified. If this is extended to the MSRF model then:

$$\begin{aligned} C_{\text{MAP}}(Y, y) &= 1 - \delta(Y - y) \\ &= 1 - \prod_{n=0}^L \delta(Y^{(n)} - y^{(n)}) \end{aligned} \quad (\text{B.4})$$

where L is the coarsest scale in Y . Thus the cost function is 1 if a labelling error occurs at any scale. Consequently, this function assigns equal cost to a single misclassified pixel at $n = 0$ and approximately 256 pixels at $n = 4$. Ideally, the cost function should assign a greater cost to segmentations with larger regions of misclassified pixels. $C_{\text{MAP}}(Y, y)$ can therefore be replaced with a Sequential MAP (SMAP) function:

$$C_{\text{SMAP}}(Y, y) = \frac{1}{2} + \sum_{n=0}^L 2^{n-1} C_n(Y, y) \quad (\text{B.5})$$

where

$$C_n(Y, y) = 1 - \prod_{i=n}^L \delta(Y^{(i)} - y^{(i)}). \quad (\text{B.6})$$

With some manipulation, it can be shown that this leads to the following recursive equations:

$$\begin{aligned}
 \hat{y}^{(L)} &= \arg \max_{y^{(L)}} \log p(y^{(L)} | x) \\
 &= \arg \max_{y^{(L)}} \log \left\{ p(x | y^{(L)}) p(y^{(L)}) \right\} \\
 &= \arg \max_{y^{(L)}} \log p(x, y^{(L)})
 \end{aligned} \tag{B.7}$$

$$\begin{aligned}
 \hat{y}^{(n)} &= \arg \max_{y^{(n)}} \log p(y^{(n)} | \hat{y}^{(n+1)}, x) \\
 &= \arg \max_{y^{(n)}} \log \left\{ p(x | y^{(n)}) p(y^{(n)} | \hat{y}^{(n+1)}) \right\} \\
 &= \arg \max_{y^{(n)}} \log p(x, y^{(n)} | \hat{y}^{(n+1)}).
 \end{aligned} \tag{B.8}$$

The use of these recursion relations will be discussed in the next section. Note that $\hat{y}^{(n+1)}$ is not a random variable but is a given set of parameters. By using the Markov properties of Y , and assuming that $Y^{(L)}$ is uniformly distributed, the SMAP recursion can be rewritten as:

$$\hat{y}^{(L)} = \arg \max_{y^{(L)}} \log p(x | y^{(L)}) \tag{B.9}$$

$$\hat{y}^{(n)} = \arg \max_{y^{(n)}} \left\{ \log p(x | y^{(n)}) + \log p(y^{(n)} | \hat{y}^{(n+1)}) \right\}. \tag{B.10}$$

B.2 Segmentation algorithm

The specific models used for X and Y will now be defined. This will be done by specifying the conditional density $p(x | y^{(0)})$, together with the coarse to fine scale transition densities $p(y^{(n)} | y^{(n+1)})$. An adaptive segmentation algorithm is then described which estimates the parameters of the MSRF during the segmentation process.

The model is restricted to having observed pixel grey-levels that are conditionally independent of each other, given their class labels *i.e.* the spatial relationship of neighbouring grey-levels in the image is not modelled. This implies that the texture will not be used as a

discriminating feature. However, each pixel X_s can in fact be a feature vector of dimension D , containing textural information as described in Chapter 3. The conditional density then has the form

$$p(x | y^{(0)}) = \prod_{s \in S^{(0)}} p(x_s | y_s^{(0)}). \quad (\text{B.11})$$

Since each pixel X_s is a D -dimensional vector, a multivariate Gaussian mixture model is used for the class-conditional probabilities $p(x_s | y_s^{(0)})$, with parameters set using the Expectation Maximization (EM) algorithm (see Appendix C for a detailed treatment of the Gaussian mixture model and the EM algorithm). The model order for each class (the number of Gaussians used) is chosen using the Rissanen criteria [89].

The model for Y is restricted to have two important properties. Firstly, the pixels in $Y^{(n)}$ will be conditionally independent, given the pixels in $Y^{(n+1)}$. Secondly, each pixel $Y_s^{(n)}$ will be dependent only on a local neighbourhood of pixels at the next coarser scale. This set of neighbouring locations to s will be denoted by ∂s . The transition distribution is then

$$p(y^{(n)} | y^{(n+1)}) = \prod_{s \in S^{(n)}} p(y_s^{(n)} | y_{\partial s}^{(n+1)}). \quad (\text{B.12})$$

Note that although the pixels in $Y^{(n)}$ are conditionally independent, they are still highly correlated since they share coarser-scale neighbours.

The choice of neighbourhood ∂s is discussed in the following sections. Two types are explored. The first is a simple quadtree model and allows exact calculation of the SMAP segmentation. The second has a more complex neighbourhood structure and produces less blocky segmentations. Unfortunately, the latter model does not allow exact SMAP estimations, and therefore a hybrid model is used.

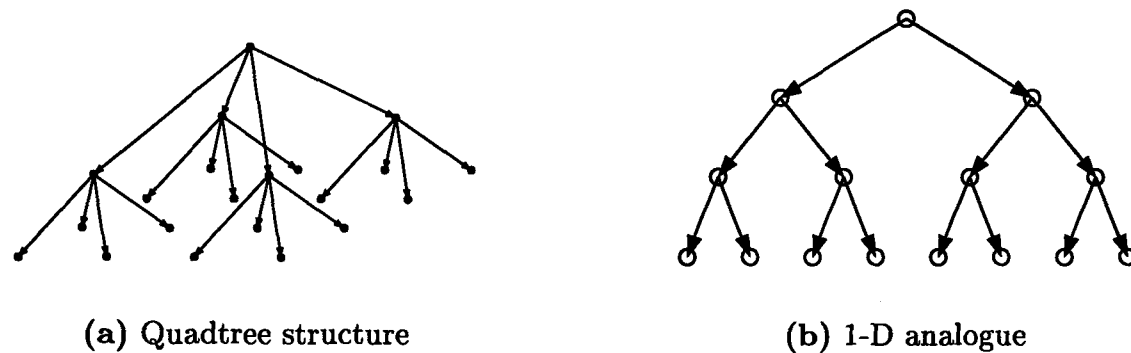


Figure B.3: The quadtree structure, and its one-dimensional analogue.

B.2.1 Quadtree model

The structure of this model is shown in Figure B.3(a), and the one-dimensional analogue is shown in Figure B.3(b). Each point is dependent on only one point in the coarser scale. The coarser-scale point is called the parent of s and will be denoted by the function $d(s)$. The following transition function is chosen for Equation B.12:

$$p(y_s^{(n)} = m \mid y_{d(s)}^{(n+1)} = k) = \theta_{n,0} \delta_{m,k} + \frac{1 - \theta_{n,0}}{M} \quad (\text{B.13})$$

where M is the number of possible classes. The parameter $\theta_{n,0} \in [0, 1]$ is the probability that the labelling will remain the same from scale $(n+1)$ to n . If a class change does occur, it is equally likely to be any one of the remaining classes.

An important property of the quadtree model is that the conditional distribution of X , given $Y^{(n)}$, has a product form that can be computed using a simple fine-to-coarse recursion. Let $x_s^{(n)}$ be the set of *leaves* of the quadtree that are on the branch starting at $s \in S^{(n)}$. It can be shown that

$$p(x \mid y^{(n)}) = \prod_{s \in S^{(n)}} p(x_s^{(n)} \mid y_s^{(n)}) \quad (\text{B.14})$$

and that the density functions $p(x_s^{(n)} \mid y_s^{(n)})$ may be computed using the following recursion:

$$p(x_s^{(n+1)} \mid y_s^{(n+1)} = k) = \prod_{r \in d^{-1}(s)} \sum_{m=1}^M p(x_r^{(n)} \mid y_r^{(n)} = m) p(y_r^{(n)} = m \mid y_{d(r)}^{(n+1)} = k). \quad (\text{B.15})$$

Converting to log-likelihoods, substituting Equation B.13 into Equation B.15, and using the expression

$$l_s^{(n)}(k) = \log p(x_s^{(n)} | y_s^{(n)} = k) \quad (\text{B.16})$$

the following new expression for the recursion is obtained:

$$l_s^{(0)}(k) = \log p(x_s | y_s^{(0)} = k) \quad (\text{B.17})$$

$$l_s^{(n+1)}(k) = \sum_{r \in d^{-1}(s)} \log \left\{ \theta_{n,0} \exp\{l_r^{(n)}(k)\} + \frac{1 - \theta_{n,0}}{M} \sum_{m=1}^M \exp\{l_r^{(n)}(m)\} \right\}. \quad (\text{B.18})$$

To perform the SMAP segmentation, these log-likelihood functions must be computed at every point in the quadtree. Then the segmentation can be computed using Equation B.10:

$$\hat{y}^{(n)} = \arg \max_{y^{(n)}} \sum_{s \in S^{(n)}} \left\{ l_s^{(n)}(y_s^{(n)}) + \log p(y_s^{(n)} | \hat{y}_{\partial s}^{(n+1)}) \right\} \quad (\text{B.19})$$

which is easily evaluated by minimizing with respect to each pixel individually, in a coarse-to-fine manner:

$$\hat{y}_s^{(n)} = \arg \max_{1 \leq k \leq M} \left\{ l_s^{(n)}(k) + \log p(y_s^{(n)} = k | \hat{y}_{\partial s}^{(n+1)}) \right\}. \quad (\text{B.20})$$

Although this quadtree model results in an exact expression for computing the SMAP estimate, it does not completely capture some aspects of image behaviour. In the following section, an augmented model which improves on the quadtree is introduced.

B.2.2 Pyramidal graph model

A major disadvantage of the quadtree model is that pixels that are spatially adjacent may not have common neighbours at the next coarser scale. Therefore the model does not enforce continuity of region boundaries when they pass across branches of the quadtree. This weakness may be corrected by increasing the number of neighbours at the next coarser level for

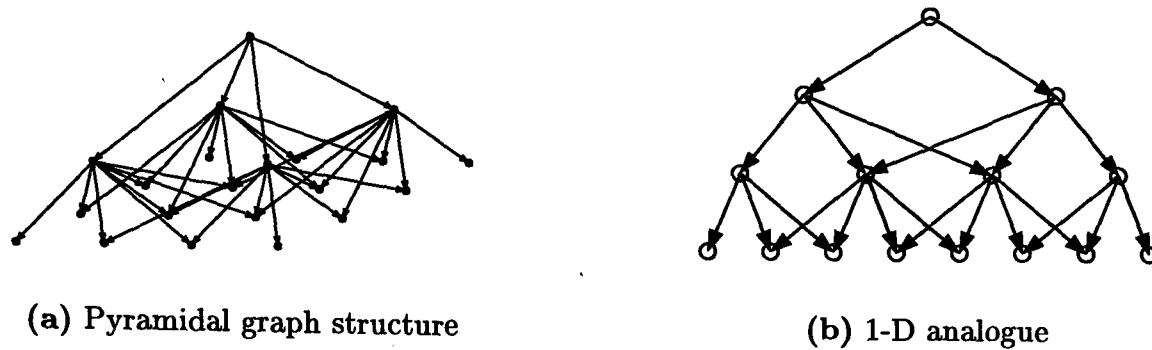


Figure B.4: The augmented pyramidal graph structure used for the MSRF model and its one-dimensional analogue.

each pixel. Such a pyramidal graph structure is shown in Figure B.4. Each point has three neighbours at the next coarser scale. If a pixel s is denoted as having coordinates (i, j) , then the neighbours are given by:

$$s_1 = (\lfloor i/2 \rfloor, \lfloor j/2 \rfloor) \quad (\text{B.21})$$

$$s_2 = (\lfloor i/2 \rfloor, \lfloor j/2 \rfloor) + (\text{odd}(i), 0) \quad (\text{B.22})$$

$$s_3 = (\lfloor i/2 \rfloor, \lfloor j/2 \rfloor) + (0, \text{odd}(j)) \quad (\text{B.23})$$

where $\lfloor \cdot \rfloor$ rounds down to the nearest integer, and $\text{odd}(i) = 1$ if i is odd, and -1 if i is even.

The transition function used for this model is a natural extension of that used for the quadtree model:

$$\tilde{p}(y_s^{(n)} = m \mid y_{\partial s}^{(n+1)} = i, j, k) = \frac{\theta_{n,1}}{7} (3\delta_{m,i} + 2\delta_{m,j} + 2\delta_{m,k}) + \frac{1 - \theta_{n,1}}{M}. \quad (\text{B.24})$$

Note that the notation $\tilde{p}$ is used to distinguish it from the quadtree model. The disadvantage of this model is that the likelihood function for the labels does not have a product form as was the case for the quadtree model in Equation B.14. Therefore there is no simple fine-to-coarse recursion of the form of Equation B.15 for computing the necessary likelihood of the image x , given the labels $y^{(n)}$.

For computations at a single scale n , this problem is overcome by assuming that the pyramid has a quadtree structure for scales finer than n (so fine-to-coarse calculation of the

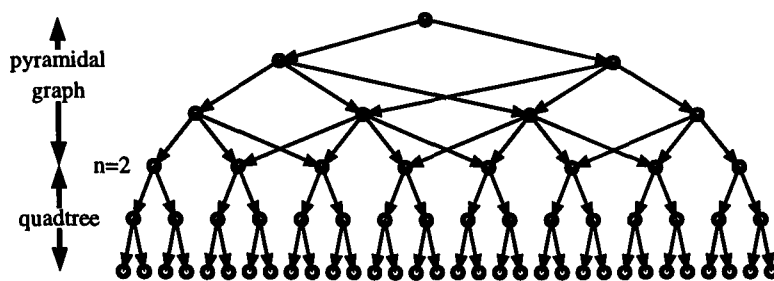


Figure B.5: The one-dimensional analogue to the hybrid graph structure. Scales greater than the current one (in this case, $n > 2$) use a pyramidal graph structure, and scales $n \leq 2$ use a quadtree structure.

$l_s^{(n)}$ functions remains the same), and a pyramidal graph structure for coarser scales (for coarse-to-fine calculations of $\hat{y}^{(n)}$). A one-dimensional analogue is shown in Figure B.5 for $n = 2$. Using this hybrid model, the conditional likelihood of Equation B.10 then has the computable form

$$\log p(x, y^{(n)} | \hat{y}^{(n+1)}) = \sum_{s \in S^{(n)}} l_s^{(n)}(y_s^{(n)}) + \log \tilde{p}(y_s^{(n)} | \hat{y}_{\partial s}^{(n+1)}) \quad (\text{B.25})$$

which results in the following recursion for the SMAP estimate of $Y^{(n)}$:

$$\hat{y}_s^{(L)} = \arg \max_{1 \leq k \leq M} l_s^{(L)}(k) \quad (\text{B.26})$$

$$\hat{y}_s^{(n)} = \arg \max_{1 \leq k \leq M} \left\{ l_s^{(n)}(k) + \log \tilde{p}(y_s^{(n)} = k | \hat{y}_{\partial s}^{(n+1)}) \right\} \quad (\text{B.27})$$

where $\tilde{p}$ is the transition function of Equation B.24, and $l_s^{(n)}(k)$ are computed using the recursion of Equation B.18. Note that Equation B.27 is the same as Equation B.20 with $\tilde{p}$ instead of p . In summary, the first step of the algorithm is a fine-to-coarse computation given by Equation B.18. The second step is a coarse-to-fine computation given by Equation B.27.

B.2.3 Parameter estimation

The parameters $\theta_n = [\theta_{n,0}, \theta_{n,1}]$ used in the algorithm described in previous sections have been assumed to be known. This is of course not the case, and they must be estimated during the segmentation process. The parameter $\theta_{n,0}$ is required for fine-to-coarse operations computing the log-likelihood functions, while $\theta_{n,1}$ is used in the coarse-to-fine SMAP segmentation. Both

of these parameter vectors will be estimated during the coarse-to-fine operations. The overall procedure will therefore require two full passes, composed of the following steps:

1. Perform fine-to-coarse operations using initial parameter values $\theta_{n,0} = 1$.
2. Estimate the parameters $\theta_{n,0}$ and $\theta_{n,1}$ during coarse-to-fine operations.
3. Perform fine-to-coarse operations using the estimated parameters.
4. Re-estimate the parameters $\theta_{n,1}$ during the final coarse-to-fine segmentation.

The parameter estimation procedure will now be derived. The value of the parameter $\theta_{n,1}$ that maximizes the likelihood of the image data, x , must be estimated, given the previous coarse-scale segmentation $\hat{y}^{(n+1)}$:

$$\hat{\theta}_{n,1} = \arg \max_{\theta_{n,1} \in \Omega} p(x \mid \hat{y}^{(n+1)}, \theta_{n,1}) \quad (\text{B.28})$$

where Ω is the set of valid parameter values. Recall that $\hat{y}^{(n+1)}$ is considered to be a known parameter, not a random variable. Using the hybrid pyramid model, the log-likelihood function has the specific form

$$\begin{aligned} \mathcal{L}(\theta_{n,1}) &= \log p(x \mid \hat{y}^{(n+1)}, \theta_{n,1}) \\ &= \log \left(\sum_{y^{(n)}} p(x \mid y^{(n)}) p(y^{(n)} \mid \hat{y}^{(n+1)}, \theta_{n,1}) \right) \\ &= \sum_{s \in S^{(n)}} \log \left(\sum_{k=1}^M \exp(l_s^{(n)}(k)) \tilde{p}(y_s^{(n)} = k \mid \hat{y}_{\partial s}^{(n+1)}, \theta_{n,1}) \right). \end{aligned} \quad (\text{B.29})$$

Note that $\tilde{p}$ does not depend on s (see Equation B.24). This likelihood may be maximized as a function of $\theta_{n,1}$ by using the EM algorithm. This algorithm requires the use of $\log p(y^{(n)}, x \mid \hat{y}^{(n+1)}, \theta_{n,1})$, which in EM terminology is the log-likelihood of the *complete* data given the parameters; x is the *observable* data, and $y^{(n)}$ is the *unobservable* data (see Section C.2 for a description of the EM algorithm). The expectation over the unobservable data $y^{(n)}$ of this log-likelihood is taken, conditioned on a previous estimate of the parameters; formally the

following is computed (see Equation C.11):

$$\begin{aligned} Q(\theta_{n,1}^{\text{old}}, \theta_{n,1}^{\text{new}}) &= \mathcal{E} \left[\log \left(p(y^{(n)}, x \mid \hat{y}^{(n+1)}, \theta_{n,1}^{\text{new}}) \right) \mid x, \hat{y}^{(n+1)}, \theta_{n,1}^{\text{old}} \right] \\ &= \mathcal{E} \left[\log \left(p(x \mid Y^{(n)}) p(Y^{(n)} \mid \hat{y}^{(n+1)}, \theta_{n,1}^{\text{new}}) \right) \mid x, \hat{y}^{(n+1)}, \theta_{n,1}^{\text{old}} \right] \end{aligned} \quad (\text{B.30})$$

where the expectation is taken over the random object $Y^{(n)}$. The EM algorithm is then effectively an iterative procedure for finding a sequence of parameters that converge to $\hat{\theta}_{n,1}$:

$$\theta_{n,1}^{\text{new}} = \arg \max_{\theta_{n,1}^{\text{new}} \in \Omega} Q(\theta_{n,1}^{\text{old}}, \theta_{n,1}^{\text{new}}). \quad (\text{B.31})$$

To further simplify this expression, a sufficient statistic for the transition distribution of Equation B.24 must be formulated. This statistic counts the number of distinct transitions that occur from a particular set of coarse-scale neighbours $y_{\partial s}^{(n+1)}$ to a point $y_s^{(n)}$. Define the statistic T such that

$$\log \tilde{p}(y_s^{(n)} = m \mid \theta_{n,1}, y_{\partial s}^{(n+1)} = i, j, k) = \sum_{l=0}^1 \sum_{h=0}^2 T_{l,h}(m \mid i, j, k) V_{l,h}(\theta_{n,1}) \quad (\text{B.32})$$

where T and V have the functional forms

$$T_{l,h}(m \mid i, j, k) = \begin{cases} 1 & \text{if } \delta_{m,i} = l \text{ and } \delta_{m,j} + \delta_{m,k} = h \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.33})$$

and

$$V_{l,h}(\theta_{n,1}) = \log \left(\frac{\theta_{n,1}}{7} (3l + 2h) + \frac{1 - \theta_{n,1}}{M} \right). \quad (\text{B.34})$$

Note that Equation B.32 is just that of Equation B.24 in a different form. There are two possible values for l (0 and 1 from $\delta_{m,i}$), and three possible values for h (0, 1 and 2 from $\delta_{m,j} + \delta_{m,k}$), hence the summation indices in Equation B.32. The index l denotes the count of pixels that stay the same from coarse to fine for the direct pyramid link s_1 of Equation B.21

for *one* such set of branches. The index h denotes the count of pixels that stay the same for the diagonal links s_2 and s_3 for *one* set of branches. Since $\tilde{p}$ does not depend on the pixel position s , $T_{l,h}$ can be estimated from all the pixels, and used as a global function for that scale. By substituting Equation B.32 into the expression for Q , the following simpler expression for Equation B.31 is obtained:

$$\theta_{n,1}^{\text{new}} = \arg \max_{\theta_{n,1}^{\text{new}} \in \Omega} \left\{ \sum_{l=0}^1 \sum_{h=0}^2 \bar{T}_{l,h}(\theta_{n,1}^{\text{old}}) V_{l,h}(\theta_{n,1}^{\text{new}}) \right\} \quad (\text{B.35})$$

where $\bar{T}_{l,h}$ is defined by:

$$\begin{aligned} \bar{T}_{l,h}(\theta_{n,1}) &= \sum_{s \in S^{(n)}} \mathcal{E} \left[T_{l,h}(Y_s^{(n)} \mid \hat{y}_{\partial s}^{(n+1)}) \mid x, \hat{y}^{(n+1)}, \theta_{n,1} \right] \\ &= \sum_{s \in S^{(n)}} \frac{\sum_{k=1}^M T_{l,h}(k \mid \hat{y}_{\partial s}^{(n+1)}) \exp(l_s^{(n)}(k)) \tilde{p}(y_s^{(n)} = k \mid \hat{y}_{\partial s}^{(n+1)}, \theta_{n,1})}{\sum_{k=1}^M \exp(l_s^{(n)}(k)) \tilde{p}(y_s^{(n)} = k \mid \hat{y}_{\partial s}^{(n+1)}, \theta_{n,1})}. \end{aligned} \quad (\text{B.36})$$

Finally, the parameters $\theta_{n,0}$ are estimated using $\bar{T}$ resulting from convergence of the EM algorithm:

$$\hat{\theta}_{n,0} = \frac{\sum_{h=0}^2 \bar{T}_{1,h}}{\sum_{l=0}^1 \sum_{h=0}^2 \bar{T}_{l,h}}. \quad (\text{B.37})$$

These values are then used in the second pass of the fine-to-coarse computations.

The complete SMAP algorithm with parameter estimation as used for the results of Section 6.2.2 is summarized in Figure B.6. This corresponds to the procedure outlined at the start of the section.

1. Set initial parameter values for all n : $\hat{\theta}_{n,0} = 1$ and $\hat{\theta}_{(L-1),1} = 0.5$.
2. Compute likelihood functions using Equation B.18 and the parameters $\hat{\theta}_{n,0}$.
3. Compute $\hat{y}^{(L)}$ using Equation B.26.
4. For scales $n = (L - 1)$ to $n = 0$:
 - (a) Use the EM algorithm for the iterative computation of $\hat{\theta}_{n,1}$ and $\bar{T}$.
 - (b) Compute $\hat{\theta}_{n,0}$ using Equation B.37.
 - (c) Compute $\hat{y}^{(n)}$ using Equation B.27.
5. Repeat Steps 2 through to 4.

Figure B.6: A summary of the complete SMAP segmentation algorithm with parameter estimation.

Appendix C

Additional mathematics

This appendix describes in detail some of the mathematics required for the methods used mainly in Chapter 7. Section C.1 introduces the Gaussian mixture model, the Expectation Maximization (EM) method is described in Section C.2, and Section C.3 applies the EM method to the Gaussian mixture model. The theory from these three sections forms the basis of the Gaussian region model developed in Section 7.3.1. Section C.4 describes the numerical methods used for the active contour model (or snake) method of Section 7.3. Finally, Section C.5 contains further details of the level set method of Section 7.4 and describes how it is implemented using fast numerical techniques.

C.1 Gaussian mixture model

A probability density function $p(\mathbf{x})$ can be modelled by a linear combination of M component densities $p(\mathbf{x} | j)$, where j indicates the component:

$$p(\mathbf{x}) = \sum_{j=1}^M p(\mathbf{x} | j)P(j). \quad (\text{C.1})$$

The coefficients $P(j)$ are called the *mixing parameters*, and satisfy the constraints:

$$\sum_{j=1}^M P(j) = 1 \quad (\text{C.2})$$

$$0 \leq P(j) \leq 1. \quad (\text{C.3})$$

They can be regarded as the prior probabilities of the data point (or feature vector) $\mathbf{x}$ having been generated from mixture component j . Similarly the component densities $p(\mathbf{x} | j)$ are normalized so that

$$\int p(\mathbf{x} | j) d\mathbf{x} = 1 \quad (\text{C.4})$$

and can hence be regarded as class-conditional probabilities (see Chapter 2). For a mixture of Gaussians, $p(\mathbf{x} | j)$ is given by:

$$p(\mathbf{x} | j) = (2\pi)^{-d/2} |\Sigma_j|^{-1/2} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_j)^T (\Sigma_j)^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right] \quad (\text{C.5})$$

where d is the dimension of $\mathbf{x}$, and $\boldsymbol{\mu}_j$ and Σ_j are the mean vector and covariance matrix respectively of Gaussian j . For an entire dataset of N points $\mathbf{x}^n$, the negative log-likelihood of all the points in the dataset for the defined mixture distribution is:

$$E = -\ln \mathcal{L} = -\sum_{n=1}^N \ln p(\mathbf{x}^n) = -\sum_{n=1}^N \ln \left\{ \sum_{j=1}^M p(\mathbf{x}^n | j) P(j) \right\}. \quad (\text{C.6})$$

This energy function is then minimized in order to find the best (in the maximum likelihood sense) estimates of the adjustable parameters $P(j)$, $\boldsymbol{\mu}_j$ and Σ_j . It is not possible to find the solution analytically since it is not known from which mixture component each data point $\mathbf{x}^n$ was generated [8]. The data consisting only of the $\mathbf{x}^n$ are known as *incomplete* data, being only the *observable* component of the *complete* data. The indices of the mixture components to which the $\mathbf{x}^n$ belong are the *unobservable* data, which are not known.

Therefore some approximating algorithm must be used to maximize the likelihood. The Expectation Maximization (EM) algorithm will be considered for this purpose, and is designed to be used with incomplete data. General EM theory will be considered in Section C.2, and applied to the Gaussian mixture model in Section C.3. The derivations are based on those found in Dempster *et al* [28] and Bishop [8].

C.2 The Expectation Maximization algorithm

Let X be a measure space of observable data, and Y one of unobservable data. Suppose that there is a set of parameters, denoted by ϕ , on which the distributions of X and Y depend. The aim is to find the ϕ that maximizes $L(x, \phi) = \log[p(x | \phi)]$ for a given set x of observed data. For instance for the Gaussian mixture example, x would be the set of all $\mathbf{x}^n$, and ϕ would consist of the parameters $P(j)$, μ_j and Σ_j . The Expectation Maximization algorithm produces a series of estimates for ϕ , each one producing a greater value for L . The procedure can then be run to convergence of ϕ .

Given an estimate ϕ^0 for ϕ , the aim is to produce another one, ϕ^1 , that is better in the sense that $L(x, \phi^1) > L(x, \phi^0)$. Now

$$p(x, y | \phi) = p(x | \phi)p(y | x, \phi) \quad (\text{C.7})$$

and therefore

$$L(x, \phi) = \log[p(x, y | \phi)] - \log[p(y | x, \phi)]. \quad (\text{C.8})$$

Now consider the following:

$$\begin{aligned}
 \log[p(x | \phi^1)] &= \log[p(x | \phi^1)] \int p(y | x, \phi^0) dy \\
 &= \int \log[p(x | \phi^1)] p(y | x, \phi^0) dy \\
 &= \mathcal{E}[\log[p(x | \phi^1)] | x, \phi^0] \\
 &= \mathcal{E}[L(x, \phi^1) | x, \phi^0]
 \end{aligned} \tag{C.9}$$

where $\mathcal{E}$ is the expectation operator over the distribution for y . Using Equation C.8, this becomes

$$L(x, \phi^1) = Q(\phi^0, \phi^1) - H(\phi^0, \phi^1) \tag{C.10}$$

where

$$Q(\phi^0, \phi^1) = \mathcal{E}[\log[p(x, y | \phi^1)] | x, \phi^0] \tag{C.11}$$

$$H(\phi^0, \phi^1) = \mathcal{E}[\log[p(y | x, \phi^1)] | x, \phi^0]. \tag{C.12}$$

It can be shown that $H(\phi^0, \phi^1) \leq H(\phi^0, \phi^0)$ for any ϕ^1 :

$$\begin{aligned}
 H(\phi^0, \phi^1) - H(\phi^0, \phi^0) &= \int \log \left[\frac{p(y | x, \phi^1)}{p(y | x, \phi^0)} \right] p(y | x, \phi^0) dy \\
 &\leq \int \left[\frac{p(y | x, \phi^1)}{p(y | x, \phi^0)} - 1 \right] p(y | x, \phi^0) dy \\
 &\leq \int p(y | x, \phi^1) dy - \int p(y | x, \phi^0) dy \\
 &\leq 0
 \end{aligned} \tag{C.13}$$

since $\log a \leq (a - 1)$ and both integrals evaluate to 1. The discrete case is similar to the continuous one.

Therefore, picking any ϕ^1 such that $Q(\phi^0, \phi^1) > Q(\phi^0, \phi^0)$ will increase the value of L ; in

particular, finding the ϕ^1 that maximizes Q will produce a better estimate for the parameters. This procedure can then be iterated until a satisfactory estimate of ϕ is produced. Explicitly, choosing ϕ^1 to maximize

$$Q(\phi^0, \phi^1) = \int \log[p(x, y | \phi^1)] p(y | x, \phi^0) dy \quad (\text{C.14})$$

in the continuous case, or

$$Q(\phi^0, \phi^1) = \sum_y \log[p(x, y | \phi^1)] P(y | x, \phi^0) \quad (\text{C.15})$$

in the discrete case, will produce a better estimate for ϕ .

C.3 Application of EM to the Gaussian mixture model

Let x now be the set of N data points $\mathbf{x}^n$ from Section C.1, ϕ be the set of parameters $P(j)$, μ_j and Σ_j of an M -component mixture model, and y be a set of labels such that y^n is the mixture component from which $\mathbf{x}^n$ came. For ease of notation, a superscript of 0 will indicate distributions given $\phi = \phi^0$, and similarly for a superscript of 1. Thus

$$\begin{aligned} \log[p^1(x, y)] &= \sum_{n=1}^N \log[P^1(y^n) p^1(\mathbf{x}^n | y^n)] \\ &= \sum_{n=1}^N \sum_{j=1}^M \delta_{jy^n} \log[P^1(j) p^1(\mathbf{x}^n | j)] \end{aligned} \quad (\text{C.16})$$

where δ is the Dirac delta function, such that $\delta_{ab} = 1$ if $a = b$, and 0 otherwise. This gives, using Equation C.15, the following expression for Q :

$$Q = \sum_y \sum_{n=1}^N \sum_{j=1}^M \delta_{jy^n} \log[P^1(j) p^1(\mathbf{x}^n | j)] P^0(y | x). \quad (\text{C.17})$$

$P^0(y | x)$ can be written as the following product:

$$P^0(y | x) = \prod_{i=1}^N P^0(y^i | \mathbf{x}^i) \quad (\text{C.18})$$

so that Q becomes:

$$Q = \sum_{y^1=1}^M \cdots \sum_{y^N=1}^M \sum_{n=1}^N \sum_{j=1}^M \delta_{jy^n} \log[P^1(j)p^1(\mathbf{x}^n | j)] \prod_{i=1}^N P^0(y^i | \mathbf{x}^i). \quad (\text{C.19})$$

Reorder the summations, take the logarithm term outside the summations over y , and consider the following:

$$\begin{aligned} & \sum_{y^1=1}^M \cdots \sum_{y^N=1}^M \delta_{jy^n} \prod_{i=1}^N P^0(y^i | \mathbf{x}^i) \\ &= \sum_{y^1=1}^M \cdots \sum_{y^{n-1}=1}^M \sum_{y^{n+1}=1}^M \cdots \sum_{y^N=1}^M P^0(j | \mathbf{x}^n) \prod_{i \neq n}^N P^0(y^i | \mathbf{x}^i) \\ &= P^0(j | \mathbf{x}^n) \sum_{y^1=1}^M \cdots \sum_{y^{n-1}=1}^M \sum_{y^{n+1}=1}^M \cdots \sum_{y^N=1}^M \prod_{i \neq n}^N P^0(y^i | \mathbf{x}^i). \end{aligned} \quad (\text{C.20})$$

Writing $(y^1, \dots, y^{n-1}, y^{n+1}, \dots, y^N) = (z^1, \dots, z^{N-1})$ for ease of notation, Equation C.20 can be simplified by considering the following:

$$\begin{aligned} \sum_{z^1=1}^M \cdots \sum_{z^{N-1}=1}^M \prod_{i=1}^N P^0(z^i | \mathbf{x}^i) &= \sum_{z^1=1}^M P^0(z^1 | \mathbf{x}^1) \left(\cdots \left(\sum_{z^{N-1}=1}^M P^0(z^{N-1} | \mathbf{x}^{N-1}) \right) \right) \\ &= 1 \end{aligned} \quad (\text{C.21})$$

since each bracketed sum equates to 1, from the innermost sum outwards. Therefore

$$Q = \sum_{n=1}^N \sum_{j=1}^M P^0(j | \mathbf{x}^n) \log[P^1(j)p^1(\mathbf{x}^n | j)]. \quad (\text{C.22})$$

In the case of a Gaussian mixture, from Equation C.5,

$$p^1(\mathbf{x}^n | j) = (2\pi)^{-d/2} |\Sigma_j^1|^{-1/2} \exp \left[-\frac{1}{2} (\mathbf{x}^n - \mu_j^1)^T (\Sigma_j^1)^{-1} (\mathbf{x}^n - \mu_j^1) \right]. \quad (\text{C.23})$$

Therefore, taking logarithms,

$$\log[p^1(\mathbf{x}^n | j)] = -\frac{d}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma_j^1| - \frac{1}{2} (\mathbf{x}^n - \mu_j^1)^T (\Sigma_j^1)^{-1} (\mathbf{x}^n - \mu_j^1). \quad (\text{C.24})$$

Thus, to within an additive constant, the expression for Q is now:

$$Q = \sum_{n=1}^N \sum_{j=1}^M P^0(j | \mathbf{x}^n) \left[\log P^1(j) - \frac{1}{2} \log |\Sigma_j^1| - \frac{1}{2} (\mathbf{x}^n - \mu_j^1)^T (\Sigma_j^1)^{-1} (\mathbf{x}^n - \mu_j^1) \right] \quad (\text{C.25})$$

which must now be maximized with respect to the $P^1(j)$, the μ_j^1 and the Σ_j^1 parameters. There is also the constraint on the $P^1(j)$ parameters to be taken into account, namely that they must sum to 1. Therefore introduce a Lagrange multiplier λ and maximize the quantity

$$\hat{Q} = Q + \lambda \left(\sum_{j=1}^M P^1(j) - 1 \right) \quad (\text{C.26})$$

so that λ is also included in the set of variables over which the maximization takes place.

Setting $\partial \hat{Q} / \partial [P^1(j)] = 0$ gives

$$\sum_{n=1}^N \left[\frac{P^0(j | \mathbf{x}^n)}{P^1(j)} \right] + \lambda = 0. \quad (\text{C.27})$$

Multiply by $P^1(j)$ and sum over j to get

$$\sum_{j=1}^M \sum_{n=1}^N P^0(j | \mathbf{x}^n) + \lambda = 0. \quad (\text{C.28})$$

Exchanging the order of summation then gives $\lambda = -N$. Substituting this back into Equation (C.27) gives

$$P^1(j) = \frac{1}{N} \sum_{n=1}^N P^0(j | \mathbf{x}^n). \quad (\text{C.29})$$

This gives an update rule for the new estimate of $P^1(j)$ in terms of the old estimate, $P^0(j)$.

The $P^0(j | \mathbf{x}^n)$ are calculated using Bayes' theorem. Setting $\partial \hat{Q} / \partial \boldsymbol{\mu}_j^1 = 0$ gives

$$-\frac{1}{2} \sum_{n=1}^N P^0(j | \mathbf{x}^n) \frac{\partial}{\partial \boldsymbol{\mu}_j^1} [(\mathbf{x}^n - \boldsymbol{\mu}_j^1)^T (\boldsymbol{\Sigma}_j^1)^{-1} (\mathbf{x}^n - \boldsymbol{\mu}_j^1)] = 0 \quad (\text{C.30})$$

and

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^1} [(\mathbf{x}^n - \boldsymbol{\mu}_j^1)^T (\boldsymbol{\Sigma}_j^1)^{-1} (\mathbf{x}^n - \boldsymbol{\mu}_j^1)] = -2(\mathbf{x}^n - \boldsymbol{\mu}_j^1)^T (\boldsymbol{\Sigma}_j^1)^{-1} \quad (\text{C.31})$$

so substituting this, taking the transpose and rearranging gives

$$\boldsymbol{\mu}_j^1 = \frac{\sum_{n=1}^N \mathbf{x}^n P^0(j | \mathbf{x}^n)}{\sum_{n=1}^N P^0(j | \mathbf{x}^n)}. \quad (\text{C.32})$$

To find $\partial \hat{Q} / \partial \boldsymbol{\Sigma}_j^1$, instead solve for $\mathbf{R}_j = (\boldsymbol{\Sigma}_j^1)^{-1}$. Differentiating Equation (C.26) with respect to $\mathbf{R}_j$ gives

$$\frac{\partial \hat{Q}}{\partial \mathbf{R}_j} = \sum_{n=1}^N P^0(j | \mathbf{x}^n) \left[\frac{1}{2} \frac{\partial}{\partial \mathbf{R}_j} \log |\mathbf{R}_j| - \frac{1}{2} \frac{\partial}{\partial \mathbf{R}_j} [(\mathbf{x}^n - \boldsymbol{\mu}_j^1)^T \mathbf{R}_j (\mathbf{x}^n - \boldsymbol{\mu}_j^1)] \right] \quad (\text{C.33})$$

and, using

$$\frac{\partial}{\partial \mathbf{R}_j} \log |\mathbf{R}_j| = \mathbf{R}_j^{-T} \quad (\text{C.34})$$

$$\frac{\partial}{\partial \mathbf{R}_j} [(\mathbf{x}^n - \boldsymbol{\mu}_j^1)^T \mathbf{R}_j (\mathbf{x}^n - \boldsymbol{\mu}_j^1)] = (\mathbf{x}^n - \boldsymbol{\mu}_j^1)(\mathbf{x}^n - \boldsymbol{\mu}_j^1)^T \quad (\text{C.35})$$

and the fact that $\mathbf{R}_j$ is symmetrical, the following is obtained:

$$\Sigma_j^1 = \frac{\sum_{n=1}^N P^0(j | \mathbf{x}^n) (\mathbf{x}^n - \mu_j^1) (\mathbf{x}^n - \mu_j^1)^T}{\sum_{n=1}^N P^0(j | \mathbf{x}^n)}. \quad (\text{C.36})$$

Equations C.29, C.32 and C.36 give the iterative update rules for each parameter. Iteration continues until the parameters converge on stable values.

C.4 Numerical methods for the active contour model

Representing the position of the snake parametrically by $\mathbf{v}(s) = \{x(s), y(s)\}$, the following energy function is defined, as it was for Equation 7.1:

$$E_{\text{snake}}^* = \int_0^1 E_{\text{snake}}(\mathbf{v}(s)) ds = \int_0^1 E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s)) ds \quad (\text{C.37})$$

where E_{image} is the energy due to image forces acting on the snake at every point, and E_{int} is the internal energy due to bending and stretching.

The internal snake energy can be written

$$E_{\text{int}} = (\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2)/2 \quad (\text{C.38})$$

where the subscript s denotes a derivative with respect to the position s along the snake. The snake energy is composed of a first order term controlled by $\alpha(s)$ and a second order term controlled by $\beta(s)$. The former creates the stretching force while the latter creates the bending force. Setting $\beta(s)$ to zero allows the snake to develop corners.

Moving to a discrete formulation where there are n points considered along the snake, the energy function of Equation C.37 becomes:

$$E_{\text{snake}}^* = \sum_{i=1}^n E_{\text{int}}(i) + E_{\text{image}}(i). \quad (\text{C.39})$$

Approximating the derivatives of Equation C.38 by finite differences, the following is obtained:

$$E_{\text{int}}(i) = \alpha_i |\mathbf{v}_i - \mathbf{v}_{i-1}|^2 / 2h^2 + \beta_i |\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}|^2 / 2h^4 \quad (\text{C.40})$$

where $\mathbf{v}_i = \{x_i, y_i\} = \{x(ih), y(ih)\}$. The parametric curve is closed, so define $\mathbf{v}_0 = \mathbf{v}_n$. To minimize the energy function of Equation C.39, derivatives have to be calculated with respect to x and y and then set to zero:

$$\begin{aligned} & \alpha_i (\mathbf{v}_i - \mathbf{v}_{i-1}) - \alpha_{i+1} (\mathbf{v}_{i+1} - \mathbf{v}_i) \\ & + \beta_{i-1} (\mathbf{v}_{i-2} - 2\mathbf{v}_{i-1} + \mathbf{v}_i) - 2\beta_i (\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}) \\ & + \beta_{i+1} (\mathbf{v}_i - 2\mathbf{v}_{i+1} + \mathbf{v}_{i+2}) + \mathbf{f}_i = 0 \end{aligned} \quad (\text{C.41})$$

where $\mathbf{f}_i = \{\partial E_{\text{image}} / \partial x_i, \partial E_{\text{image}} / \partial y_i\}$. However, in Section 7.3 the image forces $\mathbf{f}_i$ are not derived from E_{image} but are given directly.

Equation C.41 applies to all n nodes on the snake, and contains within it two equations, one for each dimension. Forming a vector $\mathbf{x}$ from the x components of all the $\mathbf{v}_i$, and likewise for $\mathbf{y}$, the above equations can be written in matrix form:

$$\mathbf{A}\mathbf{x} + \mathbf{f}_x = 0 \quad (\text{C.42})$$

$$\mathbf{A}\mathbf{y} + \mathbf{f}_y = 0 \quad (\text{C.43})$$

where $\mathbf{f}_x$ and $\mathbf{f}_y$ are vectors composed of the x and y components respectively of the image force at each snake node. The matrix $\mathbf{A}$ is a circular banded matrix [98] (a sparse matrix) with a band width of five.

To solve Equations C.42 and C.43, the right-hand sides of the equations are set equal to the product of a step size and the negative time derivative of the left-hand sides, and $\mathbf{f}_x$ and

$\mathbf{f}_y$ are assumed to be constant during a time step:

$$\mathbf{A}\mathbf{x}(t) + \mathbf{f}_x(t-1) = -\nu\{\mathbf{x}(t) - \mathbf{x}(t-1)\} \quad (\text{C.44})$$

$$\mathbf{A}\mathbf{y}(t) + \mathbf{f}_y(t-1) = -\nu\{\mathbf{y}(t) - \mathbf{y}(t-1)\} \quad (\text{C.45})$$

where ν is the step size and t is the time step. At equilibrium, the time derivatives on the right-hand sides vanish, resulting in a solution of Equations C.42 and C.43. These are rearranged to give an explicit update rule for the node positions:

$$\mathbf{x}(t) = (\mathbf{A} + \nu\mathbf{I})^{-1}\{\mathbf{x}(t-1) - \mathbf{f}_x(t-1)\} \quad (\text{C.46})$$

$$\mathbf{y}(t) = (\mathbf{A} + \nu\mathbf{I})^{-1}\{\mathbf{y}(t-1) - \mathbf{f}_y(t-1)\}. \quad (\text{C.47})$$

An iterative method has therefore been formulated, with new positions for the snake nodes being computed at each iteration. When equilibrium is reached, one of the solutions to Equations C.42 and C.43 has been found, with the snake attracted to a salient feature in the image.

C.5 Additional mathematics for the level set method

This section contains further details of the level set method of Section 7.4 and describes how it is implemented using fast numerical techniques. The notation used here is the same as that set out in Section 7.4.1. The derivation follows that of Sethian and Strain [104], Malladi *et al* [64] and Malladi and Sethian [62].

C.5.1 Outline of the algorithm

Consider again Equation 7.24:

$$\psi_t + F|\nabla\psi| = 0. \quad (\text{C.48})$$

The initial curve $\gamma(0)$ is embedded in the function $\psi(x, y, 0)$ using the scheme of Equation 7.19, which is:

$$\psi(x, y, 0) = \pm \text{distance from } (x, y) \text{ to } \gamma(0) \quad (\text{C.49})$$

where the minus (plus) sign is chosen if (x, y) is inside (outside) the initial curve $\gamma(0)$. The position of the evolving front $\gamma(t)$ must now be iterated through time using Equation C.48, discretizing both time and space. The time step is denoted by Δt , numbering each time step by n . Define ψ_{ij}^n and F_{ij}^n at discrete grid points (x_{ij}, y_{ij}) on a square lattice with spacing h , so that $\psi_{ij}^n \equiv \psi(x_{ij}, y_{ij}, n\Delta t)$. Finite difference approximations may be used to approximate both the spatial and temporal derivatives so that Equation C.48 becomes

$$\frac{\psi_{ij}^{n+1} - \psi_{ij}^n}{\Delta t} + F_{ij}^n |\nabla_h \psi_{ij}^n| = 0 \quad (\text{C.50})$$

where forward differences in time have been used. A different finite difference operator for $\nabla_h \psi_{ij}^n$ must be used to approximate the spatial gradient, which is described below.

As in Section 7.4.2, split F into the advection term F_A , the geometry term F_G , and the image term g_I , such that $F = g_I(F_A + F_G)$, and rewrite Equation C.48 as

$$\psi_t = -g_I(F_A |\nabla \psi| + F_G |\nabla \psi|). \quad (\text{C.51})$$

The geometry term F_G serves to diffuse sharp gradients and forces ψ to stay smooth for all time [102]. However, should F_G be set to zero, only the constant advection term is left. In this case, corners will develop in the front, leaving the normal to the front undefined. Therefore an entropy-satisfying approximation [103] to the difference operator must be used

for the advection term, described in Sethian and Strain [104], to produce

$$\begin{aligned} \psi_{ij}^{n+1} = \psi_{ij}^n - g_I F_A \Delta t & \left(\{\max(D_x^- \psi_{ij}^n, 0)\}^2 + \{\min(D_x^+ \psi_{ij}^n, 0)\}^2 \right. \\ & \left. + \{\max(D_y^- \psi_{ij}^n, 0)\}^2 + \{\min(D_y^+ \psi_{ij}^n, 0)\}^2 \right)^{1/2} - g_I F_G \Delta t |\nabla \psi| \end{aligned} \quad (\text{C.52})$$

where the forward, central and backward difference operators [103], in one dimension with respect to x , are defined as

$$D_x^+ \psi_i = \frac{\psi_{i+1} - \psi_i}{h} \quad (\text{C.53})$$

$$D_x^0 \psi_i = \frac{\psi_{i+1} - \psi_{i-1}}{2h} \quad (\text{C.54})$$

$$D_x^- \psi_i = \frac{\psi_i - \psi_{i-1}}{h} \quad (\text{C.55})$$

respectively. A straightforward central difference approximation can be used for $|\nabla \psi|$ for the geometry term F_G of Equation C.52, as well as for the calculation of the curvature term κ within F_G , which is given in Equation 7.27. The use of *min* and *max* in Equation C.52 is to ensure that the derivatives are always taken in an *upwind* fashion with respect to the direction of motion of the front. The backward difference operator of Equation C.55 is referred to as an *upwind* operator because it uses values upwind of the direction of information propagation in the positive x direction [103]. However, when the front is moving in the other direction, the forward difference operator is used. Due to the initialization of ψ using Equation C.49 with negative values inside the front, and the in-to-out motion due to F_A , the spatial gradient of ψ gives the direction of motion of the front. The appropriate upwind operators are switched in by the presence of the *min* and *max* functions.

Equation C.52 therefore enables a scheme to evolve $\psi(x, y, t)$, and therefore $\gamma(t)$, through time on the discrete grid. The basic level set algorithm is summarized in Figure C.1. The calculation of the speed function in Step 3 requires further discussion. As seen in Section 7.4.2, the speed function may be written as $F = g_I(F_A + F_G)$. The image term g_I has meaning only on the curve $\gamma(t)$ as it was designed to force the propagating zero level set $\{\psi = 0\}$ to

1. Set $n = 0$.
2. Initialize ψ_{ij}^0 for each grid point using Equation C.49.
3. Compute the speed function F_{ij}^n for each grid point.
4. Calculate ψ_{ij}^{n+1} from ψ_{ij}^n and F_{ij}^n for each grid point using the upwind, finite difference scheme of Equation C.52.
5. Display the new zero level set $\{\psi = 0\}$ using the scheme presented in Sethian and Strain [104].
6. Replace n by $n + 1$ and return to Step 3.

Figure C.1: The basic level set algorithm. The time step is denoted by Δt , numbering each time step by n . Define ψ_{ij}^n and F_{ij}^n at discrete grid points (x_{ij}, y_{ij}) on a square lattice with spacing h . Finite difference approximations are used to approximate both the spatial and temporal derivatives. The user decides when to terminate the algorithm. See text for details.

a complete halt in the neighbourhood of an object boundary.

Therefore a globally-defined image term must be constructed. This is done by extending g_I from the front $\gamma(t)$ by setting g_I at any point P to be equal to g_I at the point Q on $\gamma(t)$ that is closest to P . However, this construction can create a discontinuous velocity extension away from the (possibly convoluted) $\gamma(t)$, causing non-zero level sets to become uneven. In order to remedy this, the level set function ψ is re-initialized every fixed number of time steps using Equation C.49 in order to keep the level sets spaced evenly around the front.

C.5.2 Fast numerical techniques

The global extension of the image term g_I outlined in the previous section incurs a large computational cost [64]. This section outlines two techniques for reducing the computation required, firstly the narrow band extension, and secondly the fast marching algorithm.

C.5.2.1 The narrow band extension

An efficient algorithm can be constructed by observing that the front can be moved by updating ψ only at a small set of points in the neighbourhood of the zero level set instead of

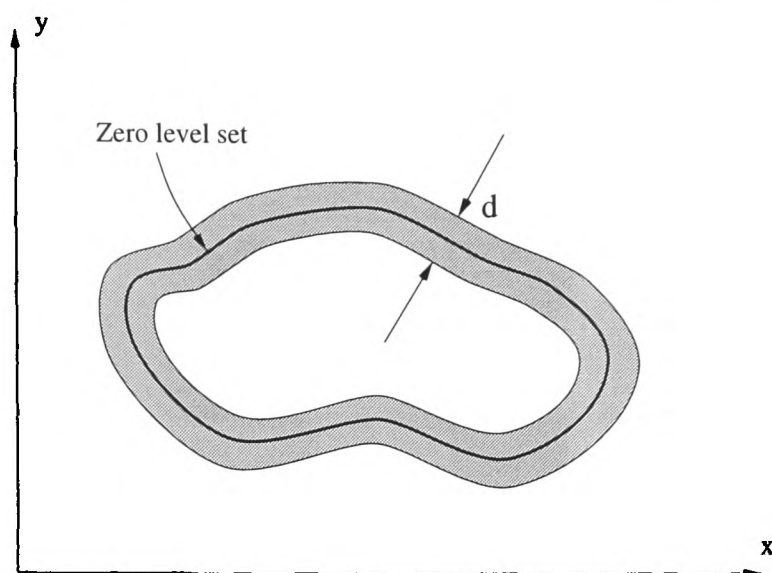


Figure C.2: The narrow band extension for the level set method. The band is of width d , around the zero level set $\{\psi = 0\}$.

updating it at all points on the grid. The set of points that are updated lie in the narrow band of width d illustrated in Figure C.2, so that the boundaries of the narrow band are a distance $d/2$ from the zero level set. The width d determines how many grid points are included in the band. The zero level set inside the band moves until it collides with the boundary, at which point a new narrow band is built, and ψ is re-initialized using Equation C.49 as it was in the previous section. The same procedure continues with the new narrow band. In this way, the computationally-expensive extension of the image term g_I need only be done for the points in the narrow band. As a further optimization, it may be assumed that g_I as defined in Equation 7.28 holds for the narrow band region, so that extending g_I is not necessary.

C.5.2.2 The fast marching method

Suppose that attention is restricted to the particular case of a front propagating with a speed F that is either always positive or always negative. This occurs when $F_G = 0$ so that the direction of motion is not affected by the local geometry of the front; only the advection term F_A and the image term g_I remain. In this case, a time-independent formulation can be constructed. Introduce $T(x, y)$, which is the time at which the propagating front crosses the point (x, y) . Then Equation C.48 becomes, in the time-independent formulation [103, 62],

$$|\nabla T| F = 1. \quad (\text{C.56})$$

Using the upwind, finite difference scheme for the gradient as described in Section C.5.1, this becomes

$$\begin{aligned} & \{\max(D_x^- T_{ij}, 0)\}^2 + \{\min(D_x^+ T_{ij}, 0)\}^2 \\ & + \{\max(D_y^- T_{ij}, 0)\}^2 + \{\min(D_y^+ T_{ij}, 0)\}^2 = \frac{1}{F_{ij}^2}. \end{aligned} \quad (\text{C.57})$$

A *fast marching* method [61, 62, 103] is used to solve this for all points on the grid, making use of a narrow band similar to that of the previous section. The key to constructing the algorithm is the observation that the upwind difference structure of Equation C.57 means that information propagates one way, that is from smaller values of T to larger values. Equation C.57 can be solved by building the solution outwards from the smallest value of T , that is $T = 0$.

The method is presented algorithmically in Figure C.4 with reference to the diagram in Figure C.3. The front is swept ahead in an upwind fashion by considering a set of points in the narrow band around the existing front, and marching this narrow band forwards, freezing values of existing points and bringing new ones into the narrow band. A proof that this method produces a viable solution to Equation C.57 is found in Sethian [103]. An efficient method for locating the grid point in the narrow band with the smallest value for T , using a min-heap data structure, is also described in Sethian [103].

C.5.2.3 Narrow band and fast marching combined

The condition for use of the fast marching method was that $F_G = 0$. With no geometry term there is no smoothness constraint on the propagating front, which is controlled only by the advection term F_A and the image term g_I which stops it in the neighbourhood of an object boundary. It is possible to use the fast marching method to propagate the front quickly for a set length of time, and then to switch to the full level set methodology (using the narrow band technique) in order to produce a smooth final contour. This is the method used for the results obtained in this thesis.

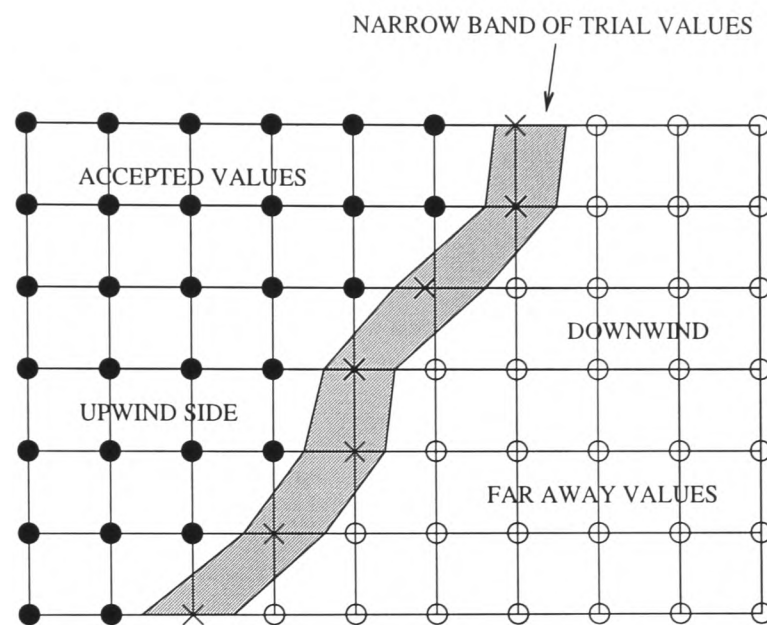


Figure C.3: A snapshot of a two-dimensional grid after several steps of the fast marching method. The front is marching from left (upwind) to right (downwind).

1. Let $(i_{\min}, j_{\min})$ be the point in *NarrowBand* with the smallest value for T .
2. Add the point $(i_{\min}, j_{\min})$ to *Accepted*; remove it from *NarrowBand*.
3. Tag as neighbours any points $(i_{\min} - 1, j_{\min})$, $(i_{\min} + 1, j_{\min})$, $(i_{\min}, j_{\min} - 1)$ and $(i_{\min}, j_{\min} + 1)$ that are not *Accepted*; if the neighbour is in *FarAway*, then remove it from that set and add it to *NarrowBand*.
4. Recompute the values of T at all neighbours according to Equation C.57, solving the quadratic equation and selecting the largest possible value as the solution.
5. Return to Step 1.

Figure C.4: The fast marching algorithm. Let *Accepted* be the set of all grid points at which the values of T are fixed. This is where the front has already passed. Let *NarrowBand* be the set of all grid points in the narrow band. Let *FarAway* be the set of all the rest of the grid points, where the marching front has not yet reached.

Appendix D

Full-size images

This appendix contains full-size versions of almost all of the images displayed throughout the thesis, showing both the visible and infra-red images, as well as details about the date and time of their capture. In other parts of the thesis, for reasons of space, either smaller versions had to be used or only the visible image was presented. Complete information can only be obtained by looking at both channels, and this is provided here.

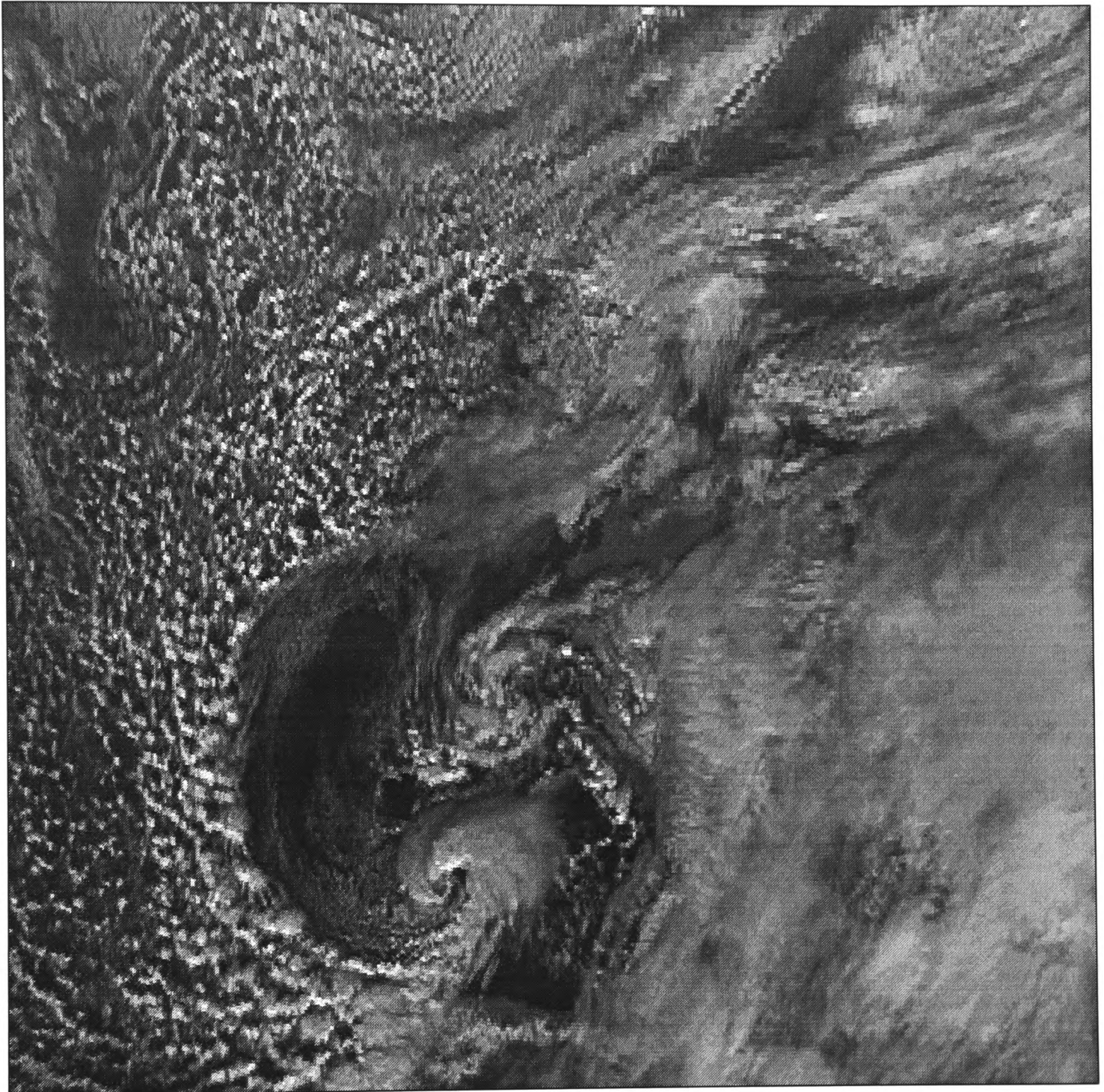


Figure D.1: The visible Meteosat image from 6 January 1994 at 1200 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding infra-red image is shown in Figure D.2.

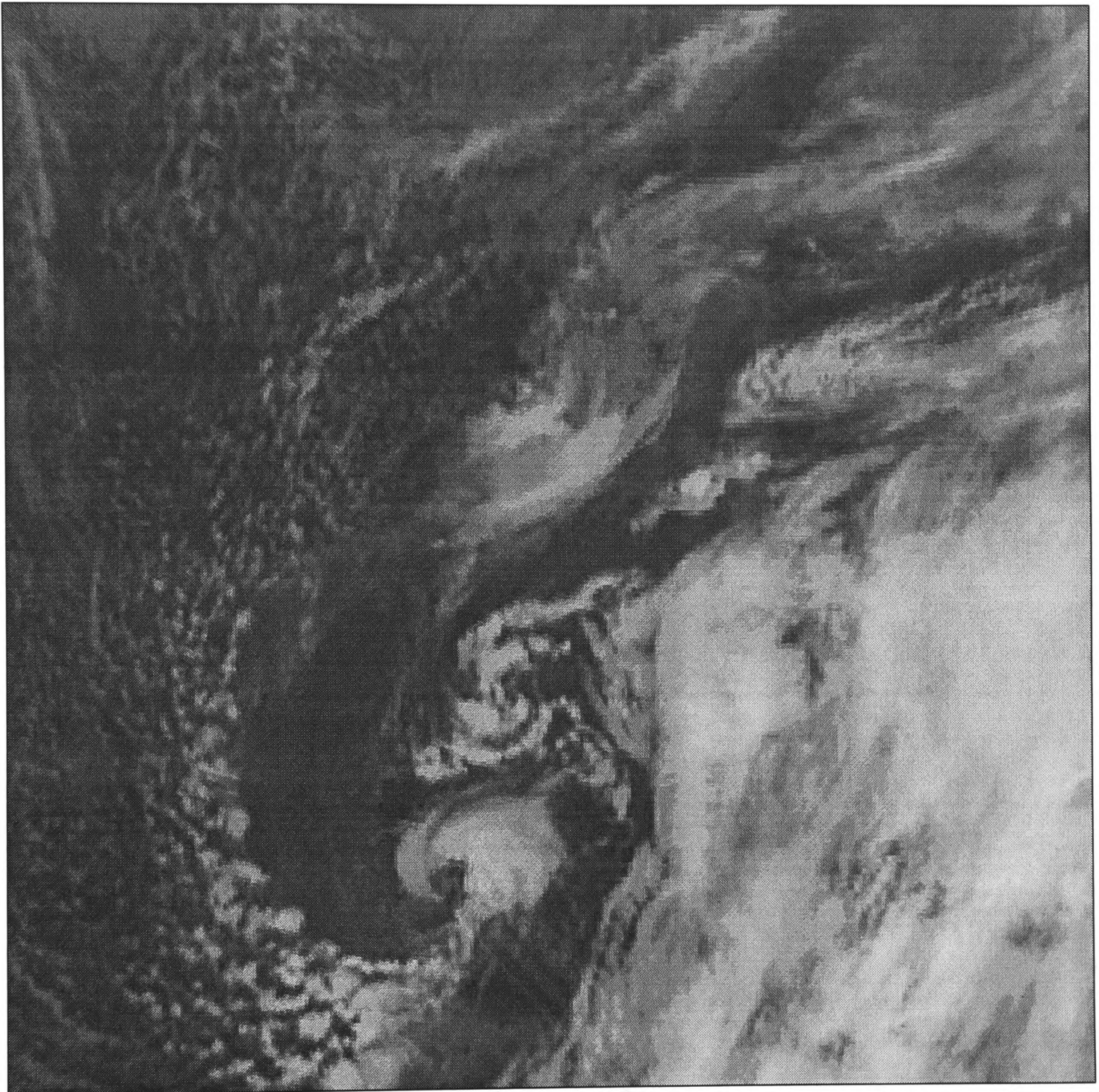


Figure D.2: The infra-red Meteosat image from 6 January 1994 at 1200 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding visible image is shown in Figure D.1.



Figure D.3: The visible Meteosat image from 3 April 1994 at 1600 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding infra-red image is shown in Figure D.4.

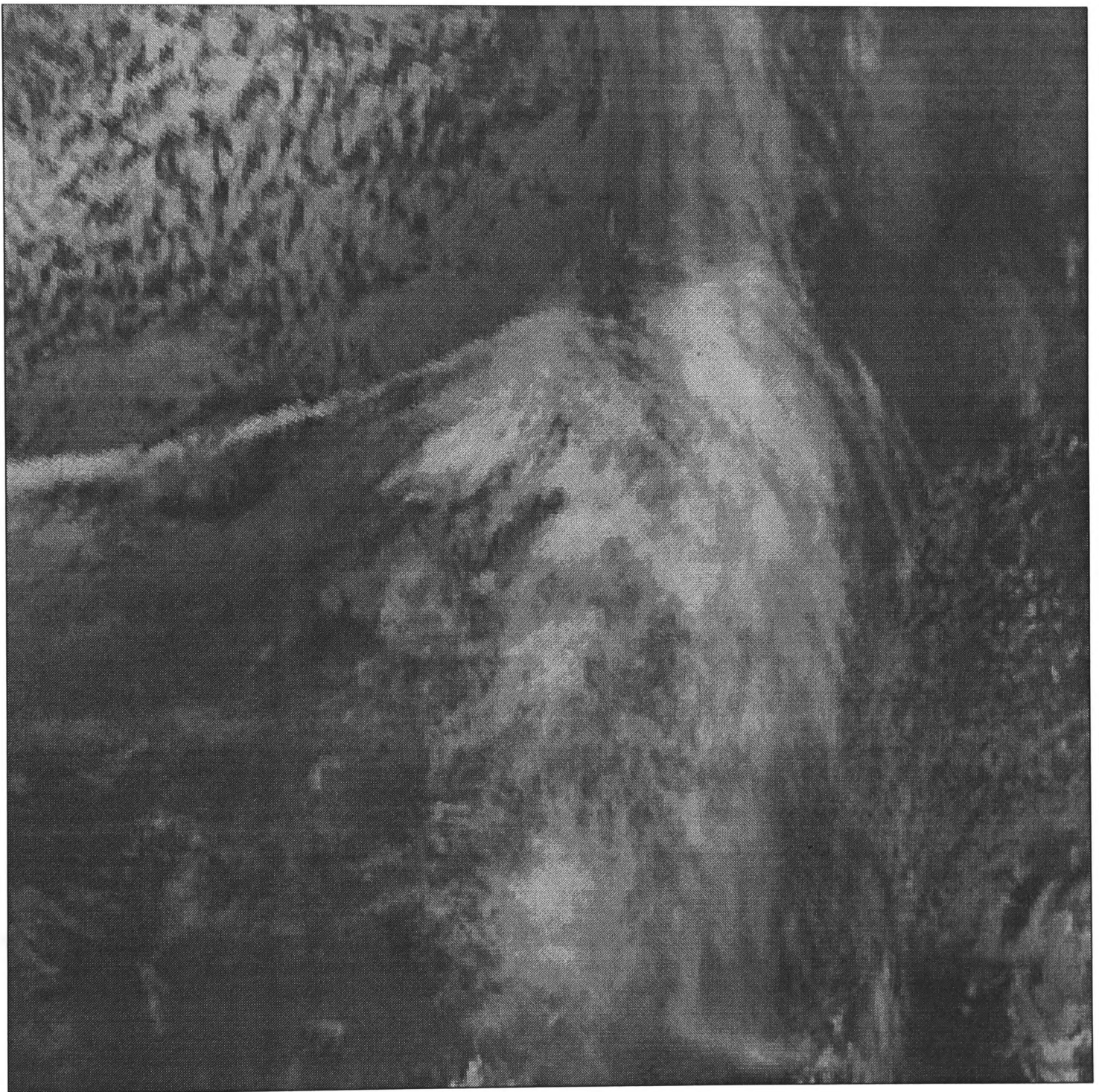


Figure D.4: The infra-red Meteosat image from 3 April 1994 at 1600 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding visible image is shown in Figure D.3.

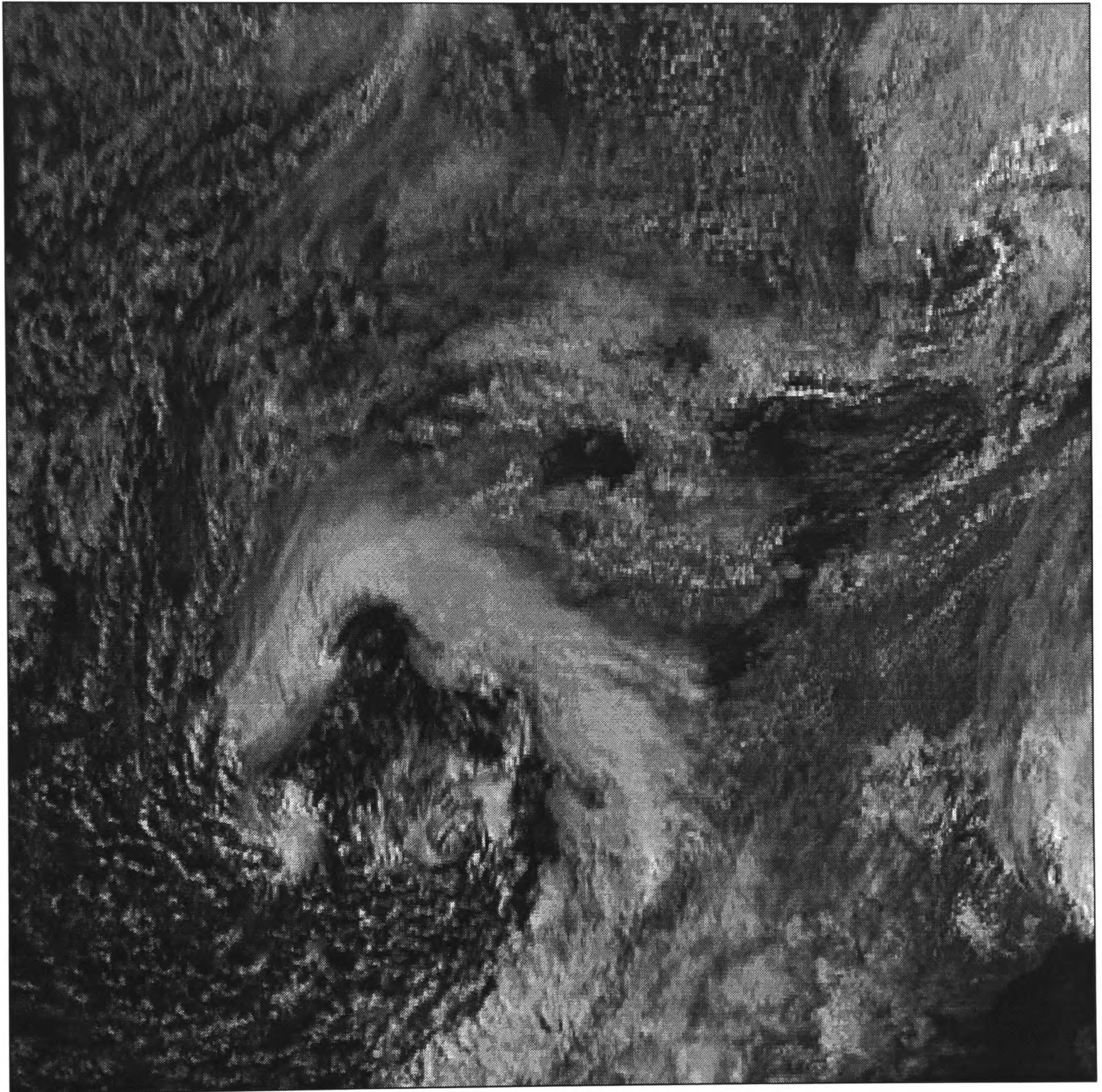


Figure D.5: The visible Meteosat image from 13 September 1994 at 1500 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding infra-red image is shown in Figure D.6.

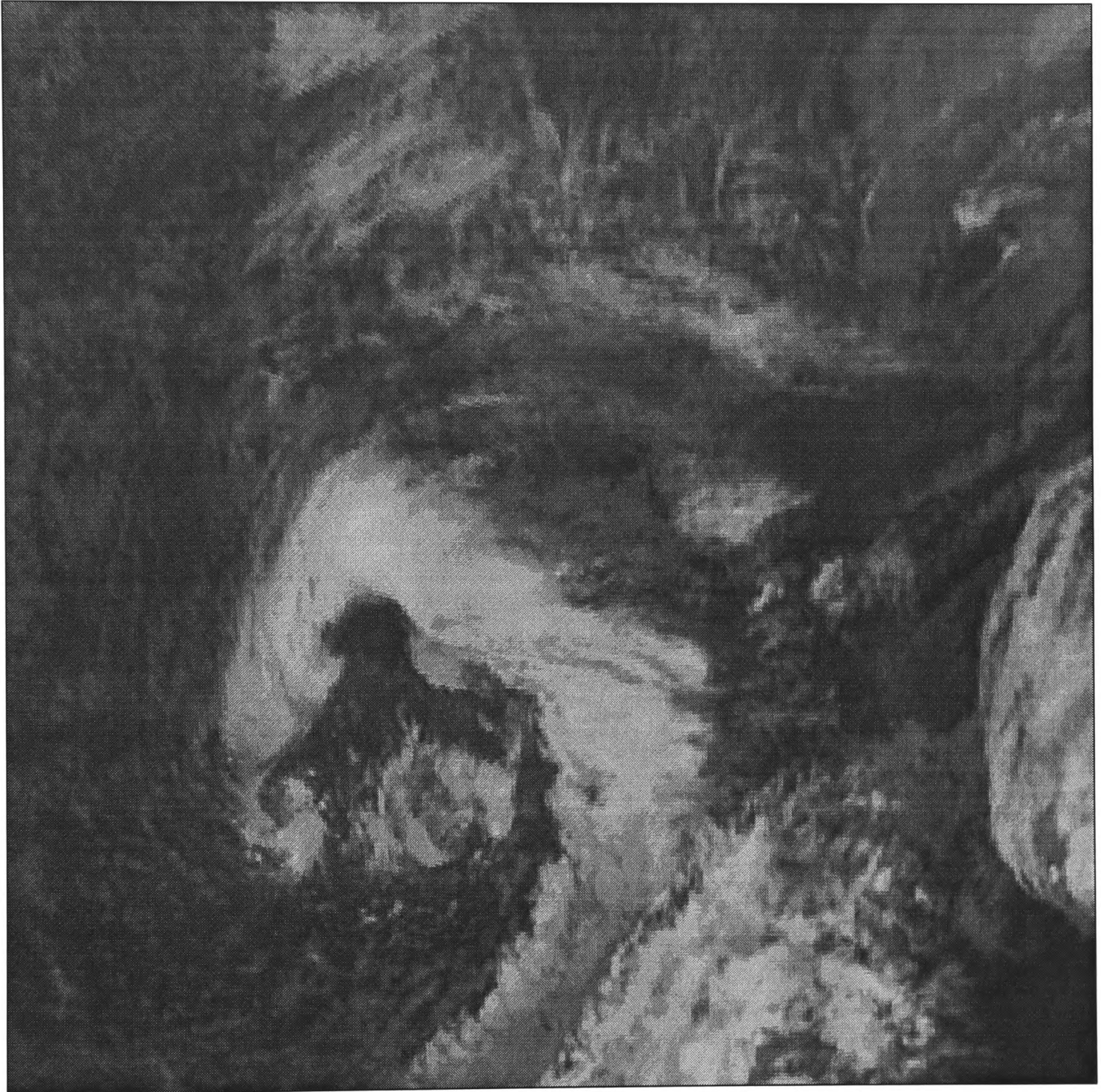


Figure D.6: The infra-red Meteosat image from 13 September 1994 at 1500 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding visible image is shown in Figure D.5.

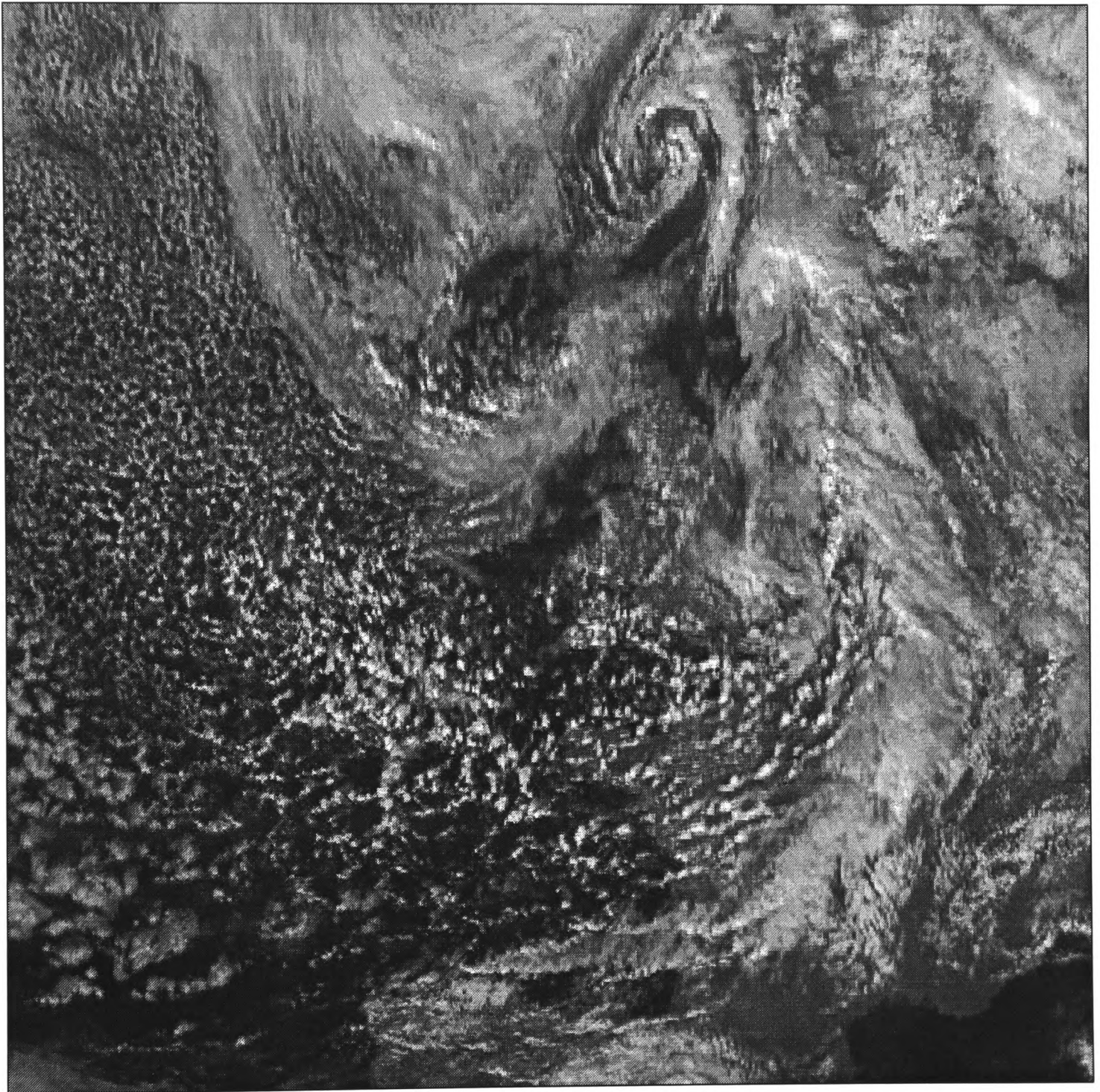


Figure D.7: The visible Meteosat image from 25 October 1994 at 1300 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding infra-red image is shown in Figure D.8.

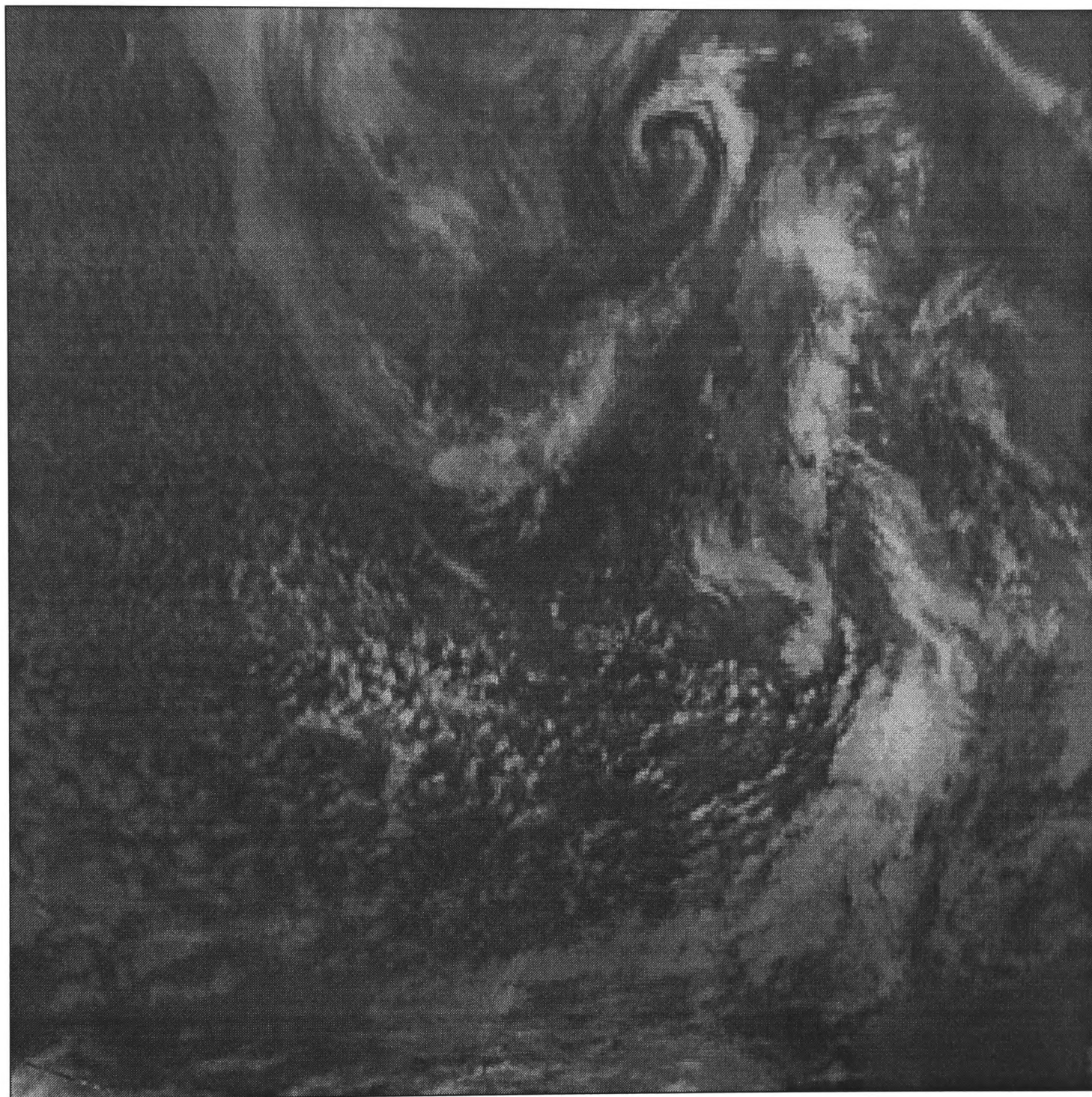


Figure D.8: The infra-red Meteosat image from 25 October 1994 at 1300 GMT, covering western Europe. The size is 640×640 pixels, or approximately 2,500 km across (each pixel represents 3.84 km). The corresponding visible image is shown in Figure D.7.

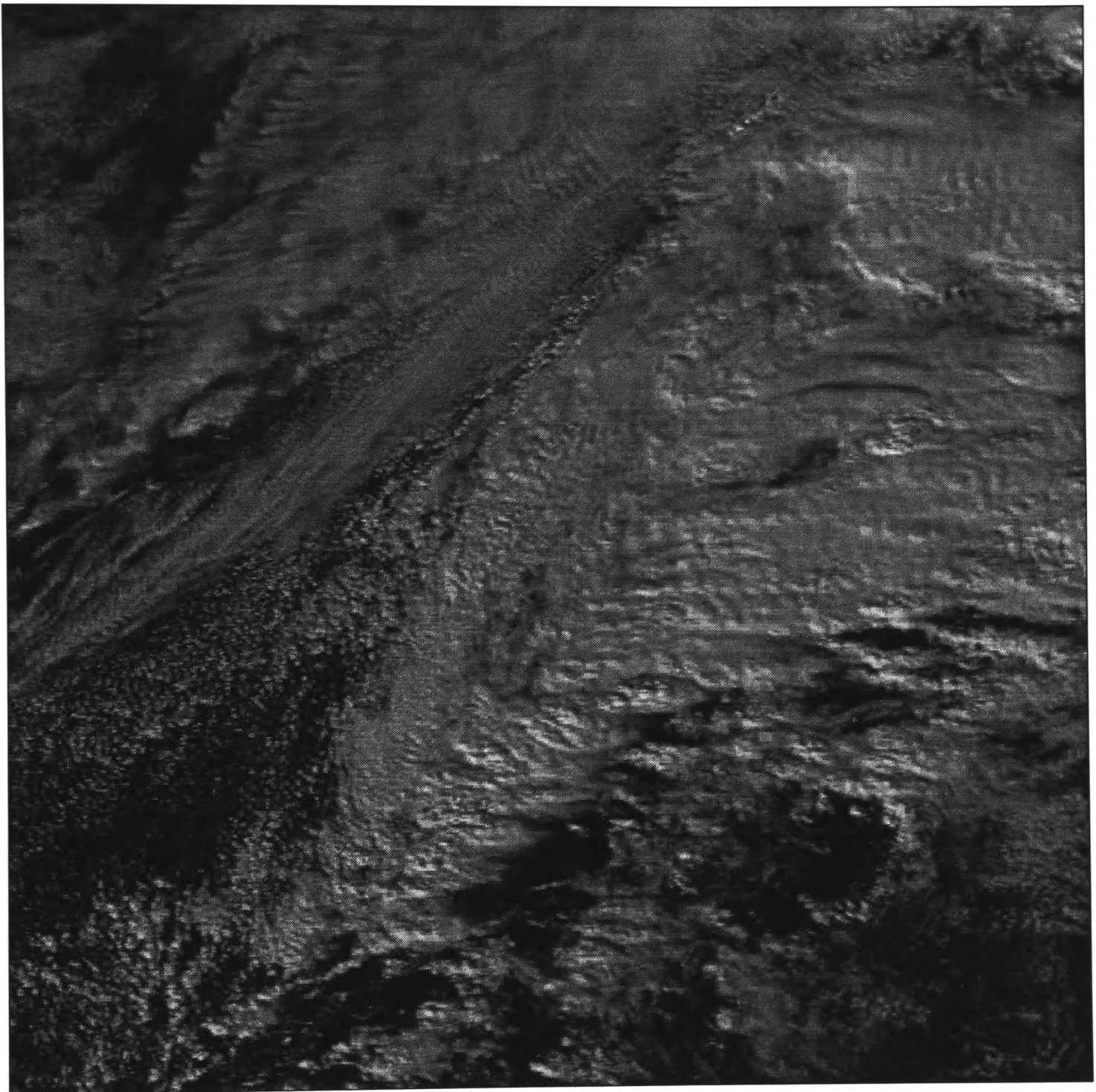


Figure D.9: The visible Bankert (AVHRR) image from 13 December 1988, off the coast of California. The size is 512×512 pixels, or approximately 560 km across (each pixel represents 1.1 km). The corresponding infra-red image is shown in Figure D.10.

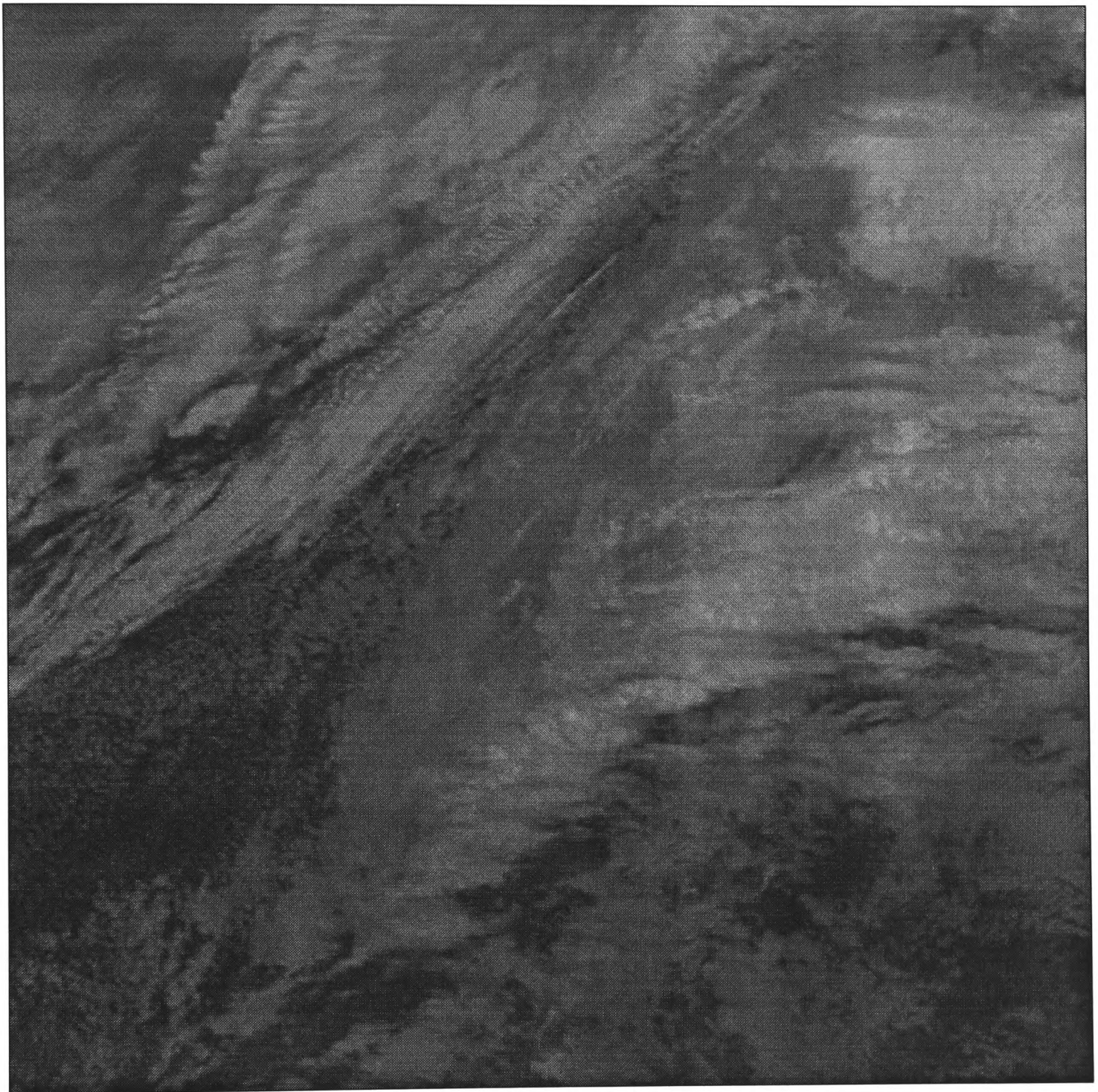


Figure D.10: The infra-red Bankert (AVHRR) image from 13 December 1988, off the coast of California. The size is 512×512 pixels, or approximately 560 km across (each pixel represents 1.1 km). The corresponding visible image is shown in Figure D.9.

Bibliography

- [1] D. W. Aha and R. L. Bankert. Cloud classification using error-correcting output codes. *AI Applications*, 11(1):13–28, 1997.
- [2] R. J. Allam, P. Dibben, G. Holpin, F. Smith, and A. P. Standley. The use of satellite imagery products for operational meteorology. In *Proceedings of the Fifth Conference on Satellite Meteorology and Oceanography*, pages 401–403, London, 1990.
- [3] J. A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research*. The MIT Press, Cambridge, MA, 1988.
- [4] M. J. Bader, G. S Forbes, J. R. Grant, R. B. E Lilley, and Waters A. J., editors. *Images in Weather Forecasting: A Practical Guide for Interpreting Satellite and Radar Imagery*. Cambridge University Press, 1995.
- [5] R. L. Bankert. Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network. *Journal of Applied Meteorology*, 33:909–918, 1994.
- [6] R. L. Bankert and D. W. Aha. Improvement to a neural network cloud classifier. *Journal of Applied Meteorology*, 35(11):2036–2039, 1996.
- [7] A. Baraldi and F. Parmiggiani. An investigation of the textural characteristics associated with gray level cooccurrence matrix statistical parameters. *IEEE Transactions on Geoscience and Remote Sensing*, 33:293–304, 1995.
- [8] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.
- [9] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: A principled alternative to the self-organizing map. Technical Report NCRG/96/030, Neural Computing Research Group, Aston University, April 1996. To appear in *Advances in Neural Information Processing Systems*.
- [10] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The Generative Topographic Mapping. Technical Report NCRG/96/015, Neural Computing Research Group, Aston University, April 1997. Accepted for publication in *Neural Computation*.
- [11] C. Bouman and B. Liu. Multiple resolution segmentation of textured images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):99–113, 1991.
- [12] C. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Transactions on Image Processing*, 3(2):162–177, 1994.
- [13] M. R. Brewer. Investigations into neural networks for satellite image interpretation. First year report for transfer of status, University of Oxford, October 1995.

- [14] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.
- [15] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.
- [16] A. Carleton. *Satellite Remote Sensing in Climatology*. CRC Press, 1991.
- [17] V. Casselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66:1–31, 1993.
- [18] V. Casselles, R. Kimmel, and C. Catalina. Minimal surfaces based object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):394–398, April 1997.
- [19] V. Casselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.
- [20] R. Chellappa and S. Chatterjee. Classification of textures using Gaussian Markov random fields. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(4):959–963, August 1985.
- [21] D. W. Chen, S. K. Sengupta, and R. M. Welch. Cloud field classification based upon high spatial resolution textural features. Part 2: Simplified vector approaches. *Journal of Geophysical Research*, 94:14749–14765, 1989.
- [22] R. T. Chin, J. Y. C. Jau, and J. A. Weinman. The application of time-series models to cloud field morphology analysis. *Journal of Climate and Applied Meteorology*, 26:363–373, March 1987.
- [23] J. Chou, R. C. Weger, J. M. Ligtenberg, K. S. Kuo, R. M. Welch, and P. Breeden. Segmentation of Polar scenes using multi-spectral texture measures and morphological filtering. *International Journal of Remote Sensing*, 15(5):1019–1036, 1994.
- [24] C. L. Crosiar. An AVHRR cloud classification database typed by experts. Technical Report NRL/MR/7531-93-7207, Naval Research Laboratory, Prediction Systems Branch, Marine Meteorology Division, Monterey, California, 1993.
- [25] G. R. Cross and A. K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):25–39, January 1983.
- [26] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.
- [27] J. Dayhoff. *Neural Network Architectures: An Introduction*. Van Nostrand Reinhold, 1990.
- [28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.
- [29] M. Desbois, G. Seze, and G. Szejwach. Automatic classification of clouds on Meteosat imagery: Application to high-level clouds. *Journal of Applied Meteorology*, 21:401–412, 1982.

- [30] P. A. Devijler and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, 1982.
- [31] C. Drewniok. Multi-spectral edge detection. Some experiments on data from Landsat-TM. *International Journal of Remote Sensing*, 15(18):3743–3765, 1994.
- [32] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.
- [33] E. Ebert. A pattern recognition technique for distinguishing surface and cloud types in the Polar regions. *Journal of Climate and Applied Meteorology*, 26:1412–1427, 1987.
- [34] M. M. Galloway. Texture analysis using gray level run lengths. *Computer Graphics and Image Processing*, 4:172–179, 1975.
- [35] L. Garand. Automated recognition of oceanic cloud patterns. Part I: Methodology and application to cloud climatology. *Journal of Climate*, 1:20–39, 1988.
- [36] L. Garand and J. A. Weinman. A structural-stochastic model for the analysis and synthesis of cloud images. *Journal of Climate and Applied Meteorology*, 25:1052–1068, July 1986.
- [37] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, Nov. 1984.
- [38] A. Gisolfi, S. Vitulano, and A. Cacace. Texture and structure. In V. Cappellini and R. Marconi, editors, *Advances in Image Processing and Pattern Recognition*, pages 179–183. North Holland, Amsterdam, 1986.
- [39] J. R. Grant and M. J. Bader. Improving NWP analyses in data-sparse areas. In *Proceedings of the Symposium on the Life Cycle of Extratropical Cyclones*, Bergen, Norway, 1994.
- [40] Z. Q. Gu, C. N. Duncan, P. M. Grant, C. F. N. Cowan, E. Renshaw, and M. A. Mugleston. Textural and spectral features as an aid to cloud classification. *International Journal of Remote Sensing*, 12:953–968, 1991.
- [41] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5), May 1979.
- [42] R. M. Haralick. Statistical image texture analysis. In T. Young and K. Fu, editors, *Handbook of Pattern Recognition and Image Processing*, pages 247–279. Academic Press, 1986.
- [43] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29(1):100–132, January 1985.
- [44] R. Harris. *Satellite Remote Sensing: An Introduction*. Routledge & Kegan Paul, 1987.
- [45] R. Harris and E. C. Barrett. Toward an objective nephanalysis. *Journal of Applied Meteorology*, 17:1258–1266, 1978.
- [46] A. K. Jain and K. Karu. Learning texture-discrimination masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):195–205, February 1996.

- [47] A. K. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.
- [48] M. Kass, A. P. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.
- [49] J. Key. Cloud cover analysis with Arctic Advanced Very High Resolution Radiometer data, Part 2: Classification with spectral and textural measures. *Journal of Geophysical Research*, 95:7661–7675, 1990.
- [50] J. Key, J. A. Maslanik, and A. J. Schweiger. Classification of merged AVHRR and SMMR Arctic data with neural networks. *Photogrammetric Engineering and Remote Sensing*, 55:1331–1338, 1989.
- [51] J. Kittler and J. Föglein. Contextual classification of multispectral pixel data. *Image and Vision Computing*, 2(1):13–29, February 1984.
- [52] J. Kittler and D. Pairman. Contextual pattern recognition applied to cloud detection and identification. *IEEE Transactions on Geoscience and Remote Sensing*, 23:855–863, 1985.
- [53] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.
- [54] S. Krishnamachari and R. Chellappa. Multiresolution Gauss-Markov random-field models for texture segmentation. *IEEE Transactions on Image Processing*, 6(2):251–267, February 1997.
- [55] K. S. Kuo, R. M. Welch, and S. K. Sengupta. Structural and textural characteristics of cirrus clouds observed using high spatial resolution Landsat imagery. *Journal of Applied Meteorology*, 27:1242–1260, 1988.
- [56] S. Lakshmanan and H. Derin. Gaussian Markov random fields at multiple resolutions. In R. Chellappa and A. Jain, editors, *Markov Random Fields: Theory and Applications*, pages 131–157. Academic Press, Inc., 1993.
- [57] J. Lee, R. C. Weger, S. K. Sengupta, and R. M. Welch. A neural network approach to cloud classification. *IEEE Transactions on Geoscience and Remote Sensing*, 28:846–855, 1990.
- [58] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.
- [59] E. Liljas. Multispectral methods for cloud classification. In *Proceedings of the EUMETSAT Workshop*, pages 475–493, Reading, England, July 1987. Preprint.
- [60] R. Malladi and J. A. Sethian. Image processing via level set curvature flow. In *Proceedings of the National Academy of Science, USA*, volume 92, pages 7046–7050, July 1995.
- [61] R. Malladi and J. A. Sethian. An $O(N \log N)$ algorithm for shape modeling. In *Proceedings of the National Academy of Science, USA*, volume 93, pages 9389–9392, September 1996.

- [62] R. Malladi and J. A. Sethian. A real-time algorithm for medical shape recovery. In *Proceedings of the International Conference on Computer Vision*, Mumbai, India, January 1998.
- [63] R. Malladi, J. A. Sethian, and B. C. Vemuri. Evolutionary fronts for topology-independent shape modeling and recovery. In *Proceedings of Third European Conference on Computer Vision*, pages 3–13, Stockholm, Sweden, May 1994.
- [64] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, February 1995.
- [65] W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- [66] Meteorological Office College, Shinfield Park, Reading. *Remote Sensing Course Notes*.
- [67] J. L. Meurle and D. C. Allen. Experimental evaluation of techniques for automatic segmentation of objects in a complex scene. In G. C. Chang, R. S. Ladley, D. K. Pollock, and Rosenfeld A., editors, *Pictorial Pattern Recognition*, pages 3–13. Thompson, Washington D.C., 1968.
- [68] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.
- [69] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulation. *Journal of Computational Physics*, 79:12–49, 1988.
- [70] D. Pairman and J. Kittler. Clustering algorithms for use with images of clouds. *International Journal of Remote Sensing*, 7(7):855–866, 1986.
- [71] G. S. Pankiewicz. The applications of pattern recognition techniques to cloud detection. Technical Report 129, UK Meteorological Office, November 1994.
- [72] G. S. Pankiewicz. A neural network cloud classifier trained with AVHRR data for use on Meteosat imagery. In *Proceedings of the 1995 Meteorological Satellite Data Users' Conference*, pages 393–400, Winchester, England, 1995.
- [73] G. S. Pankiewicz. Pattern recognition techniques for the identification of cloud and cloud systems. *Meteorological Applications*, 2(3):257–271, September 1995.
- [74] G. S. Pankiewicz. Neural network classification of convective airmasses for a flood forecasting system. Scientific Paper 45, UK Meteorological Office, December 1996.
- [75] G. S. Pankiewicz. New Meteosat pattern recognition products for use in weather forecasting. Scientific Paper 46, UK Meteorological Office, December 1996.
- [76] G. S. Pankiewicz. Toward synoptic scale cloud classification: Combining spectral and textural features. In *Proceedings of the 1996 Meteorological Satellite Data Users' Conference*, pages 111–118, Vienna, Austria, 1996.
- [77] G. S. Pankiewicz. Neural network classification of convective airmasses for a flood forecasting system. *International Journal of Remote Sensing*, 18(4):887–898, 1997.

- [78] G. S. Pankiewicz. Satellite imagery pattern recognition products for numerical weather prediction. Technical Report 206, UK Meteorological Office, January 1997.
- [79] J. Parikh. A comparative study of cloud classification techniques. *Remote Sensing of the Environment*, 6:67–81, 1977.
- [80] J. Park and I. W. Sandberg. Universal approximation using radial basis functions. *Neural Computation*, 3(2):246–257, 1991.
- [81] T. Pavlidis. *Algorithms for Graphics and Image Processing*. Computer Science Press, New York, 1982.
- [82] J. E. Peak and P. M. Tag. Toward automated interpretation of satellite imagery for navy shipboard applications. *Bulletin of the American Meteorological Society*, 73(7):995–1008, July 1992.
- [83] J. E. Peak and P. M. Tag. Segmentation of satellite imagery using hierarchical thresholding and neural networks. *Journal of Applied Meteorology*, 33:605–616, 1994.
- [84] C. B. Pease. *Satellite Imaging Instruments: principles, technologies and operational systems*. Ellis Horwood Limited, 1991.
- [85] M. A. Penaloza and R. M. Welch. Feature-selection for classification of Polar regions using a fuzzy expert-system. *Remote Sensing of Environment*, 58(1):81–100, 1996.
- [86] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*, pages 143–167. Clarendon Press, Oxford, 1987.
- [87] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian *a posteriori* probabilities. *Neural Computation*, 3:461–483, 1991.
- [88] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- [89] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):417–431, 1983.
- [90] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.
- [91] W. B. Rossow. Measuring cloud properties from space: A review. *Journal of Climate*, 2:201–213, 1989.
- [92] W. B. Rossow and L. C. Garder. Validation of ISCCP cloud detections. *Journal of Climate*, 6:2370–2393, December 1993.
- [93] W. B. Rossow, F. Moshier, E. Kinsella, A. Arking, M. Desbois, E. Harrison, P. Minnis, E. Ruprecht, G. Seze, C. Simmer, and E. Smith. ISCCP cloud algorithm intercomparison. *Journal of Climate and Applied Meteorology*, 24(9):877–903, 1985.
- [94] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

- [95] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of cognition*. MIT press, 1986.
- [96] F. F. Sabins. *Remote Sensing: Principles and Interpretation*. W. H. Freeman and Company, New York, 3rd edition, 1997.
- [97] J. W. Sammon, Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.
- [98] U. Schendel. *Sparse Matrices: numerical aspects with applications for scientists and engineers*. Ellis Horwood Limited, 1989.
- [99] R. A. Schiffer and W. B. Rossow. The International Satellite Cloud Climatology Project (ISCCP): the first project of the World Climate Research Programme. *Bulletin of the American Meteorological Society*, 64:779–784, 1983.
- [100] A. M. Seddon and G. E. Hunt. Segmentation of clouds using cluster analysis. *International Journal of Remote Sensing*, 6(5):717–731, 1985.
- [101] S. K. Sengupta, R. M. Welch, M. S. Navar, T. A. Berendes, and D. W. Chen. Cumulus cloud field morphology and spatial patterns derived from high spatial resolution Landsat imagery. *Journal of Applied Meteorology*, 29:1245–1267, 1990.
- [102] J. A. Sethian. Curvature and evolution of fronts. *Communications in Mathematical Physics*, 101(4):487–499, 1985.
- [103] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Science*. Cambridge University Press, 1996.
- [104] J. A. Sethian and J. Strain. Crystal growth and dendritic solidification. *Journal of Computational Physics*, 98:231–253, 1992.
- [105] G. Seze and M. Desbois. Cloud cover analysis from satellite imagery using spatial and temporal characteristics of the data. *Journal of Climate and Applied Meteorology*, 26:287–303, 1987.
- [106] G. L. Smith, B. R. Barkstrom, and E. F. Harrison. The Earth Radiation Budget Experiment: early validation and results. *Advances in Space Research*, 7(3):167–177, 1987.
- [107] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Chapman and Hall, 1993.
- [108] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.
- [109] V. R. Tovinkere, M. Penaloza, A. Logar, J. Lee, R. C. Weger, T. A. Berendes, and R. M. Welch. An intercomparison of artificial intelligence approaches for Polar scene identification. *Journal of Geophysical Research*, 98(D3):5001–5016, 1993.
- [110] M. Unser. Sum and difference histograms for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:118–125, 1986.
- [111] R. M. Welch, M. S. Navar, and S. K. Sengupta. The effect of spatial resolution upon texture-based cloud field classifications. *Journal of Geophysical Research*, 94(D12):14767–14781, 1989.

- [112] R. M. Welch, S. K. Sengupta, and D. W. Chen. Cloud field classification based upon high spatial resolution textural features. Part 1: Gray level co-occurrence matrix approach. *Journal of Geophysical Research*, 93(D10):12663–12681, 1988.
- [113] R. M. Welch, S. K. Sengupta, A. K. Goroch, P. Rabindra, N. Rangaraj, and M. S. Navar. Polar cloud and surface classification using AVHRR imagery: An intercomparison of methods. *Journal of Applied Meteorology*, 31:405–420, 1992.
- [114] P. Werbos. *Beyond Regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard, 1974.
- [115] 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*, 6:2269–2285, 1976.
- [116] B. Widrow and M. E. Hoff, Jr. Adaptive switching circuits. *IRE WESCON Convention Record*, 4:96–104, 1960. Reprinted in Anderson and Rosenfeld [3].
- [117] S. R. Yhann and J. J. Simpson. Application of neural networks to AVHRR cloud segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 33(3):590–604, 1995.
- [118] S. R. Yhann and T. Y. Young. Boundary localization in texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6):849–856, 1995.

