Spatial modelling of cloud

FRÉDÉRIC LÊ

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

September 2003

ASTON UNIVERSITY

Spatial modelling of cloud

Frédéric Lê

MSc by Research in Pattern Analysis and Neural Networks, 2003

Thesis Summary

Clouds have always been a source of scientific research as they play an important role in regulating the Earth's weather. Much of the information we get about clouds is obtained by satellite images, which need to be processed in order to obtain interesting information. Indeed, due to the existence of different cloud types, there is a need of building a reliable cloud classification method. The aim of this thesis will be to create a spatial retrieval of cloud types, combining visible and infrared satellite images. This includes a preprocessing of the data to separate cloudy pixels from the underlying surface, a model of cloudiness using a radial basis functions neural network and a probabilistic method to classify clouds.

Keywords: radial basis functions neural network, cloud screening, cloud modelling, cloud classification

Acknowledgements

First, I would like to thank my supervisors, Dan Cornford and Ian Nabney who have guided my research. I am grateful for all their help, their advice and their availability, especially Dan Cornford.

Moreover, I would like to thank all the Neural Computing Research Group and particularly the lecturers for having taught me the useful mathematical background to achieve this thesis. I have really appreciated their availability and their friendliness which were precious to me.

I also want to thank my MSc friends for all their help during the courses and during the research period. Their friendship and their support have been essential for me during this year.

Finally, I would also like to thank my family for their support and their financial help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The importance of clouds

Clouds consist of condensed water vapour suspended in the air, which result from the aggregation of minute particles of water or ice. They are visible signs of atmospheric processes at work. Indeed, they are essential to the Earth-atmosphere system as they fulfil different important functions.

1.1.1 Solar radiation regulation

The Sun and the Earth both emit and absorb energy, which scientists refer to as radiation. Our planet absorbs incoming sunlight and also emits its own energy. For this reason, the Earth's outgoing energy has two sources: thermal radiation that the Earth's surface and atmosphere emit and solar radiation that the land, oceans and clouds reflect back to space. The balance between incoming and outgoing energy determines the planet's temperature and, ultimately, the climate [16].

Clouds help to regulate the Earth's temperature by playing two complementary roles. By reflecting the sunlight into space, they cool the atmosphere and by absorbing sunlight and heat, they keep warming the planet. This regulation is essential to keep a 'pleasant' temperature on our planet.

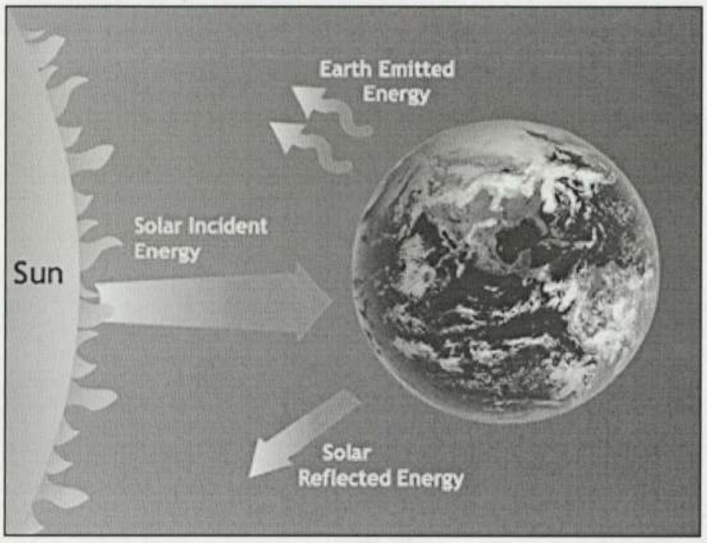

Figure 1.1: The Earth's incoming and outgoing energy.

1.1.2 The hydrologic cycle

Clouds are the essential link in the hydrologic cycle. They are responsible for the rain and contribute to life on Earth. Water is necessary for vegetation growth and clouds bring it everywhere. Moreover, as the vast majority of water evaporation occurs from the oceans, clouds are very useful as they are involved in the conversion of salt water to usable drinking water.

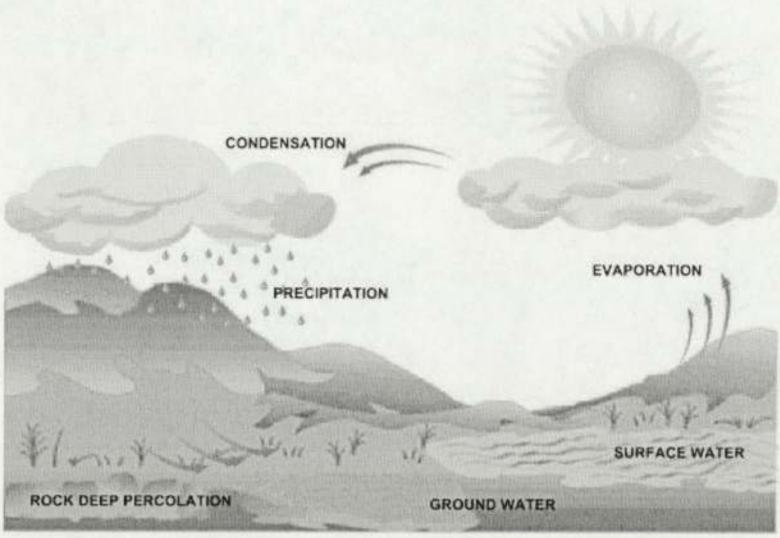

Figure 1.2: The hydrologic cycle.

1.2 International cloud classification

A classification of cloud forms was first made in 1801 by French naturalist Jean Lamarck. In 1803, Luke Howard, an English scientist, devised a classification which was adopted by the International Meteorological Commission in 1929, designating three primary cloud types, cirrus, cumulus, and stratus, and their compound forms which are still used nowadays. We distinguish ten different basic types:

Cloud type	Description
Cirrus	Fibrous like or silky sheen
Cirrocumulus	Thin white patch
Cirrostratus	Transparent clouds that make halo of sun or moon
Altocumulus	Bumpy rounded masses, like wool
Altostratus	Transparent blue/gray clouds with no halo
Nimbo-stratus	Storm cloud, dark, covers sun
Strato-cumulus	Gray or whitish layer with dark parts
Stratus	Low clouds with drizzle or snow, no halo
Cumulus	Rising mounds of cauliflower white
Cumulo-nimbus	Huge towers, storm clouds, hail, lightning

Table 1.1: Description of the ten basic cloud types.

These clouds have different shapes, motion and height in the atmosphere as shown on the figure below [7].

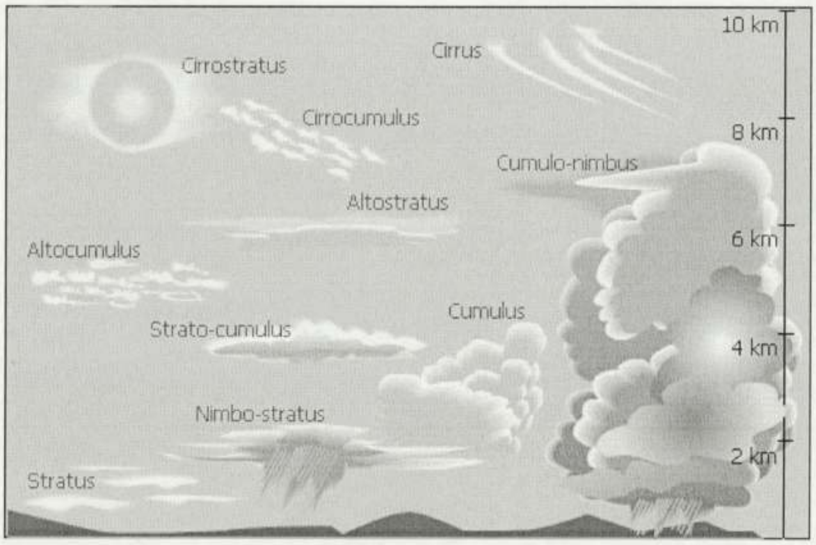

Figure 1.3: Shape and height of the ten basic cloud types.

1.3 Different approaches to cloud classification

Numerous approaches have already been taken to classify clouds, using different techniques of image processing or cloud modelling to retrieve the different classes of clouds.

1.3.1 Conventional statistical classifiers

With the development of weather satellites, several studies have investigated cloud detection and cloud classification since 1970 [1]. Considerable research has looked at different features for discriminating between clouds and other surfaces and different methods for detecting and classifying clouds have been applied. Generally, two categories of features are examined: spectral features and textural features. The first category uses the information in the cloud radiance from different spectral bands, while the second uses the spatial distribution characteristics of intensity measurements in a single spectral channel.

Multispectral threshold classifiers

Reynolds and Haar (1977) introduced a bispectral method using visible and infrared channels to focus on clouds. Visible images were used to localise clouds whereas infrared images were used to determine cloud top temperature and thus cloud height [12]. This method is an efficient way of classifying clouds into different cloud level classes and will be detailed in this thesis in Chapter 5.

Saunders and Kriebel (1988) developed a method for detecting clear sky and cloudy radiances [13]. The scheme consists of five tests applied to each individual pixel to determine whether that pixel is cloud-free, partly cloudy or cloud-filled. The tests vary depending on whether it is night or day and on the underlying surface type, sea, land or coast (mixed). They use all the channels of the 'Advanced Very High Resolution Radiometer' (AVHRR), where channels 1 and 2 measure the bi-directional reflectances and channels 3, 4 and 5 measure the infrared brightness temperatures. The first test considers the infrared brightness temperature of the pixel to discuss about cloud contamination. The pixel's brightness temperature is compared to a threshold and is then labelled as cloud-free or cloud-contamined. The second test check the local uniformity or the spatial coherence of a 3×3 pixel array using the brightness temperature. The third test verifies the dynamic reflectance, setting a threshold to detect cloudiness as clouds have a greater reflectance than the underlying surface. The fourth test makes use of the ratio of near-infrared bi-directional reflectances (channel 2) to visible bi-directional reflectances (channel 1) and the bi-directional reflectance R_n

is defined by:

$$R_n = \frac{G_n C + Y_n}{cos(\theta_0)} ,$$

where R_n is in units of percentage, G_n is the gain, Y_n is the intercept, C is the raw count value received from the satellite for the channel n and θ_0 is the appropriate solar zenith angle. The ratio used in the test is defined as:

$$Q = \frac{R_2}{R_1} .$$

The ratio Q is close to unity over clouds, as the reflectance of clouds only decreases slightly at near-infrared wavelengths and anisotropy effects are similar in both channels and hence cancel. The last test examine the difference between the 11 μm (channel 4) and 12μm (channel 5) brightness temperatures. This brightness temperature difference can be used to detect most types of clouds, the only exception being uniform low clouds. A threshold is set to separate clear sky and cloudy pixels.

A multispectral threshold algorithm has also been developed by Karlsson and Liljas to obtain an operational automated cloud classification model [8]. The final classification model, named SCANDIA (the SMHI Cloud ANalysis model using DIgital AVHRR data) was implemented and is a supervised thresholding model. AVHRR scenes are classified by using seven image features. One feature is a land/sea mask based on a geographical map, whereas the other six are the following pure spectral features, based on all five AVHRR channels:

- CH1: Bi-directional visible reflectance,
- CH1-CH2 VIS-NIR: Reflectance difference,
- CH3-CH4: Brightness temperature difference,
- CH4: Brightness temperature,
- CH5-CH4: Brightness temperature difference,
- TEX4: Local brightness temperature variance in AVHRR channel 4.

The pixels in each AVHRR scene are separated into a maximum of 57 classes (cloud types and surfaces) depending on the sun elevation. However, the final image never comprises more than 23 classes.

Classifiers based on thresholds are popular as they have the advantage of being simple in implementation and maintenance. But, the main drawback with threshold classifiers is that they are very sensitive to changes in the operational environment, i.e.

variations in the illumination conditions, climate and season. In sunglint and snow or ice regions, the reflection of the solar radiation can exceed the reflectivity of clouds which makes thresholds difficult to define.

Existing approaches to cloud detection and cloud classification

Work has also been done concerning cloud screening and cloud classification using different methods rather than spectral thresholds. Lakshmanan et al. have investigated a hierarchical texture segmentation method using K-means clustering [9]. This work was motivated by the difficulties that traditional segmentation algorithms have with satellite weather images because of the textural nature of clouds. The model proposed is a hierarchical technique combined with a texture segmentation algorithm.

A hierarchical clustering method is an agglomerative technique where the clustering algorithm at each stage merges two or more trivial clusters, thus nesting the earlier partition into a smaller number of clusters. Images are segmented using an iterative texture segmentation method that yields a hierarchical representation of the regions at different scales. This method makes use of K-means clustering to requantise the image in K levels.

A vector of statistical measurements taken in the neighbourhood of a pixel is associated with that pixel. Each pixel is then assigned to one of the K equal intervals defined in the measurement space (the gray level of the images), according to its initial gray level value. In each iteration, the best label for each pixel in the image is chosen based on a cost factor that incorporates two measures. The first one is the Euclidean distance, $d_m(k)$, between the texture vector at that pixel and the cluster mean of the candidate k, given by:

$$d_m(k) = ||\mu_k^n - T_{xy}|| \ ,$$

where μ_k^n is the cluster mean of the k^{th} cluster at the n^{th} iteration and T_{xy} is the texture vector at the pixel (x, y). The second measure is a contiguity measure, $d_c(k)$, which measures the number of neighbours whose labels differed from the candidate label k. We can formally express the distance $d_c(k)$ as:

$$d_c(k) = \sum_{ij \in N_{xy}} (1 - \delta(S_{ij}^n - k)) \ ,$$

where S_{ij}^n is the label of the pixel (i, j) at the n^{th} iteration, N_{xy} is the set of 8-neighbours of the pixel (x, y) and δ is the usual Kronecker symbol. Then the choice of the label for the pixel (x, y) in the $(n+1)^{th}$ iteration, S_{ij}^{n+1}, is given by the label $k \in S_{N_{xy}}^n$ for which the energy, $E(k)$ given by

$$E(k) = \lambda d_m(k) + (1 - \lambda) d_c(k)$$

is minimum. The value of λ used for all the images is $\lambda = 0.6$, which is the optimal value found experimentally. At the end of each iteration, the cluster attributes (μ_k) are updated based on all the pixels that were labelled as belonging to the cluster at that time. This process is repeated until the regions are such that all cluster means have reliable statistics. This method is efficient to identify small features of about 10 km^2 in satellite images but more study is required to ascertain the significance of these small features.

Other investigations such as the one of Peura et al. developed a combination of satellite images and land-based observations [11]. The method suggested is based on a source image captured by a whole-sky image and transformed to a digital array. Then, several feature images are extracted from the source image. Each feature image emphasises some distinguishing property of clouds. The features considered in the study are:

- *Edge sharpness.* Edges of a cloud can be detected by the gradient operation $\nabla f(x, y)$, which is obtained from finite difference approximations in digital images. Pronounced edges are characteristic of cumulus clouds whereas stratus, nimbostratus, altostratus and cirrostratus have indefinite contours.

- *Speck size.* This measure is quite problematic as there exists no proper definition for an edge of a cloud. A simple approach is to apply spatial band-pass masks with different radii. Specks can also be analysed by slicing them to connected gray-level segments areas of which can be calculated recursively.

- *Fibrousness.* This property, suggesting detection of lines, is typical to cirrus and sometimes cirrostratus, nimbostratus and cumulonimbus. Line segments can be detected by calculating approximations of second derivatives and dividing the result by the differences of the gray levels neighbouring the current locus. This operator is rather insensitive to edges of clouds. In addition, fibrousness can be measured by summing adjacent gray-level transitions and also by vector products of the two unequally spanned gradients.

- *Association of edge information.* The whole-sky images should be seen as a composition of objects, i.e. specks of clouds, rather than a pure texture. Local information of edges of clouds is propagated inside the corresponding specks.

As clouds consist of areas with varying appearance, some of the ten cloud genera are partitioned to subclasses (edge, speck, bulk, gap). The preclassified samples needed in the classification consist of single feature vectors. The collection of samples, called a *codebook*, is created by investigating a comprehensive set of cloud images and extracting feature vectors at positions containing representative details of cloud genera. A straightforward method of applying a codebook is to find the most resembling vector and return its label as the result. This is called the Nearest-Neighbour algorithm. A generalisation of this is the K-Nearest-Neighbours algorithm: the codebook is searched for K samples having smallest distance to the given example. The sample is assigned the class having the largest number of instances among these K vectors.

The results obtained with this method are comparable to some extent to those obtained with the other classifiers. However, the number of preclassified samples required to form a codebook can be large and the distance calculation performed to each codebook vector is a time-consuming operation. Moreover, some improvements have to be done to distinguish cloud layers.

Cloud-screening problems have also been tackled by Cadez and Smyth [5]. They chose to apply inhomogeneous statistical spatial models in the form of Markov random field (MRF) models to the cloud detection and developed an efficient algorithm for the estimation of model parameters. The standard MRF model assumes that the coupling between pixel labels is globally constant throughout the image. An inhomogeneous MRF model allows this coupling parameter to vary spatially. This model takes advantage of spatial information whereas existing cloud-screening algorithms make decisions on a pixel-by-pixel basis.

If we define S to be a set of lattice points, s a lattice point belonging to S ($s \in S$), X_s the value of X at s and ∂s the neighbouring points of s, a random object X on the lattice S with neighbourhood system ∂s is said to be a Markov random field if for all $s \in S$

$$p(x_s | x_r \text{ for } r \neq s) = p(x_s | x_{\partial r}) \ .$$

The main object of interest of this study is a rectangular $n \times m$ image S consisting of sites s_{ij} ordered in a matrix manner. The neighbourhood of the site s_{ij} is any subset $\partial_{ij} \subset S$ such that $s_{ij} \notin \partial_{ij}$. The neighbourhood system N is the set of all the neighbourhoods: $N = \{\partial_{ij} | 1 \leq i \leq n, 1 \leq j \leq m\}$. At each site, we define an intensity random variable X_{ij} (typically taking 256 gray-levels) and a hidden label random variable Y_{ij} (discrete-valued, k labels). The specific values the random variables take are denoted x_{ij} and y_{ij} respectively. This gives two sets of variables defined on the image S: $X = \{X_{11}, ..., X_{nm}\}$, and $Y = \{Y_{11}, ..., Y_{nm}\}$. From the Hammersley-Clifford theorem, a MRF for $P(Y)$ is of the form

$$P(Y) = \frac{1}{z} e^{\beta V(Y)} = \frac{1}{z} e^{\beta \sum_{i,j} V_{ij}(\partial_{ij} \cup \{y_{ij}\})} \ ,$$

where z is known as the partition function or normalising constant (yielding $\sum_Y P(Y) = 1$), β is the temperature parameter (frequently called the smoothness parameter in image segmentation) and V is the so-called potential function; its extrema are tightly connected to the optimal segmentation within the MRF framework.

To simplify the notation, $n_{ij}(y_{ij})$ will stand for the number of pixels labelled y_{ij} in the neighbourhood of the site s_{ij}. Thus, the potential function reduces to: $\beta V_{ij}(\partial_{ij} \cup \{y_{ij}\}) = \beta n_{ij}(y_{ij})$. The study also assumes that observed intensities X_{ij} only depend on the local Y_{ij} labels and are conditionally Gaussian given the local model, setting $\mu_{ij} = \mu_{y_{ij}}$, and $\sigma_{ij} = \sigma_{y_{ij}}$. Then Bayes' theorem yields a complete model coupling intensities and labels:

$$P(Y|X) \propto P(X|Y)P(Y) = \frac{1}{z} e^{\sum_{i,j} \beta_{ij} n_{ij}(y_{ij})} \prod_{ij} \frac{1}{\sqrt{2\pi\sigma_{y_{ij}}^2}} e^{-1/2\sigma_{y_{ij}}^2 (x_{ij} - \mu_{y_{ij}})^2} .$$

The goal of segmentation is then to maximise $P(Y|X)$ with respect to labels Y. The results obtained show that spatial models perform better than non-spatial models for cloud-screening problems.

1.3.2 Neural network classifiers

Since the 1980's, it has been popular to use neural network based classifiers in remote sensing. Visa et al. have presented the different current phases of the classification of satellite images and have also discussed about the advantages of using neural network classifiers [15]. The scheme is the same as the one used for the previous classifiers described. The classification of a satellite image is done in two phases, known as cloud screening and cloud classification. The clouds are separated from the surface using a selection of features compared to a cloud screening codebook and are then classified following a classification codebook.

With the results obtained, they discussed the advantages and the drawbacks of neural network classifiers compared to traditional methods of classification. It appears that the neural network classifiers are more effective than traditional classifiers. They have the ability to generalise and make simple models of processes or phenomena. They are efficient in generating and updating the codebook using training samples but they are weak in extrapolating far away outside the set of the training samples. A suitable preprocessing of the samples and a feature selection are still necessary to improve the performance of a neural network classifier. It can also be stated that an analytic, physical model is still superior to a neural network, but a neural network offers a rapid way to get good results and to study the process. The computational complexity is also an important point, depending on the implementation of the neural network and the size of the dataset. Neural network classifiers are fully automatic and can be adapted to changing situations with new examples. The comparisons with other published results show that the performance of the classifier is relatively good. Hence its flexibility rather than the results obtained tends to make it popular.

1.4 Space-time models for clouds

The initial aim of the thesis was to build a dynamic model to classify clouds. Unfortunately, due to a lack of time, only the spatial part has been investigated. The research is based on a neural network architecture. The first objective of the work is to preprocess the images by removing the surface pixels. Then, a modelling of the clouds by a radial basis functions neural network is envisaged. This choice is motivated by the fact that we wanted to study the dynamics of the clouds and it is easier to focus on the motion of the basis functions rather than the motion of each pixel during time. The use of the MLP in cloud classification is popular and provides very good results but it does not fit the model we wanted to build. This spatial and temporal approach to cloud classification using an RBF neural network for modelling and dynamic retrieval is the main originality of this thesis.

In this topic, Bailey et al. have proposed a model for studying cloud cover, using its spatial and temporal distribution [2]. Their model introduces a parameterisation of cloud cover, representing the cloud's contour by a radius vector function. A small number of Fourier coefficients of the radius vector function are used to describe the contour function and corresponding area of a cloud. The radius vector function ($0 \leq \theta \leq 2\pi$) is defined by

$$r(\theta) = a_0 + \sum_{n=1}^{p} a_n \cos n\theta + b_n \sin n\theta \ .$$

The centre of gravity of each cloud is used as the centre of the contour of the cloud. For the analysis, the Fourier series is truncated after six coefficients as experimentally, this number seems to reasonably represent the cloud and total cloud cover.

To incorporate the spatial distribution of clouds, the spatial pattern of the location of the centres of the clouds is investigated. It is possible to model the centres as a marked Markov process, where the locations of the centres of the clouds are marked by the first Fourier coefficient which is the mean radius of the cloud.

A first-order time-lagged spatial nearest-neighbour model is used to model cloud cover over time. Consider the gridded region with r rows and c columns ($L = rc$). Let $(X_{1t}, ..., X_{Lt})$' be an L-vector of values at time t. The first-order lagged nearest-neighbour model for each grid site l is

$$X_{l(t+1)} = f(X_{n(l)t}) + e_t \ , \tag{1.1}$$

where e_t is a normal distribution $N(0, \sigma^2)$ independently and identically distributed. The $n(l)$ is a nearest-neighbourhood of each site, which has a row-column index (i, j):

$$n(l) = (i,j), ((i+1),j), ((i-1),j), (i,(j+1)), (i,(j-1)) \ .$$

The map f is estimated by a feed-forward neural network with a single layer of hidden units. The form of the model is

$$f(X) = \beta_0 + \sum_{i=1}^{k} \beta_i \phi(X^T \gamma_i + \mu_i) \ ,$$

where $\phi(u) = \frac{e^u}{1+e^u}$. The parameters β, γ and μ are estimated by nonlinear least squares. The complexity of the model, i.e. the number of hidden units k, is chosen based on generalised cross validation which is a standard approach for selecting smoothing parameters in nonparametric regression. The computation of Equation 1.1 gives the temporal behaviour of the model. The results obtained give a good approximation of cloud cover.

1.5 Thesis overview

The thesis will be built around a classical approach of cloud classification using neural networks, following Figure 1.4. The main phases of the work will focus on image processing to separate clouds from the surface, cloud modelling to convert data to a neural network output and finally cloud classification to determine the types of clouds. The image processing and the cloud classification will use traditional methods such as a radiance threshold to distinguish clouds from the surface and a combination of visible and infrared images to retrieve cloud temperature and hence cloud height. The cloud modelling will envisage an original approach by using an RBF neural network to model cloudiness. This model should fit the framework developed to take in consideration the spatial and temporal behaviour of the clouds for cloud classes retrieval.

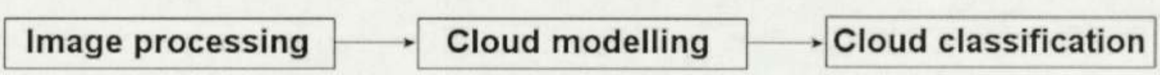

Figure 1.4: Steps for the thesis work.

Chapter 2 will present the data used for the thesis and explain their choice. Chapter 3 will deal with the removal of surface pixels using a probabilistic method to retain only cloudy pixels. Then, Chapter 4 will go through the modelling of the clouds by a radial basis function network. It will introduce the definition of a new basis function to fit clouds better than the usual basis functions defined. Finally, Chapter 5 will propose a method for cloud classification combining visible and infrared images. Further possible developments will be treated in Chapter 6.

Chapter 2

Description of the dataset

For building our model, the use of a large number of satellite images is necessary. Indeed, these images are essential as their provide exhaustive information about clouds. All these images are taken from the METEOSAT satellite. METEOSAT is a spin stabilised satellite located in a geostationary orbit at 36 000 kilometres above the Earth at the crossing of the equator and the 0-meridian. The images were provided by the Dundee Satellite Receiving Station of Dundee University (United Kingdom) and were available on the web site http://www.sat.dundee.ac.uk.

2.1 Visible images

Visible images measure the radiance of the reflected sunlight on the Earth. As a consequence, the most exploitable data are pictures taken during the daytime. These images provide essential pieces of information about clouds, such as the shape and using their brightness, we are able to distinguish them from the surface which is generally darker.

Figure 2.1: Visible satellite image of Europe taken on July, 14th 2003 at 12:00 am.

To obtain the most exploitable data, it is relevant to focus on visible images taken at midday. Indeed, by that time, the illumination of the Sun is at its maximum for the regions observed and as the Sun is at its zenith, there are no problems of biased floodlighting which means that there is less need to correct radiance to produce reflectances.

2.2 Infrared images

Infrared images measure the radiation emitted by the Earth atmosphere and therefore the temperature. The warmer a particle is, the more radiation it emits, following Planck's law. To match the appearance of the visible images, the infrared image scale has been inverted so that clouds appear as bright pixels and the surface as dark pixels. On infrared images, the brighter a pixel, the colder it is. This information will be useful to determine the height of the clouds as height and temperature are linked. Actually, the higher we are in the atmosphere, the colder the temperature.

Figure 2.2: Infrared satellite image of Europe taken on July, 14th 2003 at 12:00 am.

As we will need to combine visible and infrared images for the cloud classification, we want to observe the visible image in the infrared domain. For that reason, all infrared images are coupled with the visible images.

2.3 Working area

The visible and infrared images do not have the same size. The visible image is two times larger than the infrared one so we need to resize these images to match the size of the infrared ones. It is more relevant to shrink the size of the visible data rather than increase the size of the infrared one since this does not introduce spurious accuracy.

The full final satellite image is a rectangle of 1250×625 pixels which makes 781 250 pixels to consider. This is a computational issue as the modelling requires significant memory and CPU resources. For this main reason, we will only focus on a small region of Europe, and more precisely Spain.

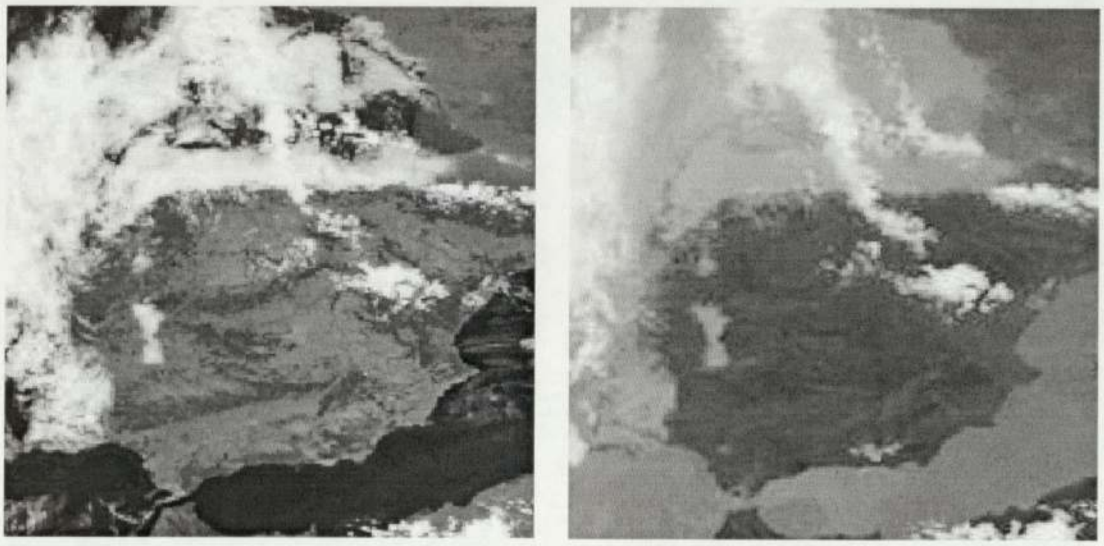

Figure 2.3: Visible and infrared satellite image of the working area taken on July, 14th 2003 at 12:00 am.

The choice of this working area is motivated by geographical reasons as we want to minimise the deformation of the image due to the curvature of the Earth. Furthermore, this region has also interesting land/sea and topographic features.

Chapter 3

Image processing

Before modelling the clouds, we need first to isolate them from the surface on the satellite images. At this stage, there is a need to build a binary classification to distinguish clouds from the underlying surface, land and ocean. The preprocessing is used to remove the surface pixels.

3.1 Mathematical background

As we want an automatic image processing method, we need to introduce a probabilistic algorithm to account for errors in the cloud screening phase. The probabilistic method is used to add some tolerance in the separation of the clouds and the surface.

3.1.1 The Gaussian mixture model

We consider the probability of a pixel being cloudy given its brightness. This will also take in consideration the basic surface type of each pixel when being non-cloudy, that is land or ocean. This probability will be represented by a Gaussian mixture model which is a combination of Gaussian densities [10]. The density has the following form for a model with M components:

$$p(\boldsymbol{x}) = \sum_{j=1}^{M} P(j) p(\boldsymbol{x}|j) \ , \tag{3.1}$$

where $p(\boldsymbol{x}|j)$ is the component density of the j^{th} basis function and $P(j)$ are the mixing coefficients which satisfy the constraints:

$$\sum_{j=1}^{M} P(j) = 1 \ ,$$

$$0 \leq P(j) \leq 1 \ .$$

Moreover, by choosing normalised density functions, we guarantee that the model does represent a density function, that is:

$$\int p(\boldsymbol{x}|j)\,\mathrm{d}\boldsymbol{x} = 1 \; .$$

In the Gaussian mixture model, the basis functions follow a Gaussian distribution. The usual choice is to consider a spherical covariance $\boldsymbol{\Sigma}_j = \sigma_j^2\mathbf{I}$ so that:

$$p(\boldsymbol{x}|j) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{\mu}_j||^2}{2\sigma_j^2}\right) \; .$$

3.1.2 The EM algorithm

To determine the parameters of a Gaussian mixture model from a data set, we need to maximise the data likelihood which is equivalent to minimising the negative log-likelihood:

$$E = -\mathcal{L} = -\sum_{n=1}^{N} \log p(\boldsymbol{x}_n) \; .$$

The expectation-maximisation (EM) algorithm iteratively modifies the Gaussian mixture model parameters, the mean $\boldsymbol{\mu}_j$, the variance σ_j^2 and the mixing coefficients $P(j)$ for each component j to decrease the error and find a local minimum value.

The choice of the EM algorithm is appropriate because it is simple to implement and to understand and it is usually faster to converge than general purpose algorithms [10].

In the Gaussian mixture model, we consider the data to be incomplete because we do not know which component j generated a given data point n. We thus introduce a variable $\boldsymbol{z}_n$, which takes on integer values in the range $[1...M]$, and denotes the unknown generating component [14].

Then using the product rule, the complete-data log-likelihood is given by:

$$\mathcal{L}^{comp}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log\{p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})P(\boldsymbol{z}_n|\boldsymbol{\theta})\} \; ,$$

where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_j, \sigma_j, P(j)\}$.

The Expectation step

Now, we need to take the expectation of the log-likelihood with respect to the distribution $P(\boldsymbol{z}) = \prod_{n=1}^{N} P(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{\theta}^{old})$. Since $\boldsymbol{z}$ is a discrete variable, the expectation over all $\boldsymbol{z}_n$ is simply a combination of N sums:

$$\mathcal{E}^{comp}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \left[\sum_{\boldsymbol{z}_1=1}^{M} \sum_{\boldsymbol{z}_2=1}^{M} \dots \sum_{\boldsymbol{z}_N=1}^{M} \prod_{m=1}^{N} P(\boldsymbol{z}_m|\boldsymbol{x}_m, \boldsymbol{\theta}^{old}) \log\{p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) P(\boldsymbol{z}_n|\boldsymbol{\theta})\} \right] ,$$

$$\mathcal{E}^{comp}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \left[\sum_{\boldsymbol{z}_1=1}^{M} \dots \sum_{\boldsymbol{z}_{n-1}=1}^{M} \sum_{\boldsymbol{z}_{n+1}=1}^{M} \dots \sum_{\boldsymbol{z}_N=1}^{M} \prod_{m \neq n}^{N} P(\boldsymbol{z}_m|\boldsymbol{x}_m, \boldsymbol{\theta}^{old}) \right] \tag{3.2}$$

$$\times \left[\sum_{\boldsymbol{z}_n=1}^{M} P(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{\theta}^{old}) \log\{p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) P(\boldsymbol{z}_n|\boldsymbol{\theta})\} \right] . \tag{3.3}$$

Since the first square-bracketed term (3.2) evaluates to unity as each of the individual sums $\sum_{\boldsymbol{z}_m=1}^{M} P(\boldsymbol{z}_m|\boldsymbol{x}_m, \boldsymbol{\theta}^{old}) = 1$ according to the constraints, we have the following expectation:

$$\mathcal{E}^{comp}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \sum_{\boldsymbol{z}_n=1}^{M} P(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{\theta}^{old}) \log\{p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) P(\boldsymbol{z}_n|\boldsymbol{\theta})\} .$$

Considering our previous notation for a mixture model, the expectation is equivalent to:

$$\mathcal{E}^{comp}(\boldsymbol{\theta}^{new}) = \sum_{n=1}^{N} \sum_{j=1}^{M} P(j|\boldsymbol{x}_n, \boldsymbol{\theta}_j^{old}) \log\{p(\boldsymbol{x}_n|j, \boldsymbol{\theta}_j^{new}) P^{new}(j)\} , \tag{3.4}$$

noting that $P(\boldsymbol{z}_n|\boldsymbol{\theta}^{new})$ is simply the prior $P^{new}(j)$.

The Maximisation step

In the M-step, we maximise $\mathcal{E}^{comp}(\boldsymbol{\theta}^{new})$ with respect to the parameters $\boldsymbol{\theta}^{new}$. So if we differentiate Equation 3.4 and set the derivatives to zero in the univariate Gaussian case, we get:

$$\boldsymbol{\mu}_j^{new} = \frac{\sum_{n=1}^{N} P(j|\boldsymbol{x}_n, \boldsymbol{\theta}_j^{old}) \boldsymbol{x}_n}{\sum_{n=1}^{N} P(j|\boldsymbol{x}_n, \boldsymbol{\theta}_j^{old})} ,$$

$$(\sigma_j^2)^{new} = \frac{\sum_{n=1}^{N} P(j|\boldsymbol{x}_n, \boldsymbol{\theta}_j^{old}) (\boldsymbol{x}_n - \boldsymbol{\mu}_j^{new})^2}{\sum_{n=1}^{N} P(j|\boldsymbol{x}_n, \boldsymbol{\theta}_j^{old})} ,$$

$$P^{new}(j) = \frac{1}{N} \sum_{n=1}^{N} P(j|\boldsymbol{x}_n, \boldsymbol{\theta}_j^{old}) ,$$

where $\boldsymbol{\theta}_j^{old} = \{\boldsymbol{\mu}_j^{old}, (\sigma_j^2)^{old}, P^{old}(j)\}$.

Each iteration of the Expectation and Maximisation steps is guaranteed to increase the likelihood, unless it is already at a maximum, so we simply repeat the E and M steps until our algorithm converges.

3.2 Finding the reference brightness

The goal of the preprocessing stage is to separate cloudy pixels from non-cloudy ones. For this purpose, we need a reference image of the working area without any clouds to make our binary comparison.

We know that clouds appear as the brightest pixels on visible satellite images. Moreover, they are moving so if for each pixel, we keep the lowest brightness over numerous images taken during a relevant period of time, we will be able to obtain an image cleared of clouds.

At that stage, we only need to consider the visible domain. Indeed, once the clouds are detected on the visible image, we will only have to keep the corresponding pixels on the infrared image. Moreover, for physical reasons, it is easier to distinguish clouds from the surface in the visible domain than in the infrared one. Actually, as the infrared channel measures the temperature, it can be confusing sometimes to distinguish clouds from the surface, especially for low clouds which have quite the same temperature as the sea surface for example. The only confusing distinctions on visible images concern snow regions and very reflective regions such as the Sahara desert, which can have the same radiance of the reflected sunlight as clouds. The first problem is handled by keeping the baseline brightness of the pixels using satellite images taken during summer as we remove the effect of snow regions as shown on Figure 3.1. The second problem has already been solved by the choice of the working area as Spain does not have very reflective land regions. The winter baseline brightness image is obtained with 28 images taken in January and February whereas the summer baseline brightness image is obtained with 27 images taken in June and July.

For the image processing, the most relevant baseline brightness to consider is the one obtained using summer images as we do not have to worry about confusions between clouds and snow radiances.

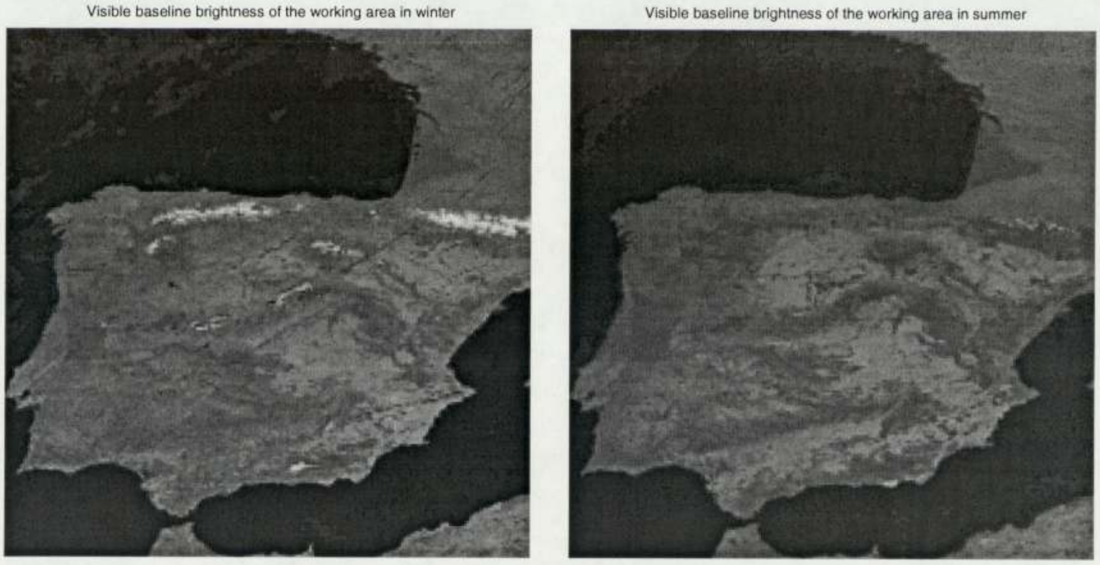

Figure 3.1: Baseline brightness of the working area in winter and summer.

3.3 Computing the probability of cloudiness

For each pixel, we want to classify it as cloudy or non-cloudy. But, as we are using an automatic model, we want some tolerance to account for errors. For this reason, we need to build a threshold based on the probability of a pixel being cloudy given its brightness and its basic surface type.

3.3.1 Fitting Gaussian mixture models to the data

As we plan to compute the probability of cloudiness given the basic category surface, we have to focus on labelled pixels to determine the influence of land or ocean surfaces. Actually, when plotting the histograms of the data, it appears that the global shape looks like a mixture distribution.

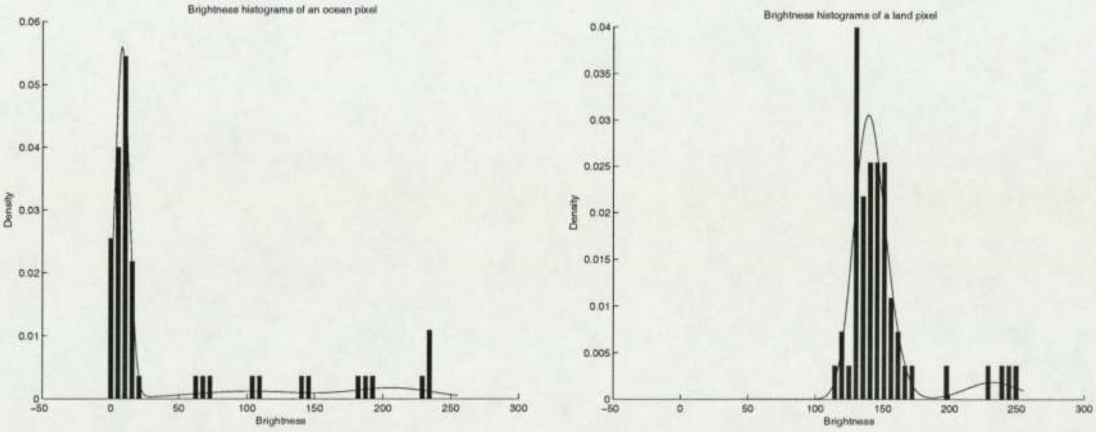

Figure 3.2: Brightness histograms of ocean and land pixels in the visible domain.

Considering the shapes of these histograms, we decided to fit a three component mixture model to the data, which should model the brightness of the basic background, the one of the low clouds and the one of the high clouds.

Moreover, we know that clouds appear brighter on the visible satellite images so we can assume that the first component of the Gaussian mixture represent the surface brightness. Then, retrieving the parameters of that Gaussian distribution, we want to define a probabilistic threshold to differentiate cloudy pixels from non-cloudy ones.

The computation of the Gaussian mixture model is made using the functions included in Netlab and the parameters are fitted using the EM algorithm also implemented in the Netlab package [10].

3.3.2 Defining a threshold for cloudiness

Given the parameters of the Gaussian mixture model, we are able to estimate the probability of each pixel being cloudy. We consider the Gaussian mixture having the smallest mean as the most probable representation of the basic brightness of that labelled pixel as land/sea pixels are darker than cloudy pixels on the visible satellite images. For that mixture j, we compute the posterior distribution using Bayes' theorem:

$$P(j|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|j)P(j)}{p(\boldsymbol{x})} \ ,$$

where $p(\boldsymbol{x})$ is given by Equation 3.1. This posterior probability satisfies the constraints

$$\sum_{j=1}^{M} P(j|\boldsymbol{x}) = 1 \ ,$$

$$0 \le P(j|\boldsymbol{x}) \le 1 \ .$$

The following graphics represent the posterior probability of the principal Gaussian mixture for each labelled pixel:

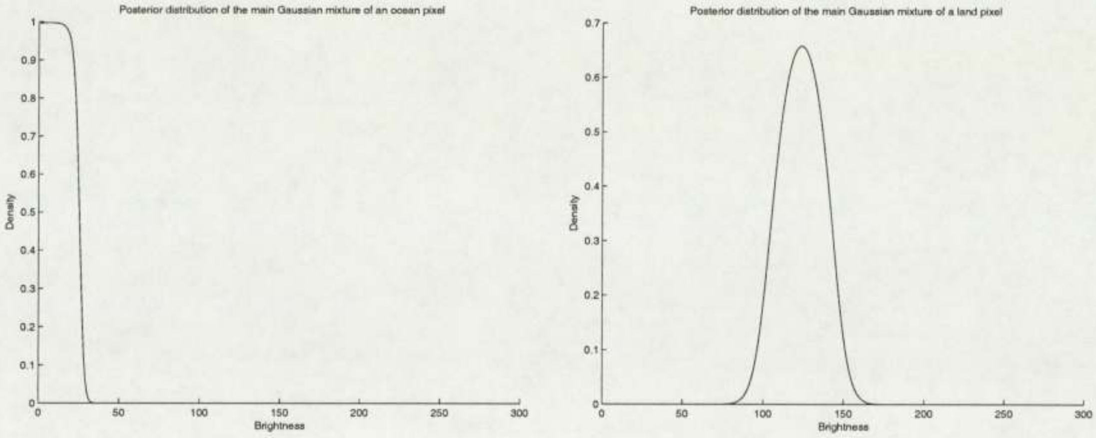

Figure 3.3: Posterior distribution of the main Gaussian mixture for ocean and land pixels.

Considering the posterior distribution of the main Gaussian mixture which denotes the probability of a pixel being non-cloudy, we can define a threshold for cloudiness which satisfy the rules below:

- If the brightness of the pixel is less than the centre of the Gaussian mixture, we consider that its probability of being cloudy is nil:

$$p(cloudiness) = 0 \ .$$

- If the brightness of the pixel is greater than the centre of the Gaussian mixture, we set its probability of being cloudy to be equal to:

$$p(cloudiness) = 1 - p(\overline{cloudiness}) = 1 - P(j|\boldsymbol{x}) \ .$$

These rules have to be set as the posterior distribution only gives a Gaussian probability of a pixel being a land or a sea pixel and not a threshold for distinguishing cloudy pixels from non-cloudy pixels. If we only consider the probability of cloudiness to be $1 - P(j|\boldsymbol{x})$, we will not get a threshold but the probability of a pixel not being a land or a sea pixel. This problem is obvious for the land pixel probability as $1 - P(j|\boldsymbol{x})$ stands for the probability of a pixel being a sea or a cloud pixel.

For each type of surface, we have a corresponding threshold which has the following form:

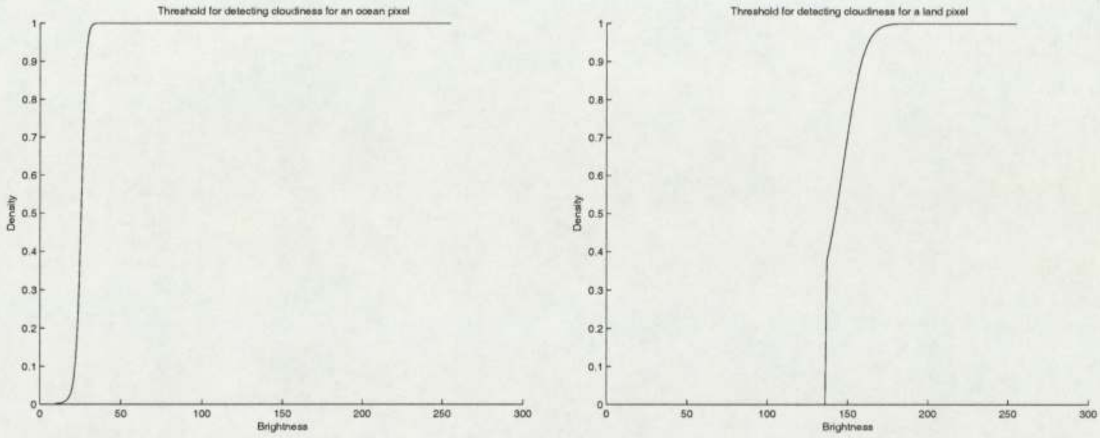

Figure 3.4: Threshold for cloudiness for each surface type.

3.4 Removal of the surface pixels

Once the threshold is defined, we are able to remove non cloudy pixels using the probability computed before. For that purpose, we need to compute the posterior brightness of the considered pixel obtained by multiplying the prior brightness with the probability of that pixel being cloudy given its brightness and the basic surface:

$$I^{new} = p(cloudiness) * I^{old} .$$

To preprocess the infrared image, we use the preprocessed visible image as a cloud mask. We only focus on the pixels referenced as cloudy in the visible domain and we keep the corresponding pixels in the infrared domain. More examples of the preprocessing can be seen in Appendix B.

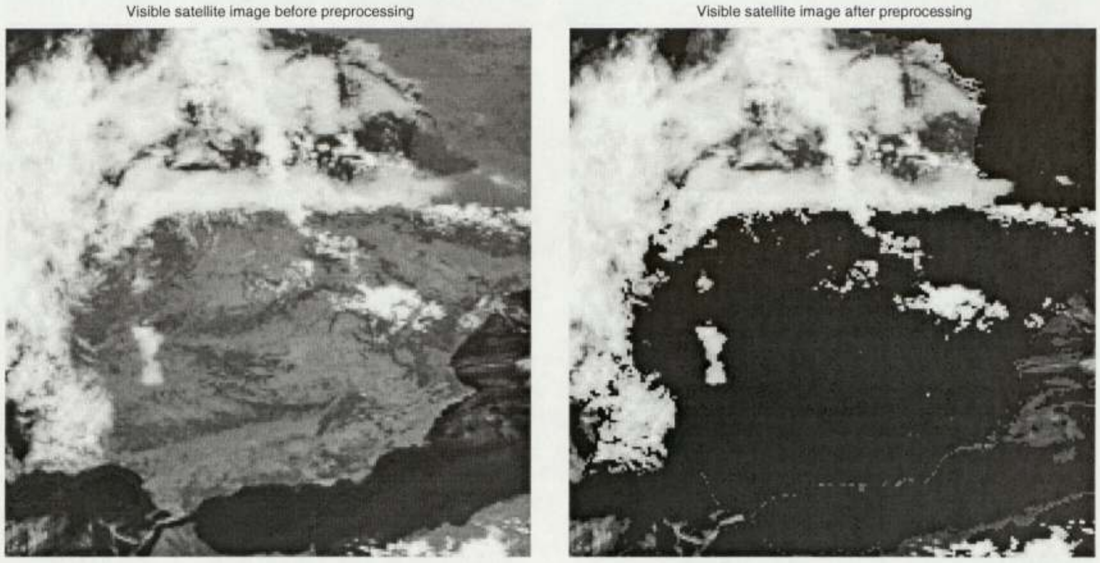

Figure 3.5: Visible image preprocessed.

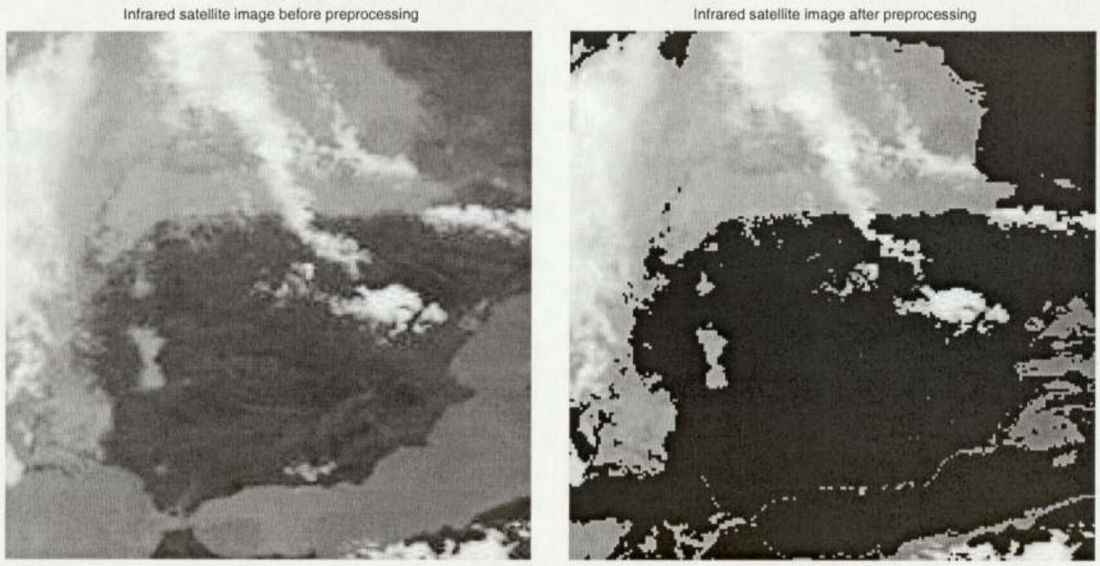

Figure 3.6: Infrared image preprocessed.

We can observe on Figures 3.5 and 3.6 that the coastlines pose some problems for the removal process. This phenomenon is due to the lack of accuracy of the registration of the satellite images taken. Indeed, when finding the baseline brightness of each pixel, if a small displacement of one pixel has occurred when taking the image, it can affect the algorithm. As we only keep the lowest brightness for each pixel, some land pixels can be assimilated to sea pixels near the coastlines which affects the probabilistic threshold. One way of solving these minor errors would be to use the cloud motion and their temporal behaviour. This model will be broached in Chapter 6.

Chapter 4

Cloud modelling

This chapter will deal with the modelling of the preprocessed image. To build our classification, we need to handle the clouds and the best way to do so is to model them by a radial basis functions neural network. Indeed, once it is done, we will only have to consider the hidden units of the network and no longer each pixel.

4.1 Theoretic support: The RBF network

Considering the nature and the shape of the clouds, a network that might fit the data well is a radial basis functions (RBF) network. This choice is motivated by its efficiency relating to density approximation, noisy interpolation and optimal classification theory [4]. Moreover, the RBF model is applicable to the dynamic model we wanted to create. As we need to study cloud motion, it is very convenient to focus on the motion of each basis function during time, assigning each basis function to a cloud.

The purpose of an RBF network is to approximate a given function h by using a linear combination of H non-linear basis functions Φ:

$$h(\boldsymbol{z}) = \sum_{i=1}^{H} w_i \Phi_i(||\boldsymbol{z} - \boldsymbol{z}_i||) + w_0 \ ,$$

where w_i is the weight associated to the i^{th} basis function, w_0 is the bias and $||\boldsymbol{z} - \boldsymbol{z}_i||$ is the Euclidean distance between $\boldsymbol{z}$ and $\boldsymbol{z}_i$.

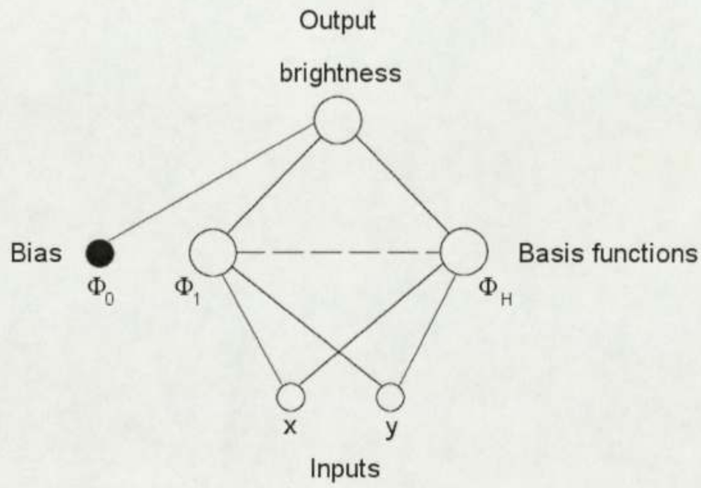

Figure 4.1: Radial basis functions network used for the cloud modelling. The two inputs are the coordinates of the pixel and the output is its brightness.

The architecture of the radial basis functions network is illustrated in Figure 4.1. Each basis function plays the role of a hidden unit. The weights are shown as lines connecting the basis functions to the output unit and the bias is shown as the weight from an extra 'basis function' Φ_0 whose output is fixed to 1.

Then, we shall write the radial basis functions network mapping of cloudiness in the following form:

$$\boldsymbol{C}(\boldsymbol{z}) = \sum_{i=1}^{H} w_i \Phi_i(\boldsymbol{z}) + w_0 \ .$$

The most common basis functions used to train the RBF are the Gaussian and the thin plate spline. Their characteristic shapes are shown in Figure 4.2.

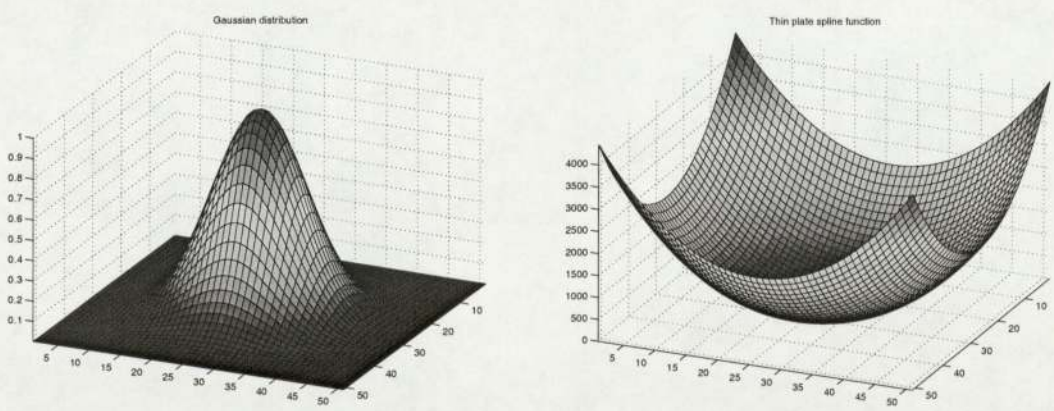

Figure 4.2: Gaussian and thin plate spline basis functions.

The Gaussian basis function can be expressed by:

$$\Phi_i(\boldsymbol{z}) = \exp\left(-\frac{||\boldsymbol{z}-\boldsymbol{\mu}_i||^2}{2\sigma_i^2}\right) ,$$

and the thin plate spline basis function by:

$$\Phi_i(\boldsymbol{z}) = ||\boldsymbol{z}-\boldsymbol{\mu}_i||^2 \log(||\boldsymbol{z}-\boldsymbol{\mu}_i||) ,$$

where $\boldsymbol{z}$ is the input vector, $\boldsymbol{\mu}_i$ is the vector determining the centre of the i^{th} basis function and σ_i is the associated width.

The Gaussian function is a localised basis function with the property that:

$$\lim_{\boldsymbol{z}\to\infty} \Phi(\boldsymbol{z}) = 0 .$$

This property is interesting for creating our model as clouds are compact and have clear edges. The thin plate spline function is derived from the theory of function interpolation and is an unbounded function that takes negative values for

$$0 \leq ||\boldsymbol{z}-\boldsymbol{\mu}_i|| \leq 1 .$$

The properties of the thin plate spline function do not seem to match with the modelling we envisage for clouds. This assumption will be verified by modelling tests done with different basis functions (see Table 4.1).

Hence, one of the advantages of the RBF network is the possibility to choose the basis function and actually, it is the main reason why we want to model clouds by an RBF network. This property makes the RBF network very versatile and makes it fit within the dynamic model framework we intended to build.

4.2 Modelling

The network chosen for the modelling of the clouds has two inputs which correspond to the coordinates of the pixel and one output which is its estimated brightness. Then, the next step for the build of the RBF network is the choice of basis function.

4.2.1 Choice of the basis function

Considering the nature and the shape of the clouds as shown in Figure 4.3, we want to create a basis function that matches them.

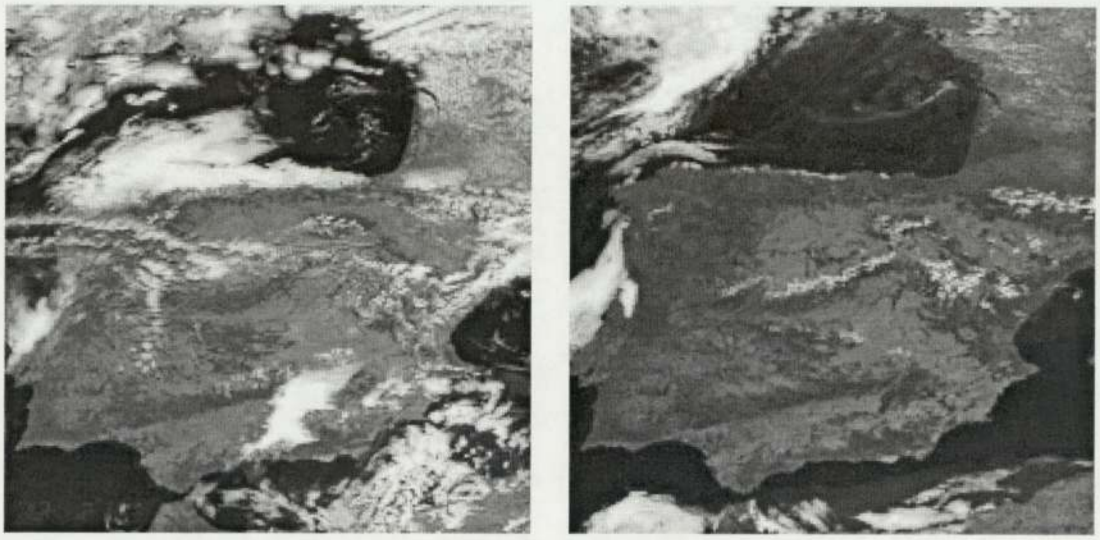

Figure 4.3: Clouds are usually optically thick and have clear edges.

Clouds have clear edges so we need a compact function to represent them. This function should increase and decrease rapidly as clouds are usually optically thick. And as we want to fit the shape and the orientation of the clouds, we want an elliptic function which can be parameterised. The following 'tanh' basis function fulfils these requirements:

$$\Phi(\boldsymbol{z}, \alpha, \beta, \gamma, \boldsymbol{\mu}, \sigma) = \frac{1}{2}[1 - \tanh(\alpha||z_x - \mu_x||^2 + \beta||z_y - \mu_y||^2 + \gamma||z_x - \mu_x||||z_y - \mu_y|| - 2\sigma)] .$$

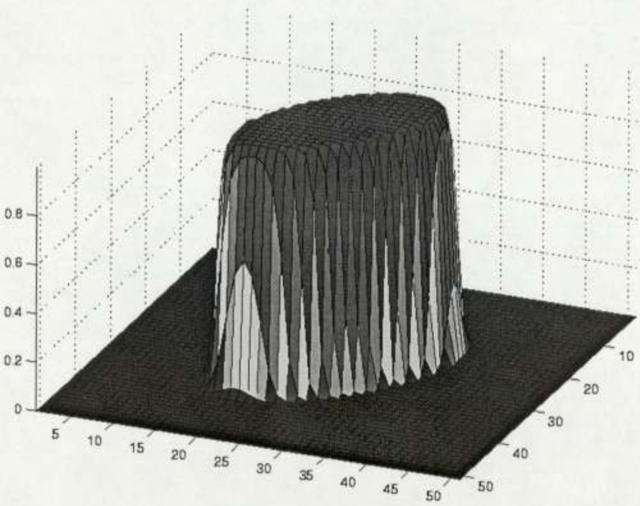

Figure 4.4: Basis function created for the cloud modelling.

To ensure the efficiency of the 'tanh' basis function, we have compared the network results obtained with that function with two usual basis functions already implemented in Netlab, the Gaussian distribution and the thin plate spline function.

	Time (in minutes)			Error		
Nb of hidden units	**Gaussian**	**tps**	**'tanh'**	**Gaussian**	**tps**	**'tanh'**
40	1.75	1.89	2.78	84 421	138 980	91 262
60	1.90	2.60	3.89	98 294	140 356	74 548
80	2.01	3.33	4.85	116 655	138 548	71 363
100	2.43	3.87	6.25	123 588	139 594	62 251

Table 4.1: Comparative table of the computational time and the network error of the RBF using three different basis functions, the Gaussian distribution, the thin plate spline (tps) function and the 'tanh' basis function defined in this thesis.

Table 4.1 gives a comparative report for the three basis functions considered. These tests were made on the same computer, using the same number of training iterations, the same initialisation method and are averaged over the same five images. We have featured the training time of the network for each function and the final error, which is the sum over all the pixels of the difference of brightness between the output network and the preprocessed satellite image brightness.

The results obtained tend to show that the basis function created is the one that fits the data the better. Indeed, if we focus on the final error, it appears that when the number of hidden units increases, the error for the new basis function decreases whereas the one for the Gaussian distribution increases and the one for the thin plate spline function stagnates. Moreover, focusing on a significant number of hidden units, the results show that the final error obtained with the basis function defined is smaller than the ones obtained with the other basis functions.

However, it appears that it takes more time to train the network with this new function, nearly two times longer than for the other basis functions. Nevertheless, the most important parameter for us is the final error as the smaller it is, the better will be the modelling.

4.2.2 Error and error gradient calculation

As we need a smooth representation of the satellite image, we want to optimise the network to obtain an output that fits the dataset the closest. For this purpose, we have to consider the error function, which denotes the difference between the output of the network and the real dataset, and minimise it. The error function is the usual sum-of-squares:

$$E = \frac{1}{2}\sum_{n=1}^{N}\{\boldsymbol{C}(\boldsymbol{z}_n;\boldsymbol{w}) - I_n^{new}\}^2 \ ,$$

where N is the number of patterns in the dataset, $\boldsymbol{C}$ is the activation of the output obtained by performing a forward propagation for the complete dataset and I^{new} is the real dataset obtained from the image processing.

We have considered this sum-of-squares error function as we assume the error to be Gaussian. Indeed, as we have lots of different errors due to the image caption and the image processing, the central limit theorem concludes that the sum of these errors can be assumed to be Gaussian.

As we want to optimise this function using an algorithm which utilises the gradient, we need to compute its derivatives with respect to the parameters. Indeed, when defining our RBF network, we have to consider the following parameters :

Parameter	**Definition**
μ	Centre of the basis function
σ	Width of the basis function
w	Output layer weight corresponding to the height of the basis function
b	Output-unit bias
α	Coefficient of the squared term along the abscissa axis
β	Coefficient of the squared term along the ordinate axis
γ	Coefficient of the convolution product

Table 4.2: Definition of the different parameters of our RBF network.

In this case, we have the following connection between the parameters and the 'tanh' basis function Φ resulting from the RBF mapping:

$$\boldsymbol{C} = \sum_{i=1}^{H} w_i \Phi_i(\boldsymbol{z}_n, \alpha, \beta, \gamma, \boldsymbol{\mu}, \sigma) + b \ ,$$

where H is the number of hidden units of the RBF network and $\boldsymbol{z}$ is the vector of coordinates (x,y) of the pixel n. The bias is used to deal with calibration issues and the cloud segmentation part.

When using the chain rule, the partial derivatives of $\boldsymbol{C}$ with respect to the parameters goes:

$$
\begin{aligned}
&\frac{\partial \boldsymbol{C}}{\partial \mu_i} = \frac{\partial \boldsymbol{C}}{\partial \Phi_i}\frac{\partial \Phi_i}{\partial \mu_i} = w_i \frac{\partial \Phi_i}{\partial \mu_i} \ , \\
&\frac{\partial \boldsymbol{C}}{\partial \sigma_i} = \frac{\partial \boldsymbol{C}}{\partial \Phi_i}\frac{\partial \Phi_i}{\partial \sigma_i} = w_i \frac{\partial \Phi_i}{\partial \sigma_i} \ , \\
&\frac{\partial \boldsymbol{C}}{\partial w_i} = \Phi_i \ , \\
&\frac{\partial \boldsymbol{C}}{\partial b} = 1 \ , \\
&\frac{\partial \boldsymbol{C}}{\partial \alpha_i} = \frac{\partial \boldsymbol{C}}{\partial \Phi_i}\frac{\partial \Phi_i}{\partial \alpha_i} = w_i \frac{\partial \Phi_i}{\partial \alpha_i} \ , \\
&\frac{\partial \boldsymbol{C}}{\partial \beta_i} = \frac{\partial \boldsymbol{C}}{\partial \Phi_i}\frac{\partial \Phi_i}{\partial \beta_i} = w_i \frac{\partial \Phi_i}{\partial \beta_i} \ , \\
&\frac{\partial \boldsymbol{C}}{\partial \gamma_i} = \frac{\partial \boldsymbol{C}}{\partial \Phi_i}\frac{\partial \Phi_i}{\partial \gamma_i} = w_i \frac{\partial \Phi_i}{\partial \gamma_i} \ .
\end{aligned}
$$

Moreover, the partial derivative of the error with respect to $\boldsymbol{C}$ is the same for all hidden unit activation functions:

$$\frac{\partial E_n}{\partial \boldsymbol{C}} = \boldsymbol{C} - I_n^{new} \ .$$

Let us label these values as $\delta_n = \boldsymbol{C} - I_n^{new}$ so that:

$$\frac{\partial E}{\partial \boldsymbol{C}} = \sum_{n=1}^{N} \frac{\partial E_n}{\partial \boldsymbol{C}} = \sum_{n=1}^{N} \delta_n \ .$$

Then according to the chain rule, the partial derivatives with respect to the output layer weights are:

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial \boldsymbol{C}}\frac{\partial \boldsymbol{C}}{\partial w_i} = \sum_{n=1}^{N} \delta_n \Phi_i \ ,$$

while the derivatives for the output-unit biases are given by:

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial \boldsymbol{C}}\frac{\partial \boldsymbol{C}}{\partial b} = \sum_{n=1}^{N} \delta_n \ .$$

Now considering the basis function Φ defined before, the derivatives of that function with respect to the other parameters are:

$$\frac{\partial \Phi_i}{\partial \mu_{xi}} = \left[\alpha_i(z_{xn} - \mu_{xi}) + \frac{\gamma_i}{2}(z_{yn} - \mu_{yi})\right]\left[1 - \Phi_i^2\right] \ ,$$
$$\frac{\partial \Phi_i}{\partial \mu_{yi}} = \left[\beta_i(z_{yn} - \mu_{yi}) + \frac{\gamma_i}{2}(z_{xn} - \mu_{xi})\right]\left[1 - \Phi_i^2\right] \ ,$$
$$\frac{\partial \Phi_i}{\partial \sigma_i} = 1 - \Phi_i^2 \ ,$$
$$\frac{\partial \Phi_i}{\partial \alpha_i} = -\frac{1}{2}||z_{xn} - \mu_{xi}||^2\left[1 - \Phi_i^2\right] \ ,$$
$$\frac{\partial \Phi_i}{\partial \beta_i} = -\frac{1}{2}||z_{yn} - \mu_{yi}||^2\left[1 - \Phi_i^2\right] \ ,$$
$$\frac{\partial \Phi_i}{\partial \gamma_i} = -\frac{1}{2}||z_{xn} - \mu_{xi}||||z_{yn} - \mu_{yi}||\left[1 - \Phi_i^2\right] \ .$$

Then, the chain rule gives the derivatives of the error function with respect to the parameters. For the centres, we have the following equations for each of the two components of the i^{th} centre:

$$\frac{\partial E}{\partial \mu_{xi}} = \frac{\partial E}{\partial \boldsymbol{C}}\frac{\partial \boldsymbol{C}}{\partial \mu_{xi}} = \sum_{n=1}^{N}\delta_n w_i \frac{\partial \Phi_i}{\partial \mu_{xi}} \ ,$$

$$\frac{\partial E}{\partial \mu_{yi}} = \frac{\partial E}{\partial \boldsymbol{C}}\frac{\partial \boldsymbol{C}}{\partial \mu_{yi}} = \sum_{n=1}^{N}\delta_n w_i \frac{\partial \Phi_i}{\partial \mu_{yi}} \ .$$

While replacing the derivatives of Φ with respect to the coordinates of the centre, we get:

$$\frac{\partial E}{\partial \mu_{xi}} = \sum_{n=1}^{N}\delta_n\left[\alpha_i(z_{xn} - \mu_{xi}) + \frac{\gamma_i}{2}(z_{yn} - \mu_{yi})\right]\left[1 - \Phi_i^2\right] \ ,$$

$$\frac{\partial E}{\partial \mu_{yi}} = \sum_{n=1}^{N}\delta_n\left[\beta_i(z_{yn} - \mu_{yi}) + \frac{\gamma_i}{2}(z_{xn} - \mu_{xi})\right]\left[1 - \Phi_i^2\right] \ .$$

In the same way, using the chain rule, we obtain the following derivatives for the other parameters, the width σ, and the coefficients α, β and γ:

$$\frac{\partial E}{\partial \sigma_i} = \sum_{n=1}^{N}\delta_n\left[1 - \Phi_i^2\right] \ ,$$
$$\frac{\partial E}{\partial \alpha_i} = -\frac{1}{2}\sum_{n=1}^{N}\delta_n||z_{xn} - \mu_{xi}||^2\left[1 - \Phi_i^2\right] \ ,$$
$$\frac{\partial E}{\partial \beta_i} = -\frac{1}{2}\sum_{n=1}^{N}\delta_n||z_{yn} - \mu_{yi}||^2\left[1 - \Phi_i^2\right] \ ,$$
$$\frac{\partial E}{\partial \gamma_i} = -\frac{1}{2}\sum_{n=1}^{N}\delta_n||z_{xn} - \mu_{xi}||||z_{yn} - \mu_{yi}||\left[1 - \Phi_i^2\right] \ .$$

4.3 Computation

The modelling of the clouds is done using Matlab programming based on the Netlab architecture for neural networks.

4.3.1 Computational issues

The image size of the working area is a square of 200×200 pixels. Experimentally, to obtain a smooth representation of the cloud field, we need at least 100 hidden units and 100 iterations to train the network. The straight-forward computation of such a network is nearly impossible with the machines at our disposal. Indeed, without any modifications of the data, the time between each iteration can approach five minutes and unfortunately, the program always crashes before the end of the training. The computer freezes because of lack of resources. If we analyse the memory problems, it appears that the main problem comes from the storage of the distances between each pixel and each basis function. To reduce the size of the matrices involved, we should decrease the number of pixels and hidden units to consider. The resizing of the satellite images will help reducing the number of pixels and the split of the images will reduce both the number of pixels and hidden units involved.

Resize of the satellite images

The first instinctive method to deal with computational problems is to reduce the resolution of the satellite images to train. The resizing of the images should not really affect our model as we are working with the basis functions and not with the pixels. The target image has to be smooth but not necessarily perfect because we are only interested in the global shape of the clouds. Anyway, the output of the network will never model the data perfectly so we can accept some loss of accuracy on the original image.

For training our network, we decide to reduce the resolution by four, leaving us with a square satellite image of 50×50 pixels. The resize consists in an average of the pixels brightness done every 4 pixels. We compute the average value for every square of 2×2 pixels. The following images show the results obtained while resizing the image.

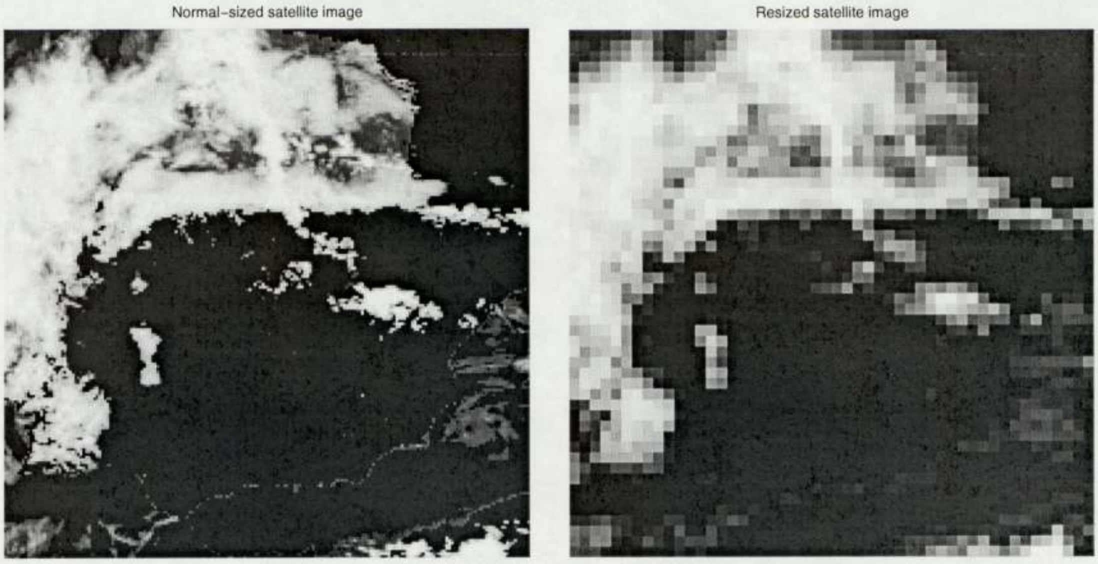

Figure 4.5: Satellite images taken before and after the resize process.

Split of the resized satellite images

Although the resize of the data is efficient to avoid early computational crashes, the time between each iteration is still long, about a minute. To reduce the training time, we split the images in four equal areas of 25×25 pixels for intensive trainings and then join them and train the whole image for a few iterations. This method helps to divide the computational expense by four.

These methods save a lot of resources and finally, we are able to have a smooth representation of the cloud fields rapidly, less than ten minutes of training with a large number of hidden units and iterations. This training time is acceptable as the dynamic framework involves the modelling of satellite images taken every ten minutes.

4.3.2 Initialisation

The initialisation of the network is a very important step in the modelling as it saves time and computational resources when training. Starting from the preprocessed images, the initialisation consists in setting relevant values to the parameters of the RBF network.

K-means initialisation

The K-means algorithm is a method for finding K vectors $\boldsymbol{\mu}_j$ (for $j = 1, ..., K$) that represent an entire dataset. The data is considered to be partitioned into K clusters, with each represented by its mean vector and each data point assigned to the cluster with closest vector [10].

The algorithm works iteratively. At each stage, the N data points $\boldsymbol{x}_n$ are partitioned into K disjoint clusters $\mathcal{S}_j$, each containing N_j data points. The error function minimised is the total within-cluster-sum-of-squares:

$$E = \sum_{j=1}^{K} \sum_{n \in \mathcal{S}_j} ||\boldsymbol{x}_n - \boldsymbol{\mu}_j||^2 ,$$

where $\boldsymbol{\mu}_j$ is the centre of the j^{th} cluster, given by the mean of the data points belonging to the cluster:

$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{n \in \mathcal{S}_j} \boldsymbol{x}_n .$$

This algorithm is traditionally used to initialise the RBF network. The following picture show the initialisation of the basis function on a preprocessed satellite image, where each cross represent the centre of each basis function.

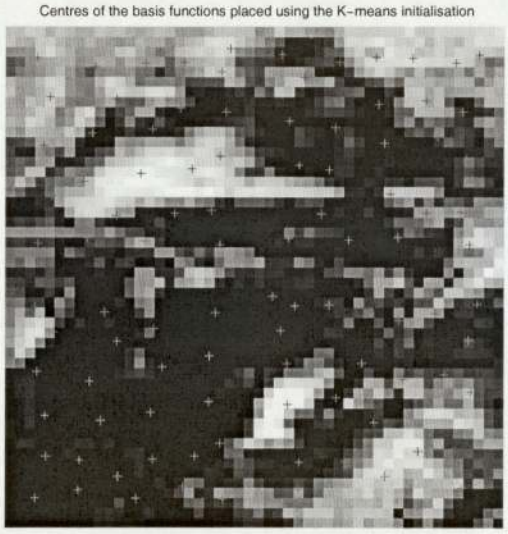

Figure 4.6: Initialisation of the basis functions using K-means algorithm.

The main drawback we can see to this initialisation is that it is not efficient for our cloud modelling. Indeed, with this algorithm, some basis functions are placed in non-cloudy areas whereas we only need basis functions to model clouds. This leads to consider more basis functions than necessary and is computationally expensive.

Adaptive initialisation

As the K-means algorithm is not efficient with our model, we need to create an algorithm that matches more our aims. First, we need an initialisation which only places basis functions on cloudy regions. Then, we also have to optimise the number of hidden units used and not putting more basis functions than needed.

The adaptive initialisation algorithm consists in assigning the first basis function centre to the brightest pixel, as clouds are brighter than the surface. Then we define a box of 9×9 pixels around that pixel and we train an RBF network of one hidden unit to that area to have a rough estimate of the parameters. The size of this box is the optimal size found experimentally as it is big enough to deal with located clouds and it is small enough to take into account the effect of small cloud features. Then we remove the effect of the basis function fitted and we choose the next remaining brightest pixel to be the second basis function centre and we iterate the processes until the number of hidden units chosen has been reached or until the sum of the brightness of the remaining pixels are less than a defined threshold. This limit helps to avoid putting too many basis functions when they are useless.

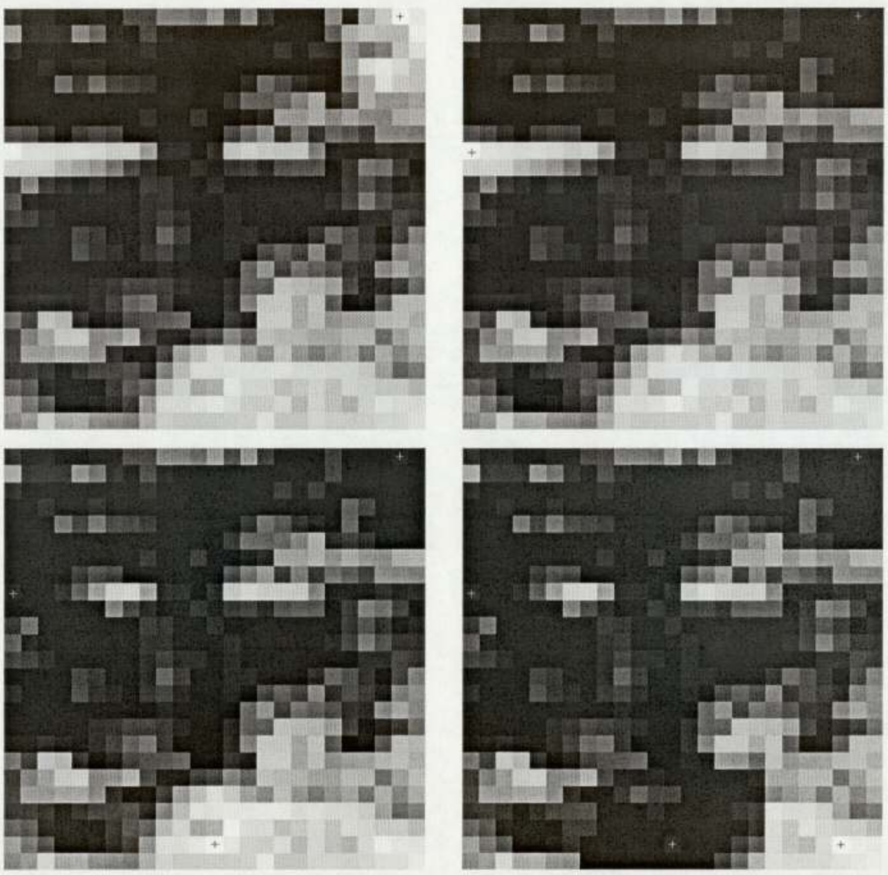

Figure 4.7: Initialisation steps of the cloud modelling for 4 hidden units. The crosses represent the centres of the basis functions.

The top left image of Figure 4.7 represents the initial region we want to model and the setting of the first basis function centre. The top right image shows the cloudiness after the removal of the effect of the first basis function as well as the setting of the second basis function centre. The left bottom image shows the cloudiness after the removal of the effect of the second basis function and the setting of the third basis function centre and the last image shows the cloudiness after the removal of the effect of the third basis function and the setting of the fourth basis function centre. This algorithm leads to the following initialisation for the basis functions as shown on Figure 4.8.

Figure 4.8: Initialisation of the basis functions using the adaptive algorithm.

As we can see, all the basis functions are assigned to a cloudy pixel on the contrary of the results obtained using K-means initialisation (see Figure 4.6). Moreover, less basis functions are set to model cloudiness, only 75 instead of 100 for the K-means initialisation. Our adaptive initialisation will clearly speed up the training of the network as the basis functions are already well placed and are only set when needed. Table 4.3 shows the comparison between the K-means initialisation and the adaptive initialisation in terms of training time and final error.

	Time (in minutes)		Error	
Nb of iterations	**K-means**	**Adaptive**	**K-means**	**Adaptive**
25	3.25	3.21	142 900	104 949
50	4.10	3.63	94 677	84 088
75	4.35	3.83	84 311	76 559
100	5.07	4.15	77 151	70 194

Table 4.3: Comparative table of the computational time and the network error of the RBF using the K-means initialisation and the adaptive initialisation.

These results confirm the efficiency of the adaptive initialisation as for the same number of iterations, the training time and the final error obtained are smaller.

We can also notice that for some regions, close basis functions are set to model cloudiness. This can be explained by the split of the image which can lead to very close basis function centres when we join the four regions. The threshold defined for stopping the setting of the basis functions plays an important role. This threshold is based on the sum of the brightness of the remaining pixels and is arbitrary chosen. The optimal value has to be found experimentally as it can vary between different images. The lower the value of the threshold, the more basis functions are to be set.

4.3.3 Optimisation

The optimisation of our RBF network is made using the scaled conjugate gradient algorithm, which is used to minimise the error between the output of the network and the data. This choice was made after comparing the performance of this algorithm with the other common optimisation algorithms, such as gradient descent, conjugate gradient and quasi-Newton algorithms.

While training the network using gradient descent and quasi-Newton methods, we get very poor results and the models do not fit the satellite image. Finally, the only efficient algorithms we could use to create our model are the conjugate gradient and the scaled conjugate gradient algorithms. Table 4.4 shows a comparison between these two methods. The computational time and the final error defined before are taken into account. For more relevant results, these tests were applied for different numbers of training iterations and each result is averaged over the same five images with the same training conditions.

	Time (in minutes)		**Error**	
Nb of iterations	**CONJGRAD**	**SCG**	**CONJGRAD**	**SCG**
25	14.32	3.21	92 161	104 949
50	15.51	3.63	90 902	84 088
75	16.26	3.83	85 765	76 559
100	17.77	4.15	84 449	70 194

Table 4.4: Comparative table of the computational time and the network error of the RBF using the conjugate gradient and the scaled conjugate gradient algorithms.

These results show clearly that the scaled conjugate gradient algorithm is better for training our network. The training is four times faster with this method than with the conjugate gradient method. Moreover, the final error obtained is smaller for a significant number of iterations and it decreases faster at each iteration.

4.3.4 Network results

Having implemented the theory defined before, we are now able to model the clouds using an RBF network with the 'tanh' basis function defined in Section 4.2.1. All the computational issues described before are used to enhance the training of the network. The results obtained are shown on Figure 4.9 where the image on the left is the preprocessed satellite image and the one on the right is the modelling obtained when training the RBF.

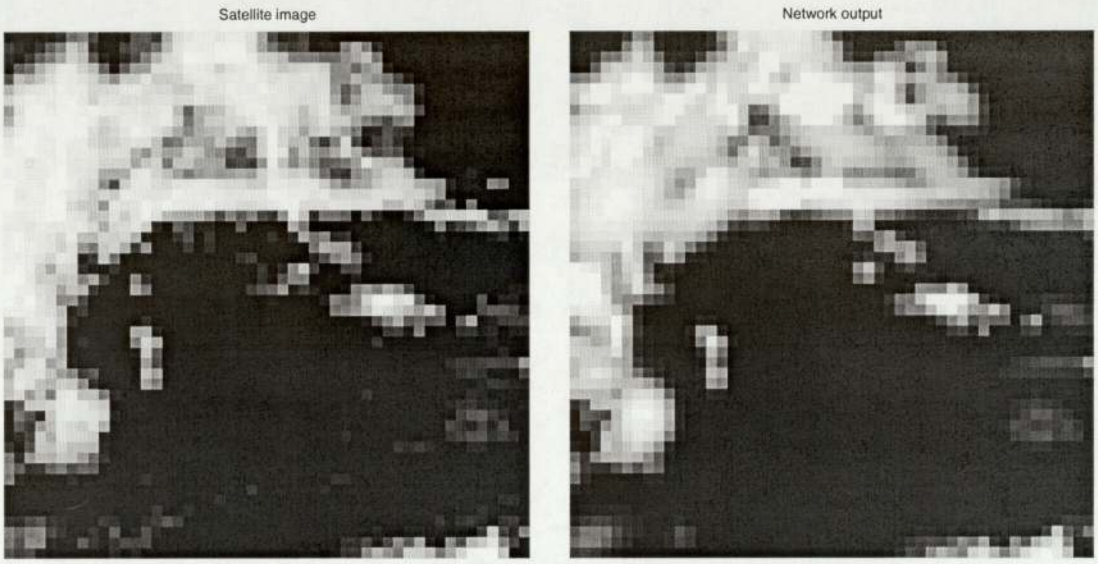

Figure 4.9: Modelling of clouds by an RBF neural network.

These results were obtained with a Pentium III computer running at 933 MHz with 256 Mb of memory. The image was modelled with 100 hidden units. Using the split process defined before, for each of the four areas, we have trained the network with 200 iterations. Then, when joining these areas, the full image has been trained for another 50 iterations to remove any graphical inaccuracies due to the previous split of the image. The training process has taken 6.54 minutes to run. The final error obtained is equal to 32 148, which gives an average error of 12.8 for the brightness of every pixel as we are working over 2,500 pixels. As the brightness takes values between 0 and 255 gray-levels, the modelling leads to an error of 5% between the brightness output of the pixel and its real satellite image brightness. More examples of the cloud modelling can be seen in Appendix C.

The cloud modelling is one of the original pieces of work done for this thesis. The use of an RBF neural network is motivated by the dynamic classification we wanted to build. To study the cloud motion, we decided to use the RBF network in order to be able to focus on the motion of each basis function during time. Our modelling introduced the implementation of a new basis function to fit the shapes and the orientation of the clouds. This function has the property of being anisotropic and this is shown on Figure 4.10, where a few basis functions are highlighted on the network output image.

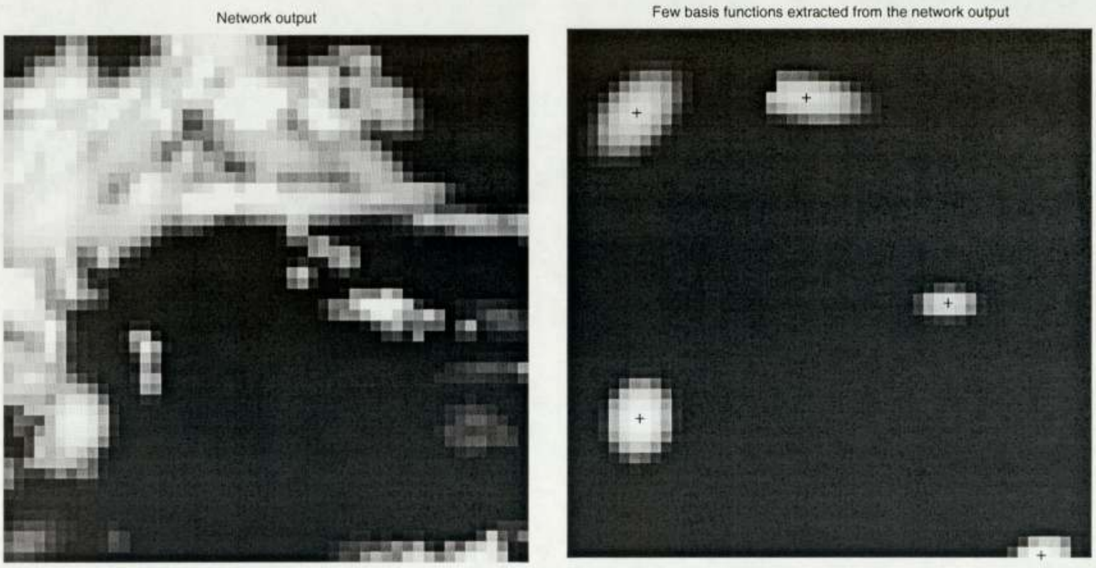

Figure 4.10: Highlight of a few number of basis functions used for the cloud modelling.

The results obtained confirm the efficiency of this basis function compared to the other common basis functions existing (see Table 4.1). Lots of computational problems have been encountered for the modelling and have been handled as described in this chapter. Hence, we manage to model the cloudiness using a reliable and rapid algorithm.

Chapter 5

Cloud segmentation

The initial aim of the research done for this thesis was to classify clouds into labelled classes. To achieve this retrieval, we planned to build a space-time model for clouds, dealing with the dynamics of each cloud modelled by a radial basis function. The first step to this dynamic retrieval is to set assumptions about cloud height so that we can apply the specific motion model defined for each class of clouds. This chapter will deal with a classic cloud segmentation using visible images as cloud masks and infrared images to distinguish cloud height.

5.1 Setting thresholds for cloud height

The classification method used to retrieve clouds is a spectral threshold method. As the infrared images give information about the temperature of the clouds, and as we know that the higher they are in the atmosphere, the colder they are, we are able to classify them according to their heights. This method combines visible and infrared images.

Having the preprocessed infrared satellite image, the first step in the construction of this segmentation is to find the boundaries separating each class of clouds. To achieve this objective, we decided to plot the histograms of the brightness of each cloudy pixel in the infrared domain. The results are obtained using the data of ten images.

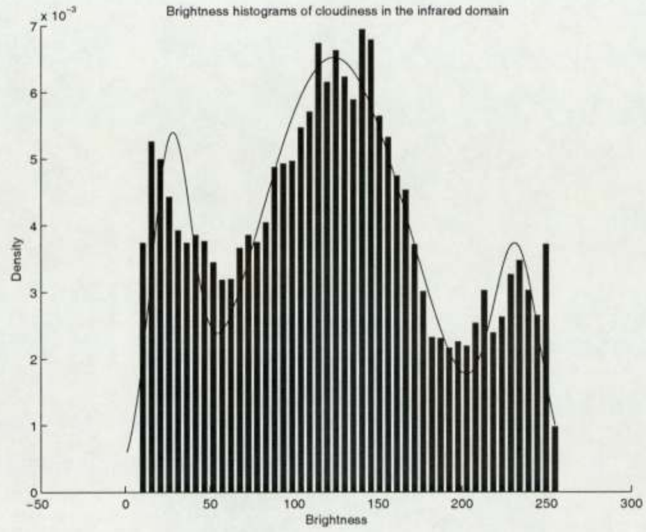

Figure 5.1: Brightness histograms of cloudiness in the infrared domain.

The results obtained on Figure 5.1 show the emergence of three distinct classes in the brightness. As the shape of the histograms looks like a mixture distribution, we decide to fit a three-component Gaussian mixture model to the data.

As we are interested in the influence of each component to set our thresholds, we need to compute the posterior distribution for each component, as described in Chapter 3. This gives the following graphics:

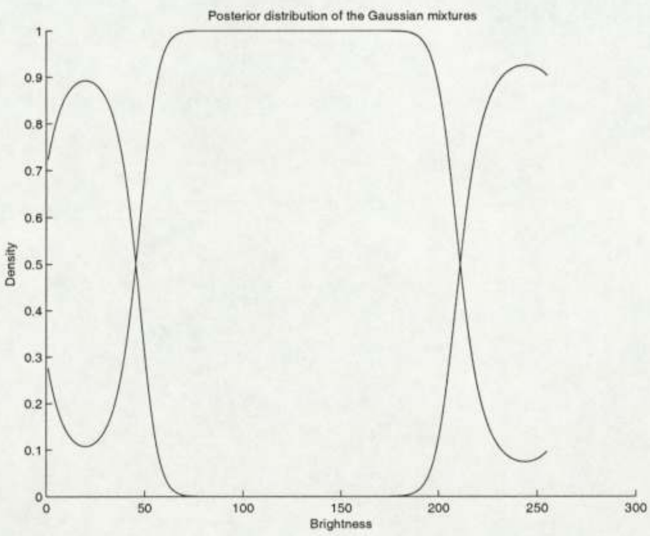

Figure 5.2: Posterior distribution of the Gaussian mixtures used to fit the cloud classes.

The analysis of these distributions leads to the definition of three classes for distinguishing clouds, low level clouds, medium level clouds and high level clouds. The thresholds needed to separate these classes can be set to the intersection of each posterior probability of Figure 5.2. Indeed, if the brightness of a pixel is lower than the first threshold, it is most probable that this pixel belongs to the first class. If the brightness is between the two thresholds, it is most probable that the pixel belongs to the second class and if the brightness is greater than the second threshold, it is most probable that the pixel belongs to the third class.

Practically, the thresholds are computed by finding the first integer in the brightness range of 256 gray-level values for which the first posterior probability becomes less than the second posterior probability. We end up with two boundaries which define three different classes.

The method used will yield to the definition of generalised thresholds for the segmentation. We could have also defined these thresholds by plotting the histograms for each image. The advantage would have been the definition of more accurate thresholds. But the drawback would have been the lack of data for some images and the shapes of the histograms would not have lead to three separated classes. This problem can appear for images which are not very cloudy or for images which do not have an equal sample of the different types of clouds.

5.2 Cloud height segmentation

Continuing on the original idea of our thesis, we are going to classify, not each pixel but each basis function modelling cloudiness. The classification is made referring to the brightness intensity of the centre of the basis function. Indeed, the centre has the highest intensity of the cloud cover modelled by that basis function, according to the spatial properties of the basis function defined in Section 4.2.1. We know that the distribution of the basis function is maximum at the centre and so is the brightness defined by the forward propagation in the RBF network:

$$y(\mu) = w\Phi(\mu) + b \ ,$$

where μ is the centre, w is the output layer weight and b is the bias.

Applying the thresholds to the brightness intensity of each centre of each basis function gives us a classification of cloudiness. Nevertheless, we can observe experimentally that low clouds cannot be retrieved. This can be explained by the fact that we are only focusing on the brightness intensity of the centres whereas the low brightness obtained on the histograms should be the ones of the pixels representing the edges of the clouds or some land/sea pixels which come from errors occurred after the image processing. As a consequence, we will only segment the clouds into two different height levels.

Before being able to classify the clouds, we need to model the infrared preprocessed image using an RBF network as described in Chapter 4. This modelling gives the following results.

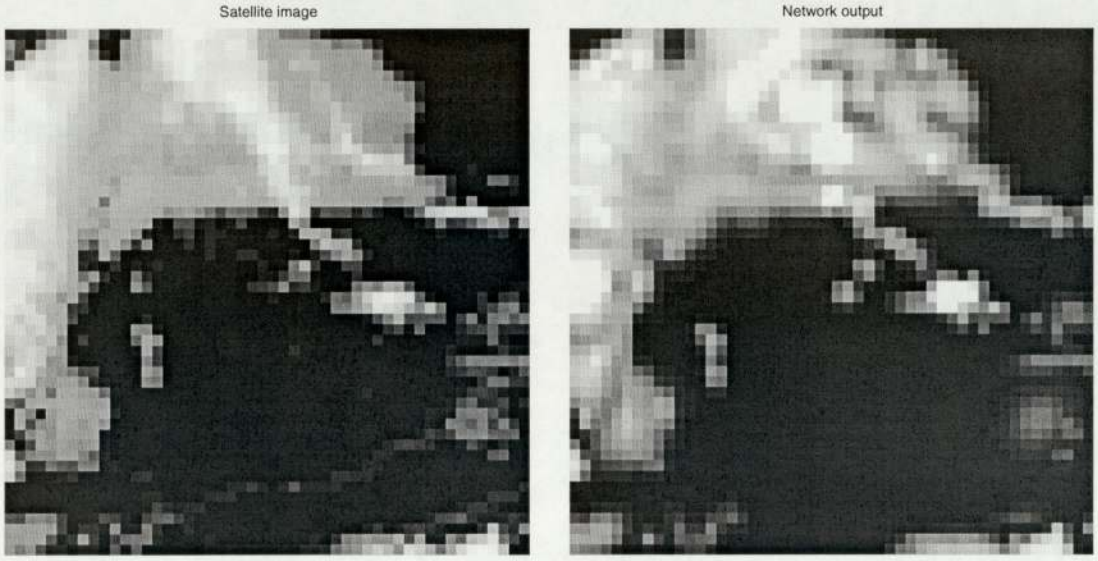

Figure 5.3: Cloud modelling by an RBF network in the infrared domain prior to a cloud segmentation.

Figure 5.4 gives the results obtained when applying the classification developed before. Each basis function modelled by the radial basis functions network is assigned to one of the class defined by the thresholds set in Section 5.1. The centre of each basis function is also plotted on the figures to make them more understandable. More examples of the cloud classification can be seen in Appendix D.

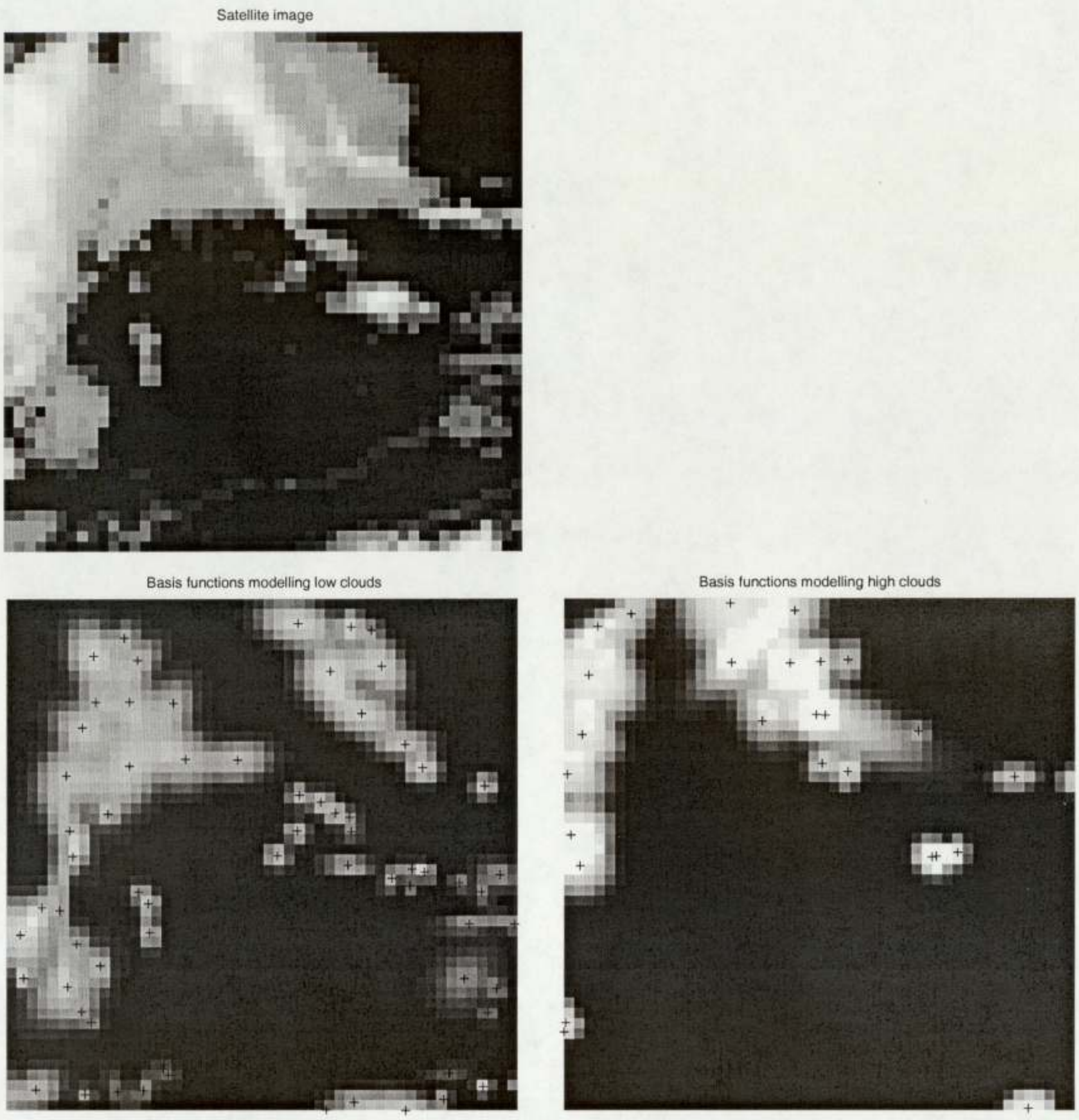

Figure 5.4: Segmentation of the basis functions into two classes representing the height of the clouds modelled in the atmosphere.

The segmentation built in this thesis in relatively basic and is not really reliable to distinguish three classes of cloud height. However, if we only define two classes, low and high, the classification gives good results. Nonetheless, this method has the same main drawback as most of the other cloud classification methods as it does not take into account the different layers of clouds. Solving this issue was one of the main motivation to consider the dynamic model which will be approached in the next chapter.

Chapter 6

Conclusions

This chapter will deal with an overview of the research achieved for the thesis as well as a further possible development introducing a temporal approach to the spatial model defined.

6.1 Achievements

The initial aim of this thesis was to build a space-time model to classify clouds. However, the final objective has not been attained due to a lack of time. All the research made in this thesis was focused on creating an appropriate framework to introduce dynamics.

The preprocessing of the satellite images was an important step in our modelling as we managed to remove the underlying surface pixels and only keep cloudy pixels. This was done by using a probabilistic method involving the definition of brightness intensity thresholds. Using a probabilistic method helps to limit errors in the preprocessing as it allows some tolerance in the separation of the pixels.

Once the image processing had been achieved, we managed to model cloudiness using a radial basis functions network. The choice of this network was motivated by the fact that we wanted to be able to focus on the motion of the basis functions in the temporal framework. The cloud modelling needs the introduction of a new basis function which fits the physical properties of the clouds better than the usual basis functions. The RBF network takes the coordinates of the pixels as inputs and returns the estimated brightness intensity in output. The modelling also takes into account computational issues by resizing the data and by developing an adaptive initialisation.

The model defined in this way should be able to fit the dynamic model defined in the next section. However, a standard cloud segmentation using infrared data has been implemented to improve the dynamic cloud classification. The method used is a spectral threshold classifier which retrieves clouds according to their height in the atmosphere. This classifier gives good results for distinguishing two classes of clouds, but not three as envisaged.

A further development to this thesis would be to introduce the temporal behaviour of clouds. This method would help distinguish the three main classes of clouds as each type of clouds has its own motion property. Moreover, this method would solve the cloud layers issues.

6.2 Improvements

The initial aim of the project was to build a space-time model. This model has not been implemented but the theory has been elaborated. This model was proposed by Dan Cornford for precipitation forecasting [6] and developed by Emmanuel Batail last year [3]. A theoretical approach can be seen in Appendix A. This framework should be useful to classify cloud types. Combining the cloud height segmentation defined in Chapter 5 with the dynamic model involving cloud motion, we should be able to retrieve the three main cloud types, cirrus, stratus and cumulus.

The research done in this thesis has lead to interesting results but some points can still be improved. The image processing has been made using visible images taken during the day. As a consequence, the definition of cloud masks will not be possible for visible images taken at night. This problem can be solved by using another channel of the METEOSAT satellite for setting cloud masks. The use of water vapour images can help as they are not affected by the illumination of the Sun although this mask will not be really accurate as there can be humidity without clouds. Another method would be to consider only the infrared images but this can also lead to poor results in defining the cloud masks as some regions can have nearly the same temperature as clouds such as sea or mountain regions. Some problems also occurred with thin clouds as they are transparent and very difficult to distinguish from the underlying surface on the visible images. Some research has to be done in this area to improve the image processing.

The cloud modelling has introduced the definition of a new basis function which has been created to match very closely the shapes and the physical properties of the clouds. During the research period, other basis functions such as quadratic functions have been tested but did not give the same good results. This basis function fulfils the constraints expected for the modelling of clouds. The definition of this function is the key point of this thesis as it conditions the cloud modelling and ultimately the cloud retrieval. The 'tanh' function presented in this thesis is the best function tested during the research period.

The computational methods used for the modelling are simple and consist in resizing or dividing the data. Some new algorithms could be tested to improve the training time, by splitting the images according to the cloud cover of an area for example. This could make use of the entropy to define areas of the images to train. The adaptive initialisation used in this thesis only initialises the centres but not the other parameters. Improvements have to be done in order to speed up the training.

The cloud segmentation used is traditional but gives good results for distinguishing cloud height. This segmentation does not need to be very accurate as it will only be used to assume the different classes of the basis functions prior to the dynamic classification. Unfortunately, this segmentation does not take into account the different cloud layers and more research has to be conducted. Hopefully, the introduction of the spatial model could solve this problem.

Finally, another direction to consider in our research is to define an integrated model where the reflectance of the reflected sunlight for each pixel is obtained by integrating the cloudiness distribution over that pixel:

$$I_{ij} = \int_x \int_y \boldsymbol{C}(x, y) \mathrm{d}x \mathrm{d}y \ .$$

This model should improve and smooth the results obtained and actually it is the model used by METEOSAT to convert the data given by the radiometers into images.

Appendix A

The dynamic model

A.1 The advection equation

Following the work made before for the cloud modelling using an RBF network in Chapter 4, we can represent cloudiness by a weighted sum of H basis functions:

$$\boldsymbol{C}(\boldsymbol{z}) = \sum_{i=0}^{H} w_i \Phi_i(\boldsymbol{z}; \boldsymbol{\mu}, \sigma) \ . \tag{A.1}$$

Then, the dynamic model combining space and time follows the advection equation which gives the evolution of $\boldsymbol{C}$:

$$\frac{\partial \boldsymbol{C}}{\partial t} + \boldsymbol{v} \nabla \boldsymbol{C} \approx 0 \ ,$$

where $\boldsymbol{v}$ represents the advection vector of the clouds. The cloud classification will be made according to the advection vector as each different type of clouds has its own motion.

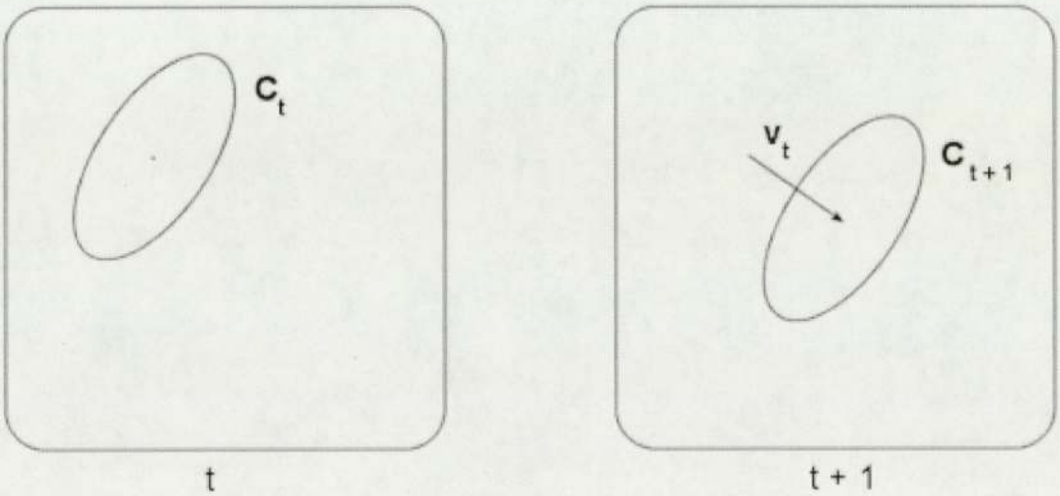

Figure A.1: The advection vector measures the movement of the clouds during time.

A.2 The time model framework

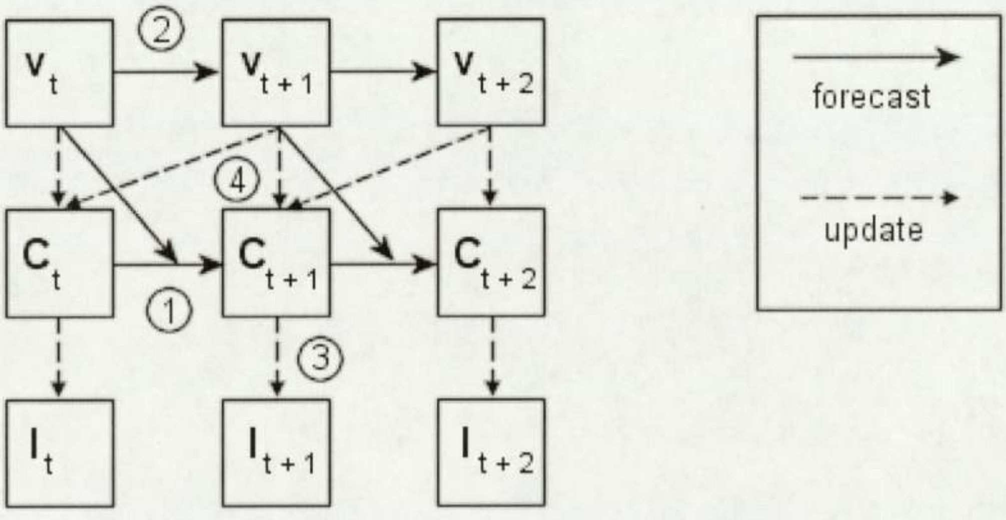

Figure A.2: Graphical model of the framework.

The framework defined by Figure A.2 shows the links between the different parameters, $\boldsymbol{v}$, the advection vector, $\boldsymbol{C}$, the cloudiness and $\boldsymbol{I}$, the brightness of the clouds. This framework develops four general steps including forecast and update of the parameters.

Initial step

The initial step is to estimate $\boldsymbol{v}_t$ so that we can compute the first step. To obtain this value, we need to know $\boldsymbol{C}_{t-1}$ and $\boldsymbol{C}_t$. Actually, this initialisation phase correspond to the fourth step of the graphical framework. $\boldsymbol{C}_{t-1}$ and $\boldsymbol{C}_t$ are easily found by using the RBF modelling for two different images very close in time. Then, $\boldsymbol{v}_t$ is the movement of the cloud during the period of time separating each image.

First step

The first step of the model is to forecast the cloudiness at the state $t+1$. Following the equation A.1, $\boldsymbol{C}_{t+1}$ can be obtained by computing the parameters of the equation at the state $t+1$. Working in a fixed area and having fixed parameters for the neural network that do not depend on time, we have:

$$w_{i,t+1} = w_{i,t} \ ,$$

$$\sigma_{i,t+1} = \sigma_{i,t} \ .$$

Then, if we assume that $\boldsymbol{v}$ is locally constant because of the slow motion of the clouds, we have the following relationship between the centres:

$$\boldsymbol{\mu}_{i,t+1} = \boldsymbol{\mu}_{i,t} + \boldsymbol{v}_t \delta t + \epsilon_\mu \ ,$$

where ϵ_μ is the error on the forecast due to the simplifications of the model and that not all apparent cell motion is due to advection. Finally, the cloudiness at the state $\boldsymbol{C}_{t+1}$ is:

$$\boldsymbol{C}_{t+1}(\boldsymbol{z}) = \sum_{i=0}^{H} w_{i,t+1} \Phi_{i,t+1}(\boldsymbol{z}; \boldsymbol{\mu}_{i,t+1}, \sigma_{i,t+1}) \ .$$

Second step

The second step of the model is to forecast the advection vector at the state $t+1$ knowing $\boldsymbol{v}_t$, using the equation:

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \epsilon_v \ ,$$

where ϵ_v reflect the fact that $\boldsymbol{v}$ changes slowly.

Third step

The third step of the model consists in updating the cloudiness $\boldsymbol{C}_{t+1}$ by fitting the modelling made to the real observation which is the cloud brightness of the satellite images. This update is done in a probabilistic way using Bayes' rule to find the updated parameters:

$$p(w, \boldsymbol{\mu}, \sigma|\boldsymbol{I}) = \frac{p(I|w, \boldsymbol{\mu}, \sigma)p(w, \boldsymbol{\mu}, \sigma)}{p(\boldsymbol{I})} ,$$

which gives, using probability properties:

$$p(w|\boldsymbol{I})p(\boldsymbol{\mu}|\boldsymbol{I})p(\sigma|\boldsymbol{I}) = \frac{p(\boldsymbol{I}|w, \boldsymbol{\mu}, \sigma)p(w)p(\boldsymbol{\mu})p(\sigma)}{p(\boldsymbol{I})} .$$

Fourth step

The fourth step of the model is the update of the advection vector $\boldsymbol{v}_{t+1}$ using $\boldsymbol{C}_t$ and the updated $\boldsymbol{C}_{t+1}$. Then, we come back to the first step and continue the algorithm as long as we have images.

Appendix B

More examples of the preprocessing

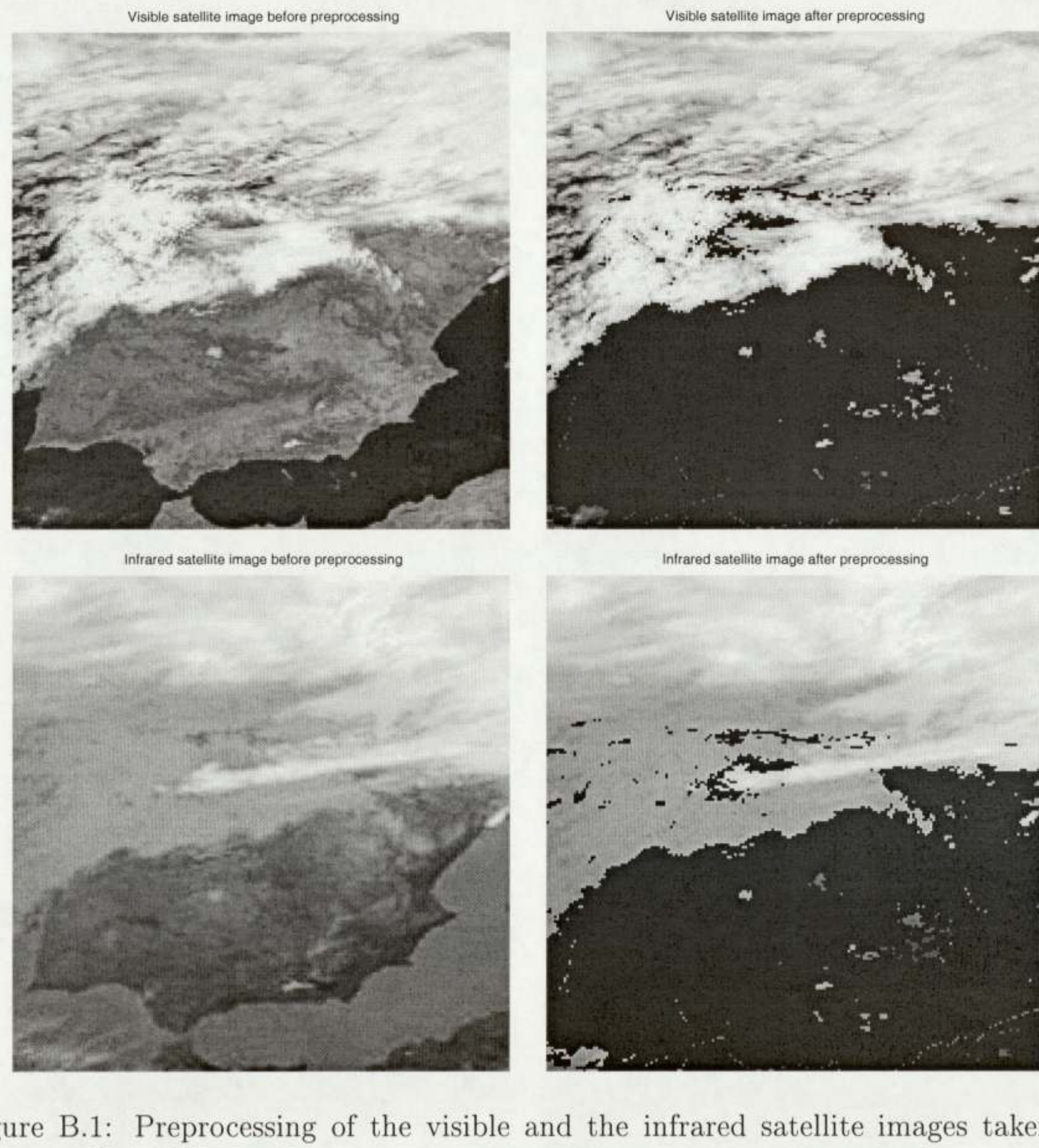

Figure B.1: Preprocessing of the visible and the infrared satellite images taken on February, 3rd 2003.

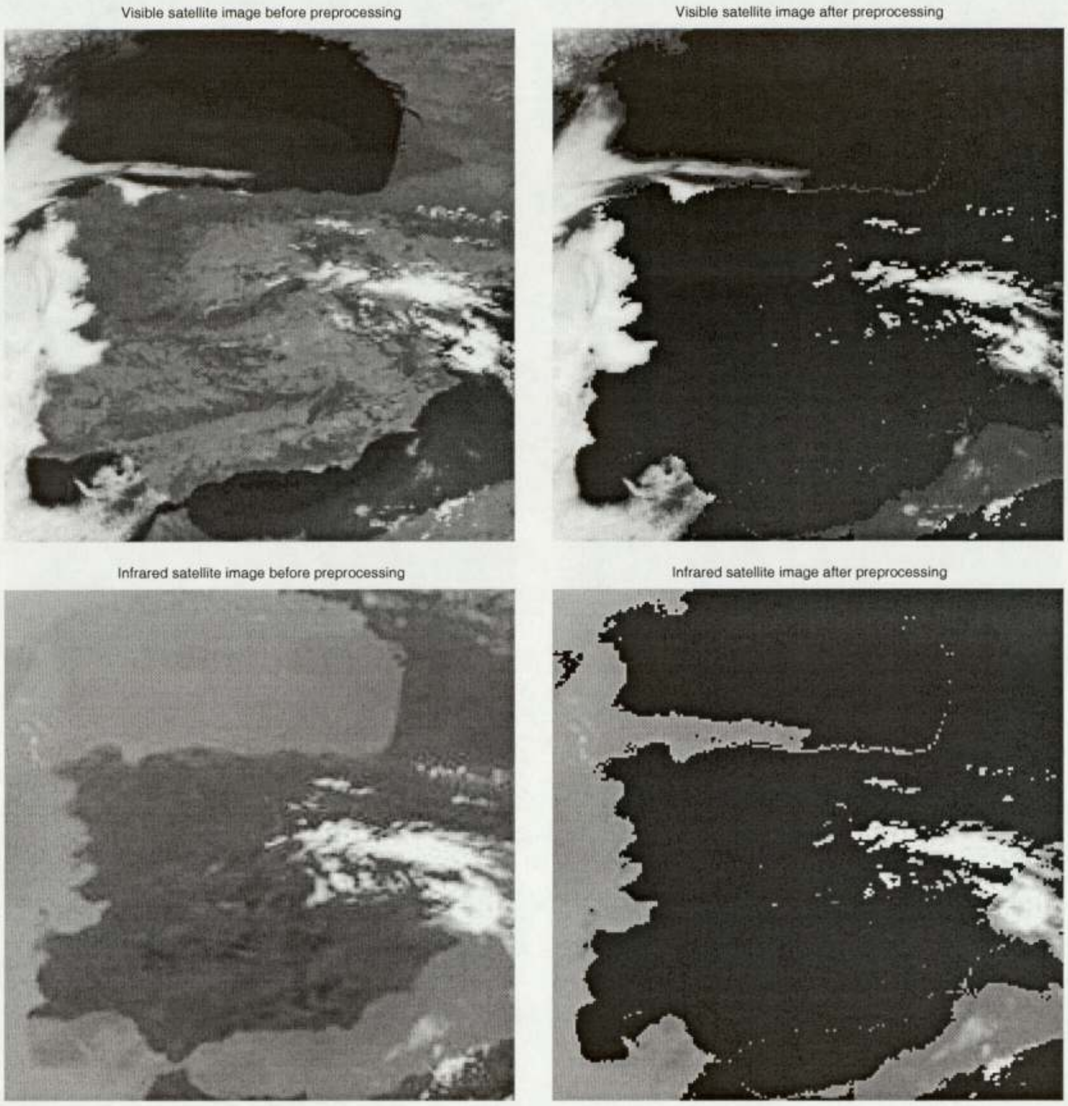

Figure B.2: Preprocessing of the visible and the infrared satellite images taken on July, 12th 2003.

Appendix C

More examples of the modelling

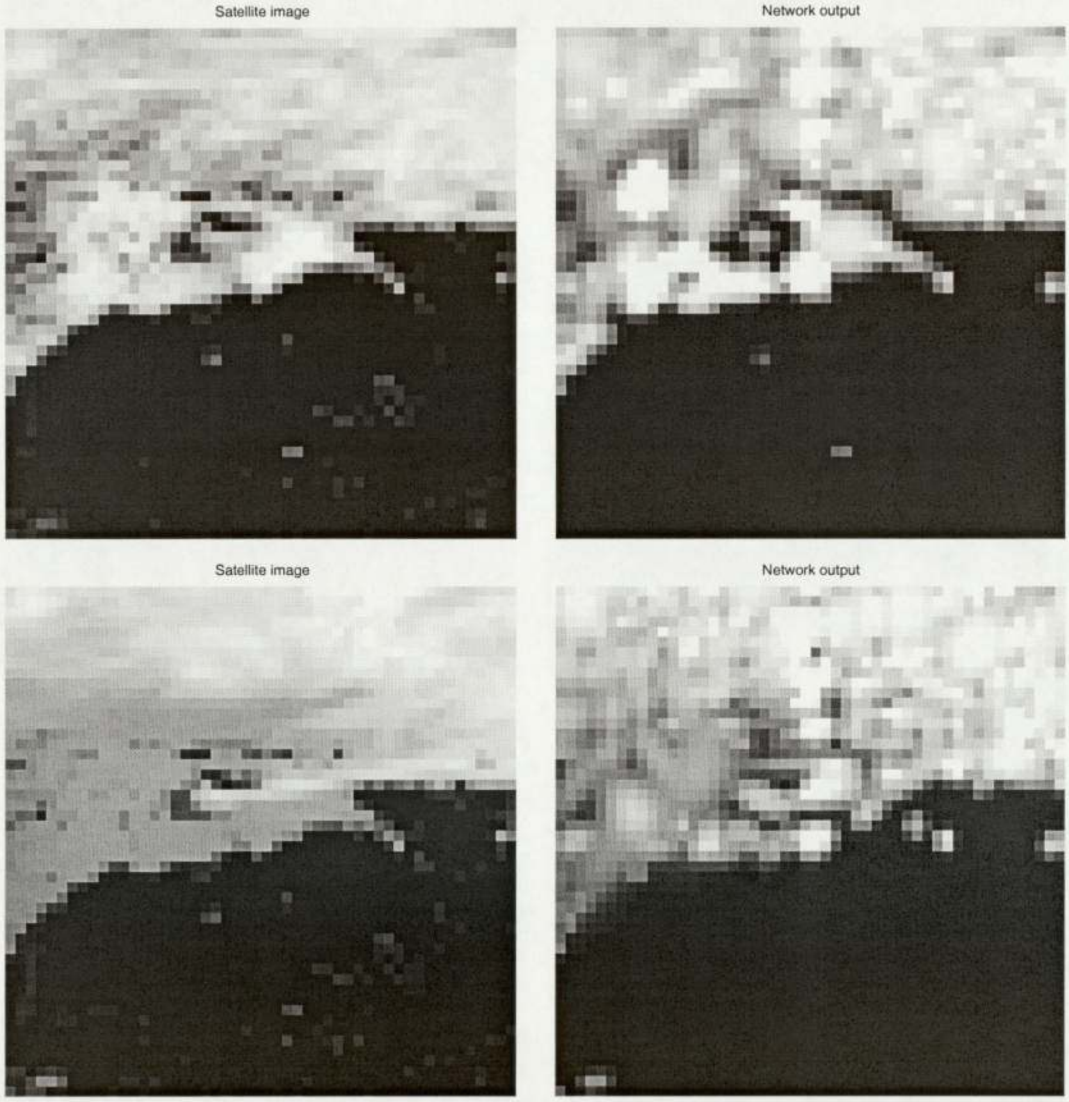

Figure C.1: Modelling of the visible and the infrared satellite images taken on February, 3rd 2003.

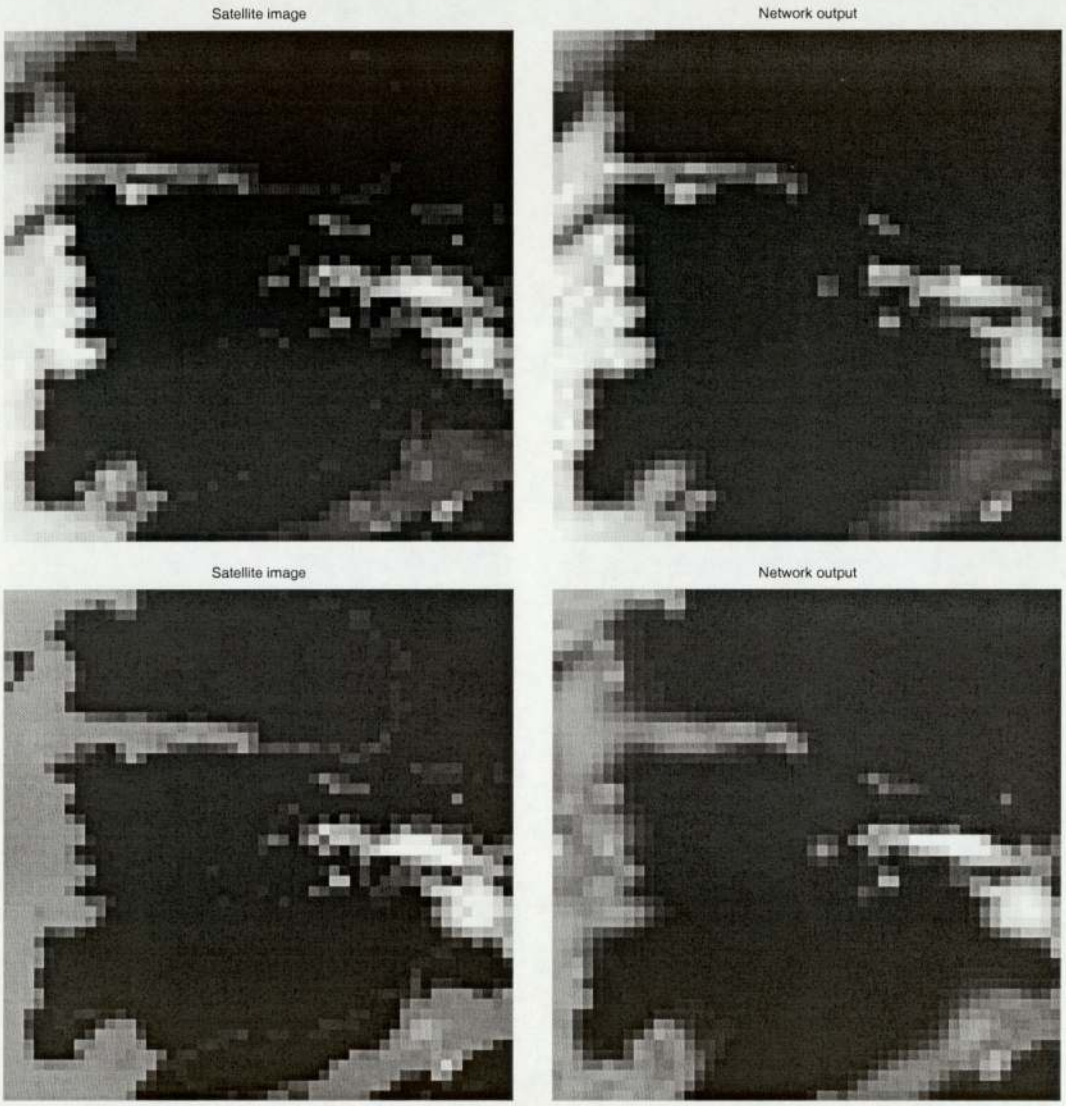

Figure C.2: Modelling of the visible and the infrared satellite images taken on July, 12th 2003.

Appendix D

More examples of the segmentation

Figure D.1: Segmentation of the basis functions based on the infrared satellite image taken on February, 3rd 2003.

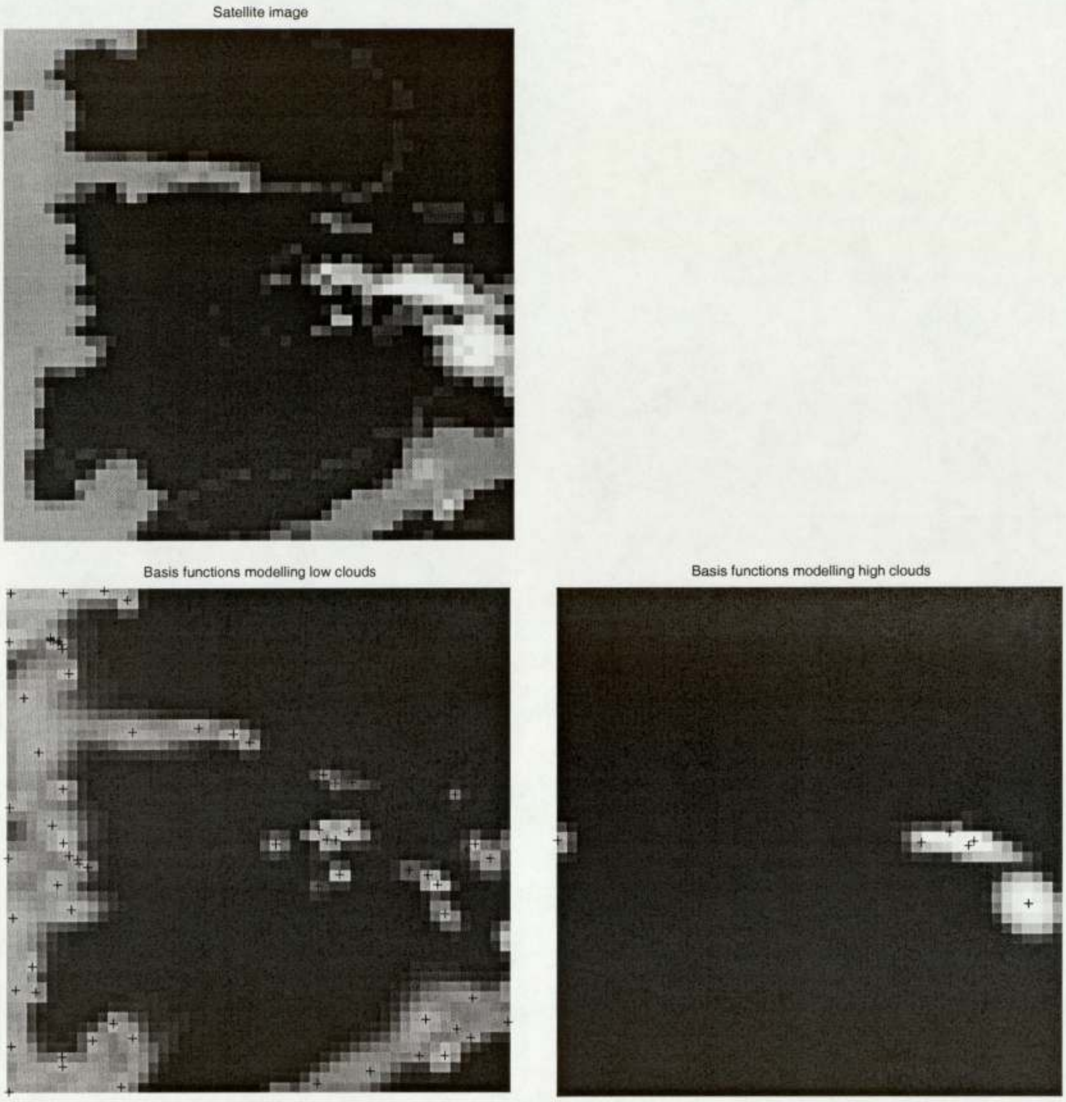

Figure D.2: Segmentation of the basis functions based on the infrared satellite image taken on July, 12th 2003.

Index

Bibliography

[1] R. K. Anderson and N. F. Veltishchev. The use of satellite pictures in weather analysis and forecasting. Technical Report WMO-No. 333, World Meteorological Organisation, 1973.

[2] B. A. Bailey, L. M. Berliner, W. Collins, and J. T. Kiehl. Modeling the Spatial and Temporal Distribution of Cloud Cover. *Technometrics*, September 1997.

[3] E. Batail. Statistical models for rainfalls: characterisation and forecasting. Master's thesis, Aston University, 2002.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[5] I. Cadez and P. Smyth. Modeling of Inhomogeneous Markov Random Fields with Applications to Cloud Screening. Technical Report No. 98-21, University of California, August 1998.

[6] D. Cornford. A Bayesian state space modelling approach to probabilistic quantative precipitation forecasting, November 2002. Aston University.

[7] Microsoft Encarta, 2002.

[8] K.-G. Karlsson and E. Lijlas. The SMHI model for cloud and precipitation analysis from multispectral AVHRR data. Technical report, Swedish Meteorological and Hydrological Institute, August 1990.

[9] V. Lakshmanan, R. Rabin, and V. DeBrunner. Hierarchical Texture Segmentation of Weather Radar and Satellite Images. *IEEE Transactions on Geosciences and Remote Sensing*, 2003.

[10] I. T. Nabney. *Netlab: Algorithms for Pattern Recognition.* Springer, 2002.

[11] M. Peura, A. Visa, and P. Kostamo. A New Approach to Land-Based Cloud Classification. *ICPR'96, 13th International Conference on Pattern Recognition*, August 1996.

[12] D. W. Reynolds and T. H. V. Haar. A bi-spectral method for cloud parameter determination. *Monthly Weather Review*, 105:446–457, 1977.

[13] R. W. Saunders and K. T. Kriebel. An improved method for detecting clear sky and cloudy radiances from AVHRR data. *International Journal of Remote Sensing*, 9:123–150, January 1988.

[14] M. Tipping. Statistical Pattern Analysis: Lecture Notes, February 1998.

[15] A. Visa, J. Iivarinen, K. Valkealahti, and O. Simula. Neural Network Based Cloud Classifier. In *Proc. International Conference on Artificial Neural Networks (ICANN'95), Industrial Session 14 (Remote Sensing)*, 1995.

[16] The importance of clouds and aerosols to climate change. http://earthobservatory.nasa.gov/Library/CALIPSO/CALIPSO2.html, 2002.