Real World Image Analysis

Benoit Maroillez

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

September 2003

ASTON UNIVERSITY

Real World Image Analysis

BENOIT MAROILLEZ

MSc by Research in Pattern Analysis and Neural Networks, 2003

Thesis Summary

This thesis considers some of the image processing problems in trying to construct an automated system for detecting near-shore water-borne pollution (oil slicks) using land mounted visible band cameras. In particular, we develop a novel approach to the uniform scene illumination problem to retrieve reflectance more accurately, prior to segmentation. We also introduce a simple Kalman filter approach to exploit some of the dynamic information content across images through time to improve the slick segmentation through a probabilistic model.

Keywords: Illumination removal, Homomorphic filtering, Radial Basic Functions, Image segmentation, Gaussian mixture models, Kalman filter.

Acknowledgements

First, I would like to thank my supervisor, Prof David Lowe, for his general attention and advice during this project. Secondly, I am grateful to Dr Dan Cornford for his availability and his help on the Kalman filter implementation, and to Dr I. Nabney for his many tips to optimise my Matlab code.

I am generally grateful to all my Aston lecturers, Dr M. Opper, Prof D. Saad, Dr I. Stainvas, Dr L. Csato and Dr P. Tino for the quality of their lectures. Many thanks also to Vicky Bond and Alex Brulo.

I would also like to thank all the fellow students from the Neural Computing Research Group for their constant support and friendly presence through all this year and especially the MSc students in PANN, Benoit, Frédéric, Hiroko, Jody, Rémi, Vagelis, and the PhD students Boremi and Stephane.

I also want to thank my friends from Aston, Amos, Anthony, Anne, Becky, Charlotte, Didi, Glaf, Hannah, Luca, Mendy, Sarah, and my flatmates Ahad, Collin, Franck, Matthew, Shy and Us, with whom I've spent a wonderful year.

To finish with my acknowledgements, I would like to thank my parents and my family for their love and financial support, especially my sisters Amélie and Armelle who made the effort to visit me in March. Many thanks to Elodie for her love support.
In memory of my beloved grand mother Suzanne.

Contents

List of Figures

List of Tables

Chapter 1

The Aims of the Project

1.1 Introduction

This project derives from an earlier European Union-funded project called Blue Water. Blue Water [1] was an attempt to produce a cheap and semi-automatic shore based video monitoring system for the early detection of marine surface oil slicks. The aim is to reduce the need for expensive blind spot sampling. Because satellite platform sensors have inadequate resolution($> 1km^2$) for the small scale slicks, the idea is to use land-mounted visible band cameras to track these surfaces in the pictures taken. In image segmentation and tracking problems, most demonstrations are based on tracking people or solid objects, such as cars, across relatively well-lit and relatively stationary backgrounds. With this problem of automated near-shore water-borne pollution monitoring using land-mounted cameras, we are interested in the problem of how to extract quasi-stable sea surface features such as slick deposits from oil discharges or agitated fish shoals or other marine biomaterial. The presence of an oil slick on the surface of the water tends to produce a damping effect of capillary waves leaving a visible signature which is diffuse but stable. The effect of this damping is to alter the local texture leading to regions of more highly reflective surface areas, compared to non-slick sea surfaces. Due to the typical camera angles, the reflective areas tend to mirror the sky, so leading to areas which appear brighter in intensity than normal sea state activity.

The aim of our project is to identify, locate, segment and track this bright object (the oil slick) on the near-shore pictures.

1.2 Data and Camera Location Characteristics

1.2.1 Different Sites

The data used in this project was provided by the Blue Water project. Blue Water camera systems are composed of a digital camera controlled by a server. Camera systems have been installed in different sites, in Cyprus at Yermasoyia (figure1.1) and on the Italian coast, in Sardinia (figure1.2) and River Po.

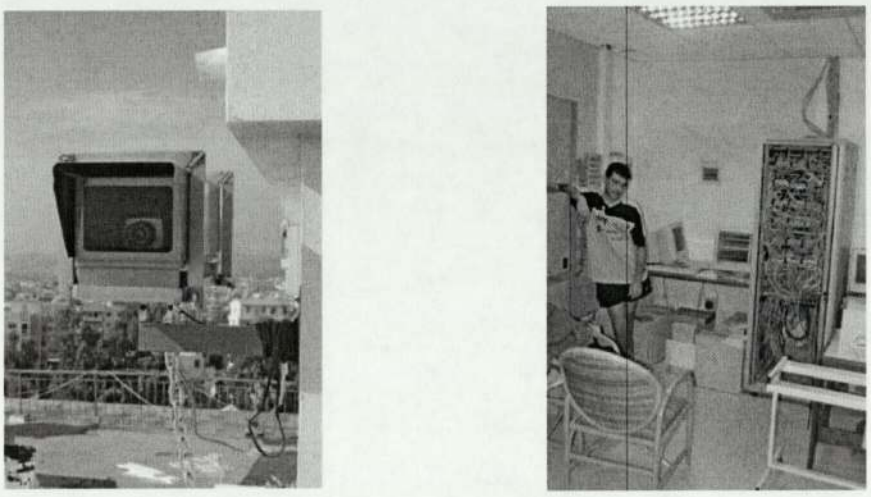


Figure 1.1: BW system (camera and control room installed in Yermasoyia (Cyprus)).

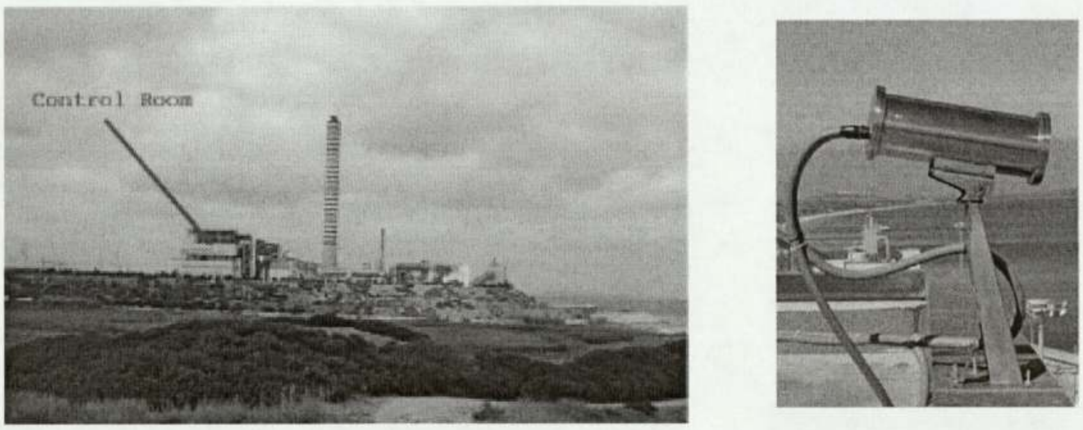


Figure 1.2: BW system (camera and control room) installed in Sardinia.

Cameras have been installed at a height to cover an area of a few hundred square kilometres. A lot of adjustments were made in the Blue Water projects to ensure that the cameras were capturing significantly the prints of pollution.

1.2.2 Data Selection

The data was provided on 13 CDROM's of digital pictures taken from the systems. 5 CDROM's are focused on the site of La Casella on the River Po in Italy, the 8 others contain digital pictures taken from a tourist resort in Yermasoyia (Cyprus). Specifications of the digital images are part of the Blue Water project. The standard image size has been set to 576 x 768 with a resolution of 8 bit/pixel. Most of the images available are in greyscale (year 2001-2002). All of the methods in this thesis were applied to greyscale images only (using one channel). However, some images have been gathered in colour (Yermasoyia November 2001).

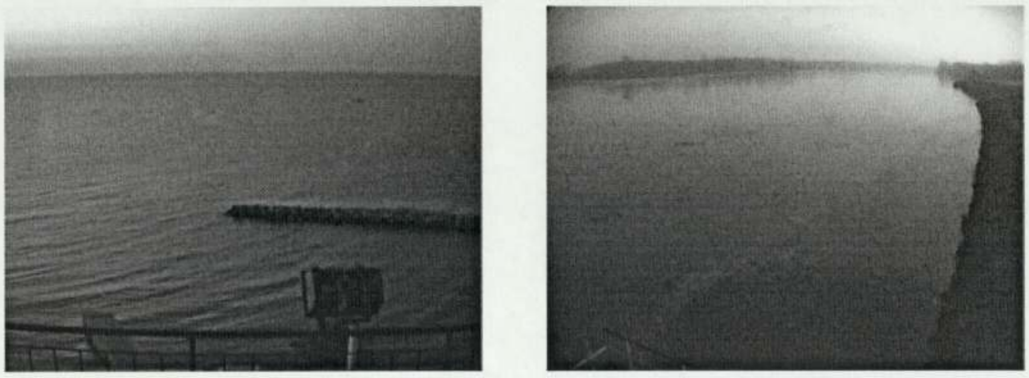


Figure 1.3: Pictures produced from the BW system on the Cyprus coast (left) and on the River Po (right).

After observing different samples from both sites, we noted different and specific problems for each of them. The data used in this project was specifically taken from the Yermasoyia site.
Eventually, we want to use temporal information to build a simple model capable of improving the detection of slicks. The Yermasoyia BW system provides us with a picture every 10 minutes comparing to the La Casella site which provides us with one every 30 minutes. Moreover, because of the angle chosen for the digital camera in La Casella, only a little part of a picture can be objectively used to detect oil slicks. The rest of the picture has to be eliminated because of strong reflecting effects of the shore or incident angles being too small. (The difference of contrast due to slicks is unobservable). If we consider that part of the picture, which represents an area close to the shore (a few hundred square meters), the time rate obtained from the camera and the current of the River Po, monitoring the same pollution prints on two consecutive pictures seems unusable.
At the other site, pollution can be observed and tracked in an area representing half of the picture. Moreover, the low speed of the evolution of oil slicks moving across the ocean and the frame rate combined with the area covered (a few dozen square kilometres), would hopefully allow us to track the pollution across frames.
Once the site was selected, the setting was studied, as part of the preliminary work, in order to quantify pollution on the sea and quantify how fast a slick is moving.

1.2.3 Site Characteristics

The camera system setting in Cyprus was located on the Hotel Miramare. The camera was placed on the roof of the hotel, facing the sea area to monitor.
The idea was to maximise the area that the camera can monitor. The optimal place

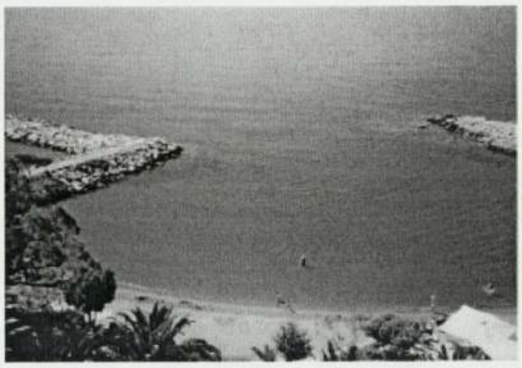


Figure 1.4: Hotel Miramare in Cyprus and the view from its roof.

chosen (the roof) was around 25 metres above sea level, giving a good view of the sea and a good place to locate the camera (the site is a tourist resort where the camera has to be discrete (big and not really nice looking object)).
The camera angle was set so as to capture significant sea area, but not too steep since this would image the sea bed instead of the sea surface. Figure 1.5 shows the camera orientation employed.

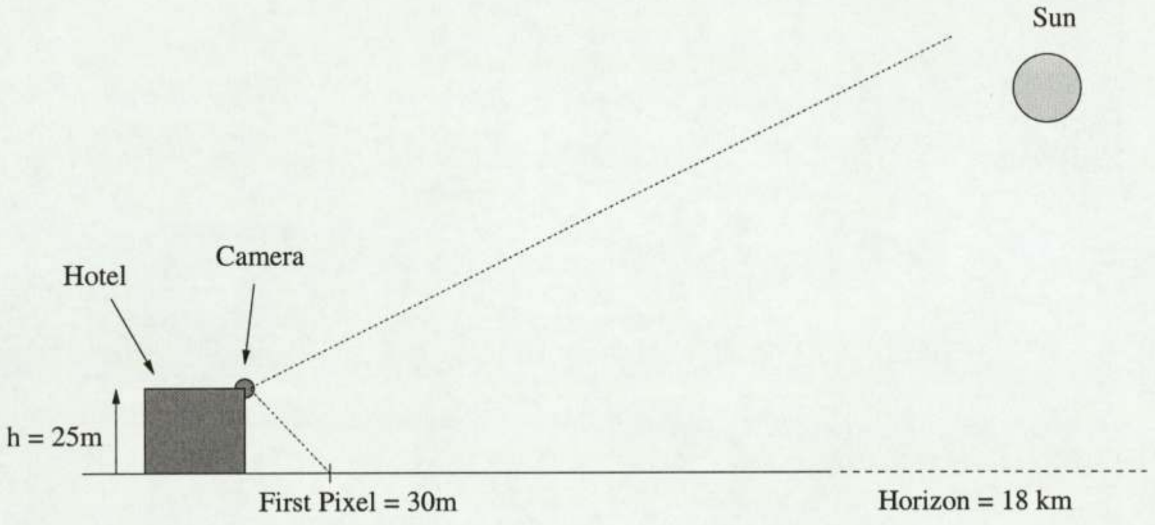


Figure 1.5: Scheme of the camera setting.

The camera location and orientation introduces geometry and intensity effects in the pictures.
Indeed, our image space reflects a transformation of the real dimension space. In the image, it is obvious that a pixel in the background is representing more space than a pixel in the foreground. It is important to identify the transformation (see figure 1.6) of the pictures and sort out what is the real area represented by each pixel. One difficulty is a lack of calibration information regarding the camera field-of-view aspect ratio and pixel sizes. When a picture has to be shifted back to real dimensions, some extra information about the camera lens (the focal length for example) are needed. The camera lens introduces distortion effects on pictures, such as the horizon not appearing as a straight line. Even if this distortion could not be removed exactly, we used some

general knowledge about distance to horizon and pixel sizes of appearing objects, such as boats which we can guess the real size, to approximate distances and area on the pictures.

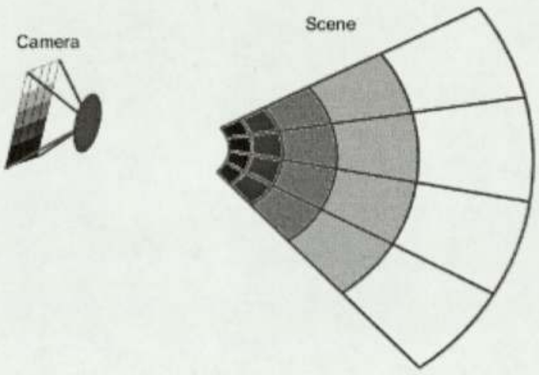


Figure 1.6: Distortion between image space and real space.

The approximation from pixel space to real space is explained in section 4.2 where it is used in the time model setting.

1.3 The Environmental Noise

After the preliminary work on the site and the approximation from pixel space to real space, an analysis was carried out to establish what the pictures contained. The object of interest appears mainly as a bright surface on the sea. Oil slicks appear very rarely as dark objects and such phenomena occur only with particular weather conditions such as a very dark sky or very cloudy sky. Dark areas within the image of the sea are probably the result of wakes created by human activity (such as the objects appearing in the left frame in Figure 1.7).
However, many of the bright objects that appeared on pictures have nothing to do with oil slicks. Therefore, it was important to look into the data we had and identify what kind of other bright objects are captured by the camera.
These objects should be classified as noise that we have to eliminate. The noise classification includes a study of its possible source and their characteristics.

1.3.1 The Sun and Weather Activity

Most of the bright objects are resulting from direct reflection of sun. The sun can create a very strong nonuniform illumination on the image. It can be reflecting or backscattering across the ocean.
If the sun is the source of light that allows slicks to shine on a sea surface, it is, at the same time, the main source that can annihilate or alter the visual detection of oil slicks.
Therefore, it is important that we find what kind of sun illumination the camera can capture, its different forms and the way they can evolve. By looking into the data from the same day, different shapes can be found depending mostly on how the weather

was. Indeed, weather (like strong wind) can change the shape of the sea surface deteriorating the observation of bright objects from a picture to another. For example, sun illumination can be diffuse in bright sunlight without wind and can turn into small bright objects when the wind creates waves (see figure 1.7).

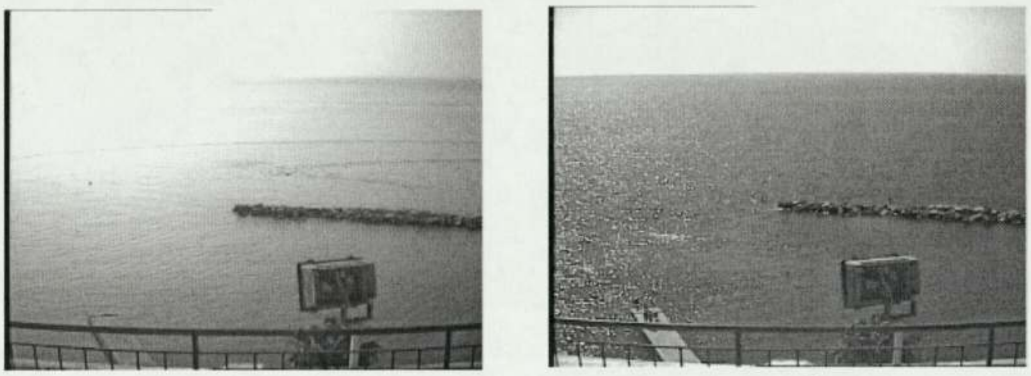


Figure 1.7: Weather can alter the profile of the sun illumination.

Beyond the different forms that sun illumination can take in the ocean, it is evolving according to time (cycles day/night), as it follows seasons. This evolution during a day is a slow process that rotates from low light, at early morning and afternoon, to glare, around mid day (see figures 1.7 and 1.8).

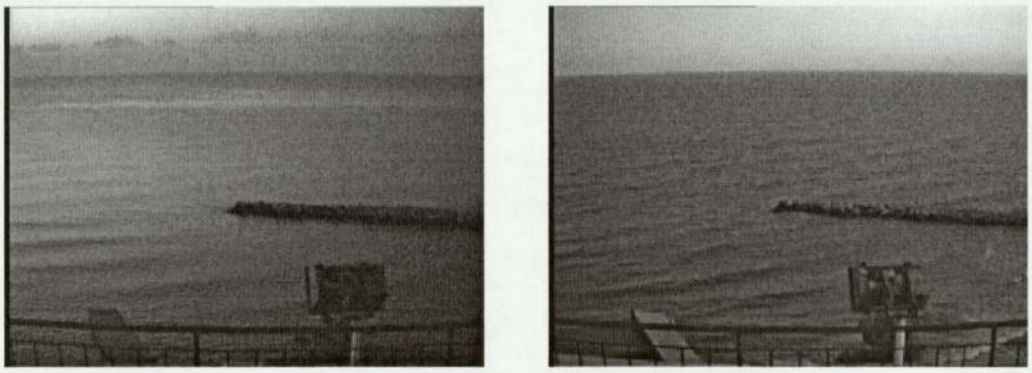
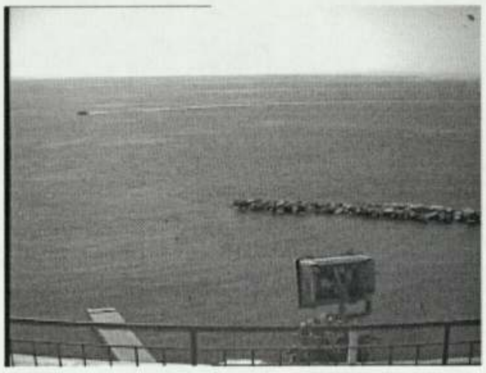


Figure 1.8: Examples of low light pictures.

In low light pictures, contrast tends to uniformity where brightness of any object is equalised.
Slicks can become hardly detectable.
In glare pictures (left picture in Figure 1.7), contrast is removed by the extreme illumination. Any object that should be hidden in the glare become undetectable because of the light saturation.

1.3.2 Human Activity

The second category of bright objects identified, that were not oil slicks, was the result of human activity. Indeed, the site selected is a tourist resort (Hotel Miramare). It is common to observe human activity such as people swimming, wind-surfing, water-skiing or yachting (see left picture of figure 1.9).


boat guiding a parachute

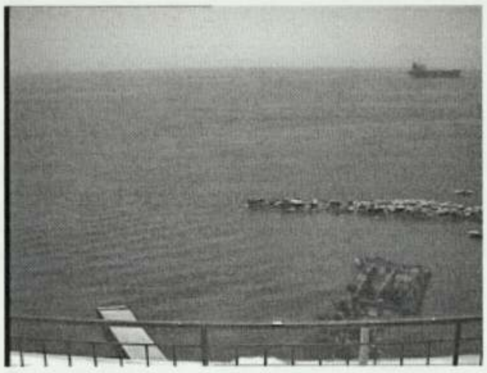

tanker crossing horizon

Figure 1.9: Human activity resulting from the tourist resort and ships travelling.

The other human activity detected on the pictures are tankers that cross the horizon leaving in their wake a lighter but stable print (see right picture of 1.9).

1.3.3 The Landscape and Fauna Activity

The rest of the bright objects that appear on pictures are part of the landscape. It is obvious that there is no need to look for oil slicks in these parts. They are composed of the sky and installations of the tourist resort (rocks and a sea path). All these surfaces should be ignored.
Because all these surfaces are stable across frames (the camera location and orientation are fixed), we decided to create a black and white template image to express the prior knowledge of being part of the background or not (see figure 1.10).

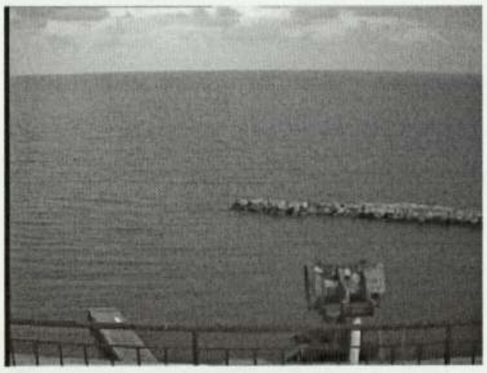


image of the site

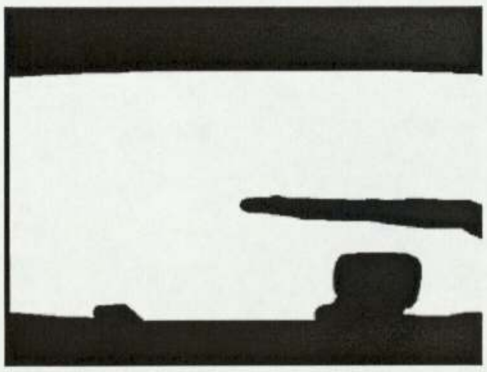

associated template image

Figure 1.10: Black and White template image created for Yermasoyia.

The black and white image will be processed through a Matlab file to retrieve a binary matrix (interesting area or not) with the size of a frame.
Moreover, it is important to note that rocks are known to host micro organisms that produce natural oil. This phenomenon due to fauna can appear close to the rocks and derives with waves or wind across the sea. This natural oil produces a similar damping effect as an oil slick, making it appear as a bright slick hardly differentiable from an oil slick.

1.4 Physics of Image Observations

Even if most of the data from Yermasoyia appeared as greyscale images, they are composed of the three classical colour channels (red, green, blue).
The perception of light and colour comes from specific electro-magnetic radiation. Their amplitude radiation which is a function of the wavelength λ, should be in a particular range $(380 - 780\, nm)$. Cameras are built to be sensitive to the same dominant wavelengths discretised to the three classical channels (red, blue green).
Concerning pictures, we are interested in retrieving the intrinsic properties of the objects present on a picture. Following the work of H.G. Barrow and J.M. Tenenbaum [2], the central problem in recovering intrinsic scene characteristics is that the information is confounded in the original light-intensity image: a single intensity value encodes all the characteristics of the corresponding scene point. Recovery depends on exploiting constraints, derived from assumptions about the nature of the scene and the physics of the imaging process. A classical model presented in image processing books for image intensity [5] is:

$$f(x,y) = i(x,y) \times r(x,y) \tag{1.1}$$

We will based our approach to suppress illumination effects, $i(x,y)$, on that common and simple model.
Although equation (1.1) appears to provide a simple model, note that from the observation $f(x,y)$, we really need the reflectance characteristics $r(x,y)$. However, we cannot extract $r(x,y)$ since it is an ill-posed problem: for each pixel we have twice as many unknowns as equations. We need to find a technique that could compensate for this lack in data knowledge.

1.5 The Thesis Plan

After this introduction presenting the issues that appeared in the monitoring of near-shore images, the following chapters will present methods to solve some of these issues.

In chapter 2, the problem of removing the illumination characteristics $i(x,y)$ from the observations $f(x,y)$ will be considered. We will present a novel approach to retrieve the reflectance $r(x,y)$.

The subsequent chapter 3 will consider the segmentation of $r(x,y)$ to extract possible slick regions. We will build a model which fits to the data and creates a method to identify and locate the oil slicks.

In chapter 4, we will consider that equation (1.1) also hides multiple time scale effects. In particular, the assumption that slicks evolve on a slow time scale compared to background sea-states will allow the introduction of dynamic models to improve the initial probabilistic segmentation.

The thesis plan can be summarised by the following workflow:

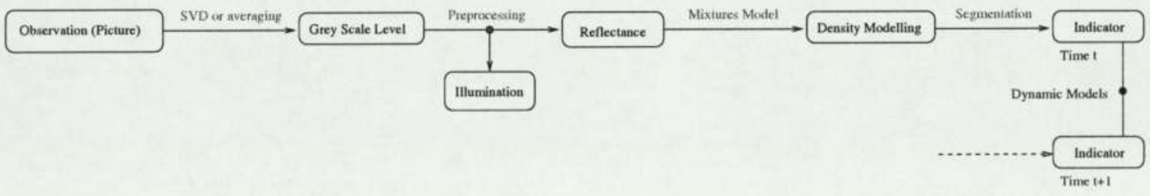


Figure 1.11: The Global Process.

Chapter 2

Image Preprocessing

This chapter will present the work achieved on preprocessing of the image.The main problem we are concerned with is to compensate for the nonuniform illumination effects. A classical filter in image processing to remove illumination from the image, and retrieve reflectance [12], is the homomorphic filter. We will explain the theory behind this filtering, its implementation, the results obtained, the advantages and the drawbacks that can lead us to look for a new preprocessing model.
This new model [7], consisting of an RBF network fitting intensity, will be described later. As for the homomorphic filter, we will explain the theory behind this model, its implementation and the results obtained .
We will conclude with a comparison of the two models used for achieving the preprocessing. In the intensity model used (see equation (1.1)), we have to invert a product which is not a trivial problem in this case due to the lack of informations (twice as many unknowns as equations, see section 1.4).
An idea can be to separate the two factors i and r by taking the logarithm of equation (1.1) so that the effects are additive rather than multiplicative:

$$g(x,y) = ln\ f(x,y) = ln\ i(x,y) + ln\ r(x,y) \tag{2.1}$$

The preprocessing will take place in the log domain in which an additive effect is simpler to remove.
In order to get observable results, images can be rescaled or thresholded using secondary filters whose aim is only to enhance the quality of the reflectance image retrieved.
Rescaling is made using the equation for linear rescaling (2.2):

$$f_{new}(x,y) = Floor[\frac{f_{old}(x,y) - min_{x,y}(f_{old}(x,y))}{max_{x,y}(f_{old}(x,y)) - min_{x,y}(f_{old}(x,y))} * 255] \tag{2.2}$$

Thresholding is made by fitting a single Gaussian of the intensity histograms. A new lower bound for intensity is determined as the lower intensity value obtained for 5% of the Gaussian maximum. A new upper bound for intensity is determined as the higher intensity value obtained for 5% of the Gaussian maximum.

2.1 The Homomorphic Filter

2.1.1 Theory

The first method used for preprocessing the image was based on homomorphic filtering. Indeed, illumination through an image has special properties dealing with frequencies. Therefore, instead of using spatial filters, that could fail depending on how the illumination source is oriented, we have used a homomorphic filter, specifically focused on the characteristic of the object we want to suppress (illumination).
Illumination is generally of a smooth nature and yields low-frequency components in the Fourier transform of the image. Different materials (objects) on the other hand, imaged next to each other, cause sharp changes of the reflectance function, which cause sharp transitions in the intensity of an image. These sharp changes are associated with high-frequency components.
From equation (2.1), we transform our image to the spectral domain taking the Fourier transform:

$$\mathcal{F}\{g(x,y)\} = \mathcal{F}\{ln[i(x,y)]\} + \mathcal{F}\{ln[r(x,y)]\} \tag{2.3}$$

because the Fourier transform is linear we can write (2.3) into:

$$G(w_x, w_y) = I(w_x, w_y) + R(w_x, w_y) \tag{2.4}$$

where $I(w_x, w_y)$ is the Fourier transform of $ln\ i(x,y)$ and $R(w_x, w_y)$ is the Fourier transform of $ln\ r(x,y)$. Then, it is possible to apply a filter, with a transfer function named $H(w_x, w_y)$ to the log Fourier transorm of the intensity $G(w_x, w_y)$:

$$\begin{aligned} S(w_x, w_y) &= H(w_x, w_y) \times G(w_x, w_y) \\ &= H(w_x, w_y) \times I(w_x, w_y) + H(w_x, w_y) \times R(w_x, w_y) \end{aligned} \tag{2.5}$$

Using the linearity of the inverse Fourier transform, we get:

$$\begin{aligned} s(x,y) &= \mathcal{F}^{-1}(S(w_x, w_y)) \\ &= \mathcal{F}^{-1}(H(w_x, w_y) \times I(w_x, w_y)) + \mathcal{F}^{-1}(H(w_x, w_y) \times R(w_x, w_y)) \\ &= i'(x,y) + r'(x,y) \end{aligned} \tag{2.6}$$

We then exponentiate $s(x,y)$ to get the enhanced image:

$$\begin{aligned} s'(x,y) &= exp[s(x,y)] \\ &= exp[i'(x,y)] \times exp[r'(x,y)] \\ &= i''(x,y) \times r''(x,y) \end{aligned} \tag{2.7}$$

Now $i''(x,y)$ and $r''(x,y)$ are the illumination and reflectance of the "enhanced" image. However, using the spectral properties of illumination and reflectance, we can choose an appropriate filter so that the "enhanced" image represents only reflectance, assuming that the spectral content of reflectance and illumination is disjoint.
In the Log-Fourier domain, we have to use a high-pass filter that will suppress the low frequencies. Variations of the illumination should then be reduced while edges (and details) should be sharpened. The cross section of a homomorphic filter, see figure 2.1, presents the spectral property of the desired filter.

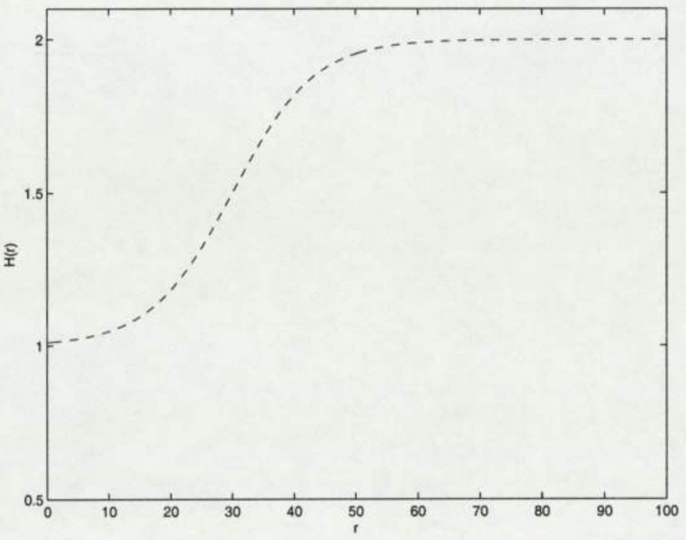


Figure 2.1: A cross-section of a homomorphic filter as a function of polar frequency $r \equiv \sqrt(w_x^2 + w_y^2)$.

Such a filter can be expressed by the following transfer function:

$$H(w_x, w_y) = \frac{1}{1 + e^{-s(\sqrt{w_x^2+w_y^2}-w_0)}} + A \qquad (2.8)$$

where s is the slope, w_0 the cutting frequency and A the amplification coefficient. We summarise our approach in figure 2.2.

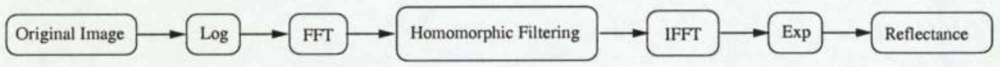


Figure 2.2: Homomorphic Filtering Block Diagram.

2.1.2 Implementation and Results

The implementation of the homomorphic filtering, for preprocessing the image (Figure 2.3), was done in Matlab.
To shift the image in the Fourier domain, where the homomorphic filter needs to be applied, we used the two-dimensional discrete Fourier transform already implemented in Matlab (function `fft2`). However, we must ensure that our Fourier transform is well centred so that our homomorphic filter will cut the right frequencies. Indeed, the zero-frequency component should be in the centre of the spectrum so that the high-pass filter is effective. Centring was achieved using Matlab function `fftshift`. Once the image has been centred in the spectral domain, we can process it with the homomorphic filter.
The main problem faced was the setting up of the coefficients of the filter. Each of the three coefficients controls a different part of the filter. The cutting frequency w_0 should reflect the boundary, in the spectral domain, that separates details (high frequency)

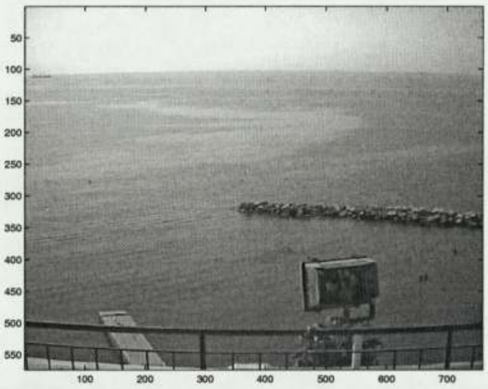


Figure 2.3: Original image to be preprocessed with the homomorphic filter.

from the illumination structure (smooth nature). Our idea, to figure out a good value for w_0, was to compute the spectrum of the image and try to identify both clusters (Figure 2.4).

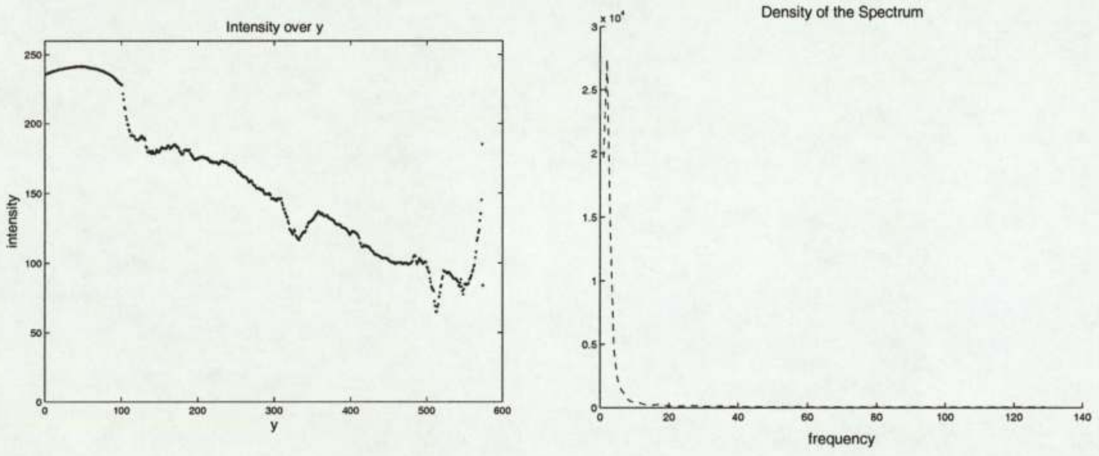
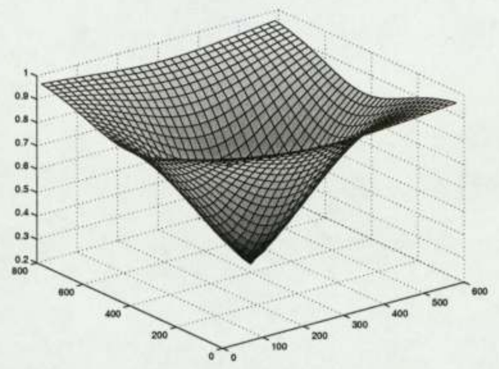


Figure 2.4: Horizontal Intensity and Spectrum Density.

The spectrum density presents a peak in the low-density region that we believe to be the illumination structure of the picture. The rest of the density spectrum is diffuse and associated with low coefficients (details have very different structure like wave/boat leading to a spread of frequency). Using the spectrum, we set the cutting frequency to 20.
The slope s and the amplification factor A are more difficult to set. The slope controls the smoothness of the way we cut around w_0 while the amplification factor is strengthening the effects of the filter (only the uncut frequencies are amplified).
If we consider a 3D space where the base is made of the 2D spectral domain (w_x,w_y), the setting of s can turn the homomorphic filter from a cone to a cylinder (Figure 2.5).

If the illumination is assumed to be clearly separated in the spectral domain from the reflectance structure, the filter should be set as a cylinder. The slope was set to 5 and the amplification factor to 0.


A cone with $s = 0.01$

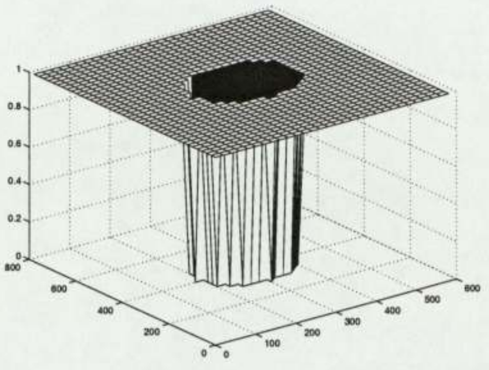

A cylinder with $s = 2$

Figure 2.5: Control of s over the smoothness of the filter.

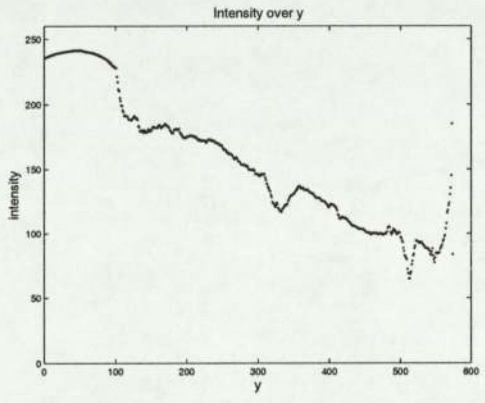


Figure 2.6: Before filtering.

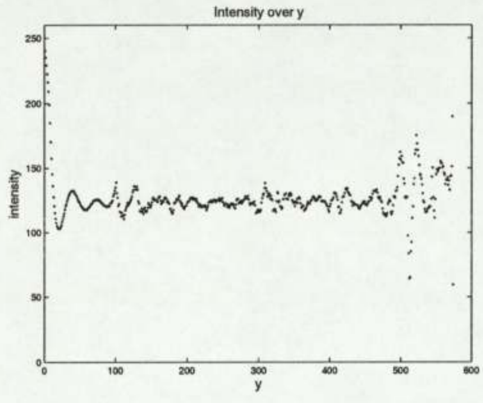
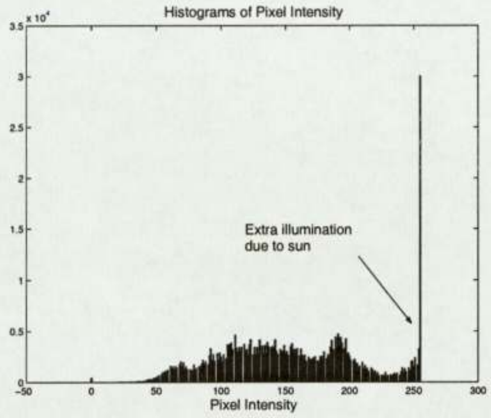


Figure 2.7: After filtering.

As, we can see in figures 2.6 and 2.7, the general decreasing shape of horizontal intensity (illumination is coming from horizon) has been flattened. Moreover, secondary peaks that represent objects in the pictures are still present with a similar amplitude. The levelling of the new curve for intensity (Figure 2.7) is a hint that global illumination has been removed from the picture.
This hint is confirmed by the histograms of intensity in the pictures. Histograms of the original include a secondary and significant peak in the high values of intensity. This peak, that is in the range $220 - 240$, should represent the bright sunshine illumination. In the filtered image, this peak has been removed (Figure 2.8).
At the same scale, the resulting histograms are less spread on the range $0 - 250$. Therefore, the resulting image is darker (Figure 2.9).
Some rescaling and thresholding was done to ensure there was no important loss in the dynamic range.

2.1.3 Advantages and Drawbacks

The principal advantage of the preprocessing homomorphic filter comes from its basis. Homomorphic filtering is based on intrinsic spectral properties of the two components of an image: illumination and reflectance. In theory, the spectral content should be



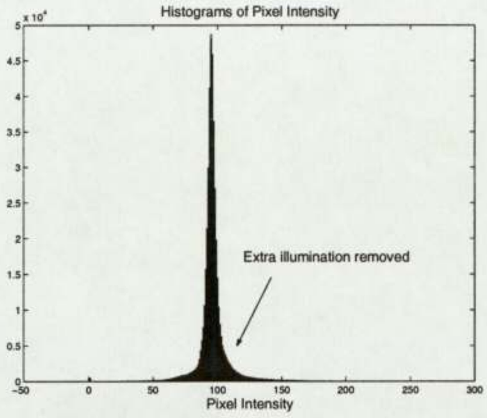


Figure 2.8: Histograms of the intensity of the original image and the filtered image.

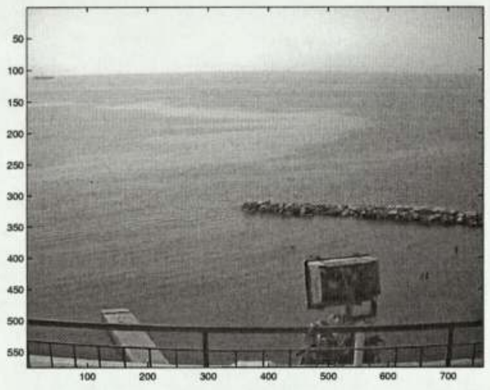
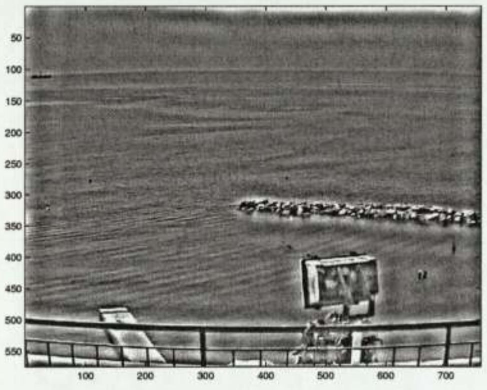



Figure 2.9: Image before and after filtering.

disjoint. In such conditions, an homomorphic filter should allow us to remove entirely the illumination without removing details that compose reflectance. Another advantage of the homomorphic filter is its simple implementation and its low computational cost. Most of the process time is due to Fourier transform and inverse Fourier transform. The use of the fast Fourier transform (functions `fft2` and `ifft2`) appeared as a good solution to reduce this time.

However, the filtered image can have strong edge effects due to the characteristics of the homomorphic filter (Figure 2.9). This is mainly due to the fact that on real world images the spectral content is not clearly disjoint. The peak identified for the illumination component hides smooth components from the reflectance.

Moreover, when applying the filter to an image, the quality of the filtering is variable. For different conditions of illumination, the homomorphic filter gives variable results. In real world images, and in our specific case of sea coast monitoring, these conditions can change in a short time range (see section 1.3.1). The setting of the homomorphic filter is the key to a good process. That means that for every single frame that we have to process, we need to tune properly the filter, finding the optimal parameter s, w_0 and A. If w_0 can be retrieved from the spectrum, s and A are tuned empirically.

Therefore, the homomorphic filter can hardly become adaptive. With the amount of data that needs to be processed and the diversity of their illumination conditions, a new adaptive filter is needed.

2.2 The RBF Illumination Filter

2.2.1 Theory

Radial Basis Function (RBF) networks [14] are related to kernel methods for density estimation and regression and to normal mixture models. The idea of an RBF model is to expand a given function f using a set of basis function of the form $\Phi(\| x - x_j \|)$, where Φ is a non-linear function to be chosen. The output is then taken to be a linear combination of the basis functions:

$$f(x) = \sum_j w_j \Phi_j(\| x - x_j \|) + w_0 \tag{2.9}$$

Where w_j is the weight given to the j^{th} basis function and w_0 is the bias. Several forms of basis function can be used, the most commonly used are the Gaussian and the thin-plate spline. The Gaussian basis function is:

$$\Phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) , \tag{2.10}$$

where σ controls the smoothness properties of the interpolating function. The Gaussian is a localised basis function with the property that $\lim_{x \to \infty} \Phi(x) = 0$.
The thin-plate spline basis function is

$$\Phi(x) = x^2 \ln(x) \tag{2.11}$$

A radial basis function network uses several RBFs as hidden units (See Figure 2.10). The number M of basis functions needs to be less than the number N of data points,

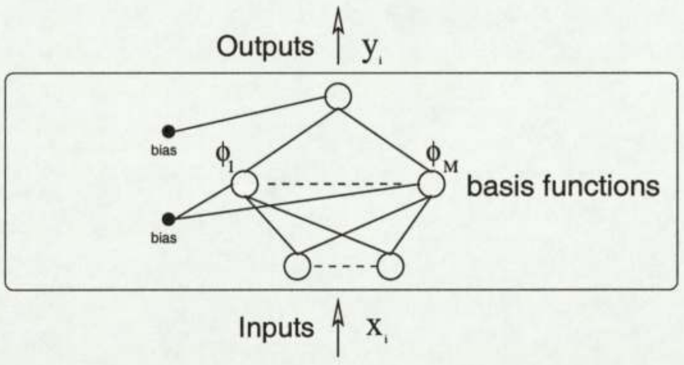


Figure 2.10: RBF network.

each basis function has its own width σ_j. The interpolation formula (2.9) is then:

$$y_k(x) = \sum_{j=1}^{M} w_{kj} \Phi_j(x) + w_{k0} . \tag{2.12}$$

The Gaussian basis function can be expressed by:

$$\Phi_j(x) = \exp\left(-\frac{\| x - \mu_j \|^2}{2\sigma_j^2}\right) \tag{2.13}$$

and the thin-plate spline basis function by:

$$\Phi_j(x) = \| x - \mu_j \|^2 \ln(\| x - \mu_j \|) , \tag{2.14}$$

where x is the input vector and μ_j is the vector determining the centres of the basis function Φ_j. Once the basis function has been chosen, we have a simple model whose parameters can be found by a least squares procedure, or any other optimisation procedure.
For a large class of basis functions, RBF networks are universal approximators [14]. Besides, they possess the property of best approximation, which means that the set of functions corresponding to all possible choices of the adjustable parameters includes the optimal approximation. The advantage of this network family is that RBF models are very fast to train as the "first layer parameters" of the basis functions can be fixed and then trained as a quadratic optimisation problem. In contrast, networks that require non-linear optimisation, such as MLPs, are generally much slower to train.

2.2.2 Implementation and Results

The idea is to replace the homomorphic filtering by a RBF fitting of the picture.
RBF networks may help us to remove a smooth structure which would be assumed to be the illumination. To achieve this removal properly, RBF networks will be constructed in an unusual way.

Fixing the centres

First, the centres of the RBF structure will be positioned around the pictures with a fixed position (Figure 2.11). It is assumed that they will act as virtual sources that generate global illumination across the observed scene.
Netlab was used to create the RBF network and train it. Netlab is a neural network extension to Matlab developed by Ian T. Nabney[11]. A RBF network is created using the function `rbf` and trained using the function `rbftrain`. In our model, the RBF centres are fixed while their locations are optimised in the original Netlab code, using a few iterations of the k-means algorithm.

Choosing the activation function

An important setting in RBF fitting is the choice of the activation function for the network. As explained in section 2.1.1, we suppose that illumination is from a smooth structure over the image. In terms of smoothing the image, there was no real reason to prefer one activation function from another. Both of the thin-plate spline and Gaussian are suitable for smooth fitting even if 3D plot are fundamentally different(Figure 2.12).

With a Gaussian activation function, the smoothness of the fitting will be directed

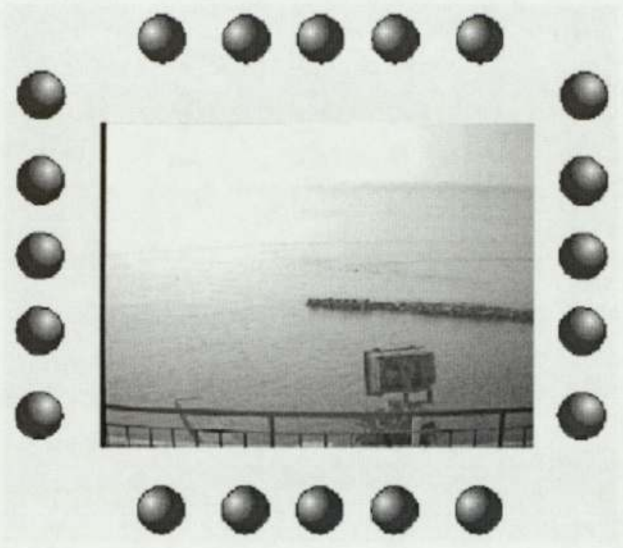


Figure 2.11: RBF centres as light sources.

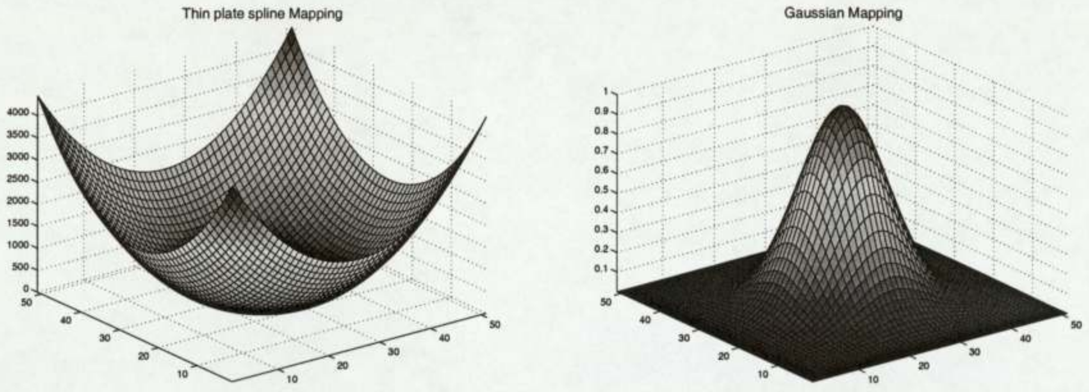


Figure 2.12: Comparing a 3D Gaussian mapping to the thin-plate spline mapping.

by the widths of the Gaussian. Indeed, the width parameter is used to set Gaussian variances. To ensure a good fitting, the existing `rbftrain` code uses the largest squared distance between centres.
In practice, both activation functions led to similar results. Both functions can be used to train the RBF network. Our choice for the final model ended with the Gaussian which appears to be slightly faster in the Matlab implementation (Table 2.1).

Gaussian	Thin-plate spline	Difference
34.1	40.9	-6.8

Table 2.1: Average execution time (in seconds) for each activation function realised on 20 different pictures.

Positioning the centres around the pictures

In our model, our centres are disposed around the pictures to act "intuitively" as light sources. The idea was to distribute equally the centres along the sides of a picture. That will allow us to model accurately illumination coming from any particular direction. The setting of initial position of the RBF centres was coded in the Matlab function `initiatecenter`. The number of hidden units (number of centres) was settled as a multiple of 4 (same number of light sources on each side).
One issue faced was the setting of the distance from a centre to the picture. This distance governs the widths of the Gaussian centres and can consequently act in the smoothness of the filter. Positioning a centre very far from the picture will result in increasing variance of the Gaussian (square of the maximum inter centres distance). Gaussians will then become very broad and result in a flat fitting of the illumination as an average intensity. The opposite effect, resulting from placing the centres on the edges of a pictures, could be the appearing curvature of sharp Gaussians on the illumination mapping. In practise, the width of each picture provides us with wide Gaussians even when the centres are positioned on the edges. In the experiments here, the centres have been positioned on the edges.

Choosing the number of hidden units

In a RBF network, the number of hidden units controls the complexity order of the retrieved function. In our model, the hidden units are the virtual light sources that have generated the illumination. Increasing the number of hidden units should allow us get a finer approximation of the illumination across the picture. However, increasing the number of hidden units can also dramatically raise the computational cost and the execution time. On the other hand, reducing it to a very small value will generate very simple models that will present high error bounds with the true illumination mapping (illumination is smooth but nonlinear).
The choice of the number of hidden units raises the problem of the quality of the illumination that we want to fit. Indeed, we have no prior knowledge about the smoothness of the illumination mapping. By increasing the number of hidden units, the RBF networks will start to fit details that belong to the reflectance part. By reducing it, the fitting will tend to a linear model that will be inaccurate. Moreover, in a broad range of value ($12 - 20$ hidden units), the illumination profile appears as smooth enough not to fit details significantly.
The number of hidden units was set empirically as a good compromise between not capturing the reflectance details and quality for the illumination retrieved. The current code is using 16 hidden units.

Training the network

The input to the RBF network is a 2 dimensional vector of the pixel space coordinates. The output is a vector of the corresponding intensity of the inputs. Instead of presenting to the network a full image, which corresponds to an input size of $576 * 768 * 2 = 884736$, we generally take a sub-grid of the image as the set of inputs (See left frame in Figure 2.13), decreasing the size of the training set in the order of a few thousands.

Results

The same code was applied to different pictures to test its generalisation.
In the first result presented here, the training data was selected on the entire picture. The ratio of selected pixels horizontally was set to 1 pixel every 25. The vertical ratio was set to 1 pixel every 15. The figure 2.13 presents the illumination structure retrieved (right) using this approach.

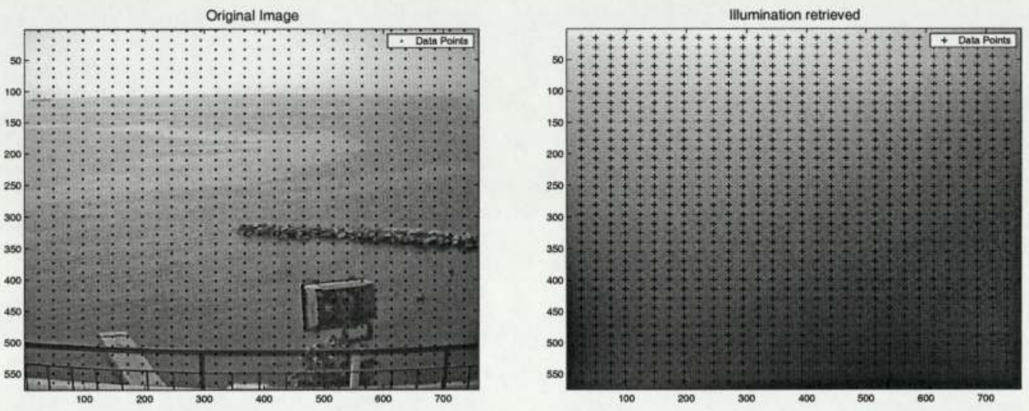


Figure 2.13: Retrieving illumination using the RBF model on the full image.

To measure the efficiency of the new model, we use the same criteria as the ones used for the homomorphic filter. The effect observed on the horizontal intensity is similar to the one observed previously. Indeed, in the figures 2.14 and 2.15, the general decreasing shape of horizontal intensity (illumination is coming from horizon) has been flattened. Moreover, secondary peaks that represent the objects in the pictures are still present with a similar amplitude. The new curve for intensity (Figure 2.15) is a hint that global illumination has been removed from the picture.

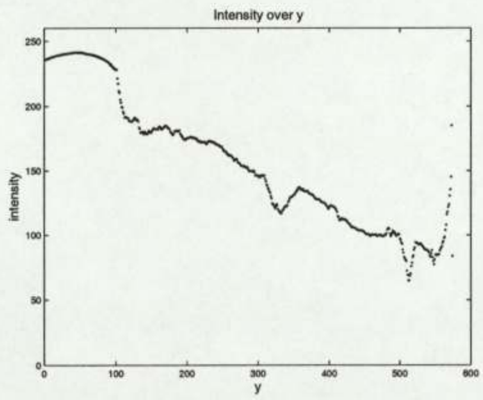


Figure 2.14: Horizontal intensity before filtering.

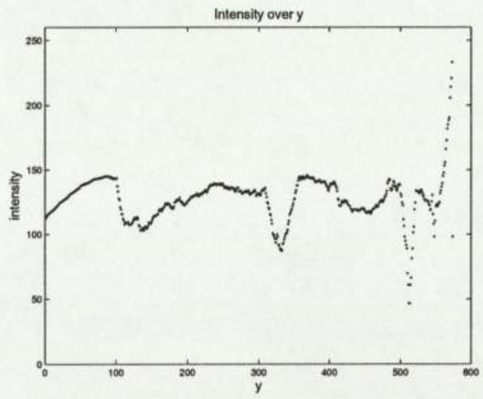


Figure 2.15: Horizontal intensity after filtering.

This hint is confirmed by the histograms of intensity in the pictures. Histograms of the original include a secondary and relevant peak in the high value of intensity. This peak, that is in the range 240 − 250, represents the bright sunshine illumination. In the filtered image, this peak has been removed (Figure 2.16).

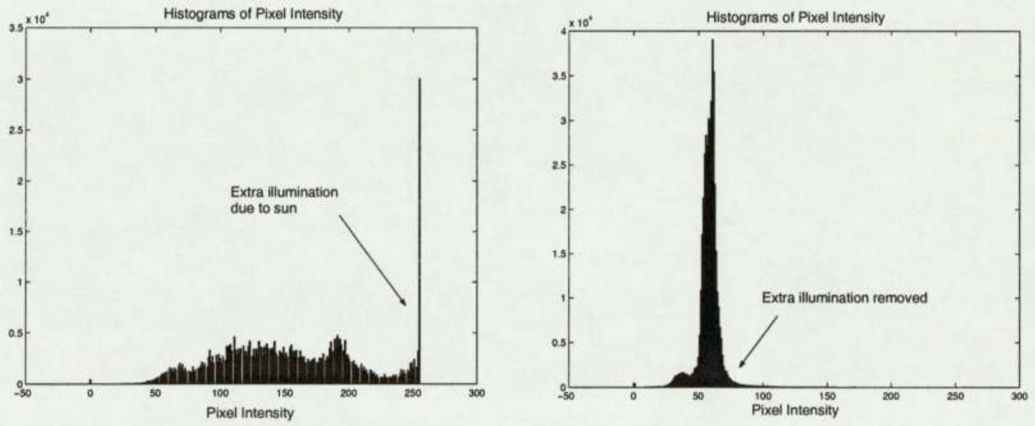


Figure 2.16: Comparing intensity histograms before and after filtering.

For the second and all subsequent images, the illumination was retrieved using a grid from the upper part of the picture to speed up the process (Figure 2.17).

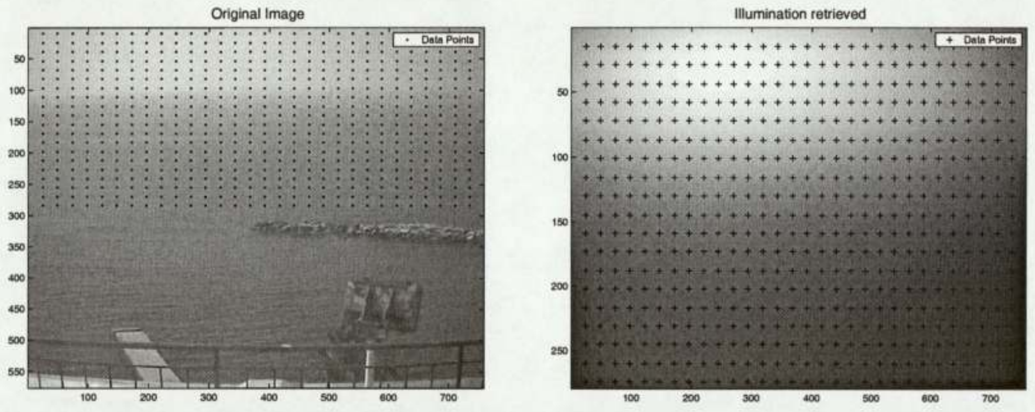


Figure 2.17: Retrieving illumination using the RBF model.

The comparison of horizontal intensity before and after filtering reveals the efficiency of our RBF model (Figures 2.18 and 2.19). This is confirmed by the intensity histograms presented in figure 2.20.

2.3 Comparison of the two models

The preprocessing of the image has been achieved using two different techniques. The basic technique, called homomorphic filtering, is based on spectral properties of the illumination. The low frequency part of the spectrum of the image is dominated by the illumination component. The homomorphic filter is a high-pass filter defined by three parameters s, w_0 and A. If w_0 can be approximated with an analysis of the spectrum, the optimal set of parameters requires hand-tuning. Therefore, there is no guarantee that the same filter will work across different times of day or seasons. The

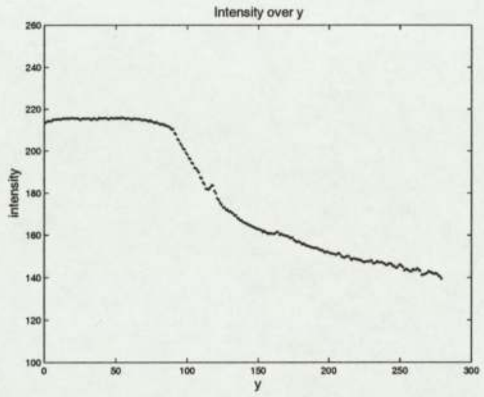


Figure 2.18: Horizontal intensity before filtering.

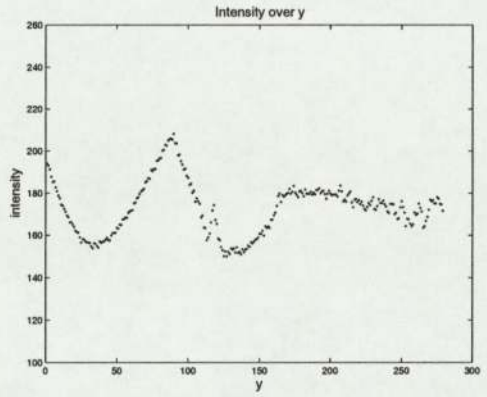
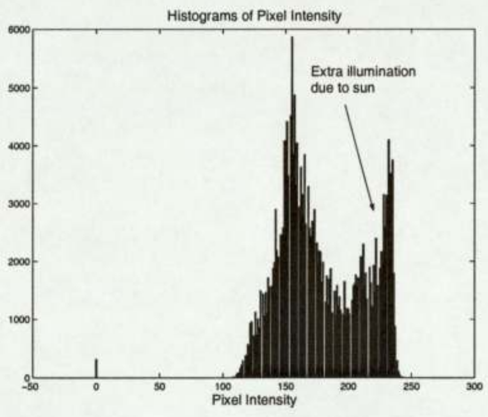


Figure 2.19: Horizontal intensity after filtering.



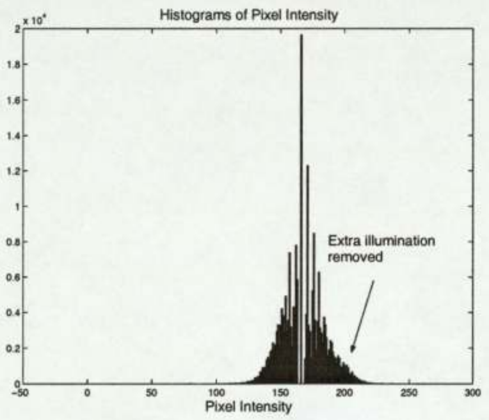


Figure 2.20: Comparing intensity histograms before and after filtering.

homomorphic filter is not adaptive
The second model was built following the need for an adaptive model. A RBF network is set with centres around the pictures and trained on the image intensity. The smooth fitting retrieved (mixture between number of hidden units and activation function) is taken for the illumination as the centres act as virtual light sources. The main advantage of this model to the previous one is its adaptiveness. Processing pictures from different times of day or season with the same filter still gives good results.
Concerning the quality of the reflectance image retrieved using both filters, a comparison seems hard to achieve. Indeed, the true illumination mapping is unknown and both reflectance approximations are fairly good. However, the use of the homomorphic filter can let edge effects (due to the spectral properties of the filter) appear on the picture. These undesired edge effects of the homomorphic filter have to be balanced with the unknown error made by fitting details as part of illumination in the RBF model (difficult choice of hidden units).
The final choice for the RBF model was driven by its high adaptiveness and high flexibility (quality simply directed by number of hidden units).

2.4 Summary

In this chapter we have discussed the problem of nonuniform illumination, which has to be compensated for prior to any detailed image segmentation. We have introduced and compared two methods, which rely mainly on the smooth profile of the illumination compared to the sharp one of the reflectance:

- an homomorphic filter, which is a high-pass filter applied in the spectral domain that solves the problem using assumptions on frequencies.
- a RBF filter which solves the problem by extracting a smooth structure from the intensity mapping.

Our results indicate that the RBF filter is more accurate (no edge effects) and more adaptive (the same filter gave good results on different frames) than the homomorphic filter.

Chapter 3

Image Segmentation

Once the reflectance image has been retrieved, the image segmentation for detecting slicks can start. The idea is to consider that a pixel can belong to one of a different set of populations. As discussed in the introduction, the object to be tracked is a bright and stable surface on the sea. In our case, one of the population will be labelled as "oil slicks". The task is to achieve a probabilistic classification of each pixel, i.e. approximate to which population group they belong to. Classification problems are usually solved using clustering algorithms. The aim of such algorithms is to fit a model to the data. The model is then used to establish a probability mapping that allows the classification of a pixel. The segmentation process is the use of this probability map to create and separate specific populations of pixels.

3.1 Density Modelling

A useful and popular class of models for clustering and density modelling are Mixture Models which are convex combinations of basic model components ([3],[9]).
Here, we consider models in which the density function is formed from a finite linear combination of basis functions. The model for the density will be written as:

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

where M is the number of basis functions, $p(\mathbf{x}|j)$ the component densities and $P(j)$ the mixing coefficients.
Such a representation is called a mixture distribution. There is a strong similarity between equation (3.1) and the expression of the unconditional density of data taken from a mixtures of several classes. $P(j)$ is equivalent to the prior probability of the data point having been generated from component j of the mixture. These priors (mixing coefficients) are chosen to satisfy the constraints:

$$\sum_{j=1}^{M} P(j) = 1\,, \tag{3.2}$$

$$0 \leq P(j) \leq 1\,. \tag{3.3}$$

The density modelling of a frame will be achieved using such a model, using for a basic component a Gaussian density function. The resulting probabilistic mapping will be used to create the segmentation algorithm.

3.1.1 Gaussian Mixture Model

Because of their probabilistic nature, Gaussian mixtures are in principle preferred over models that cut a data set $(\mathbf{x}_1, ..., \mathbf{x}_N) \in \mathbf{R}^{N*d}$ in discrete parts. They are usually providing more information than discrete models, considering a full distribution for the data set instead of sampling points. The density components are now Gaussian distributions:

$$\begin{aligned} p(\mathbf{x}|j) &\sim \mathcal{N}(\mu_j, \Sigma_j) \\ &= p(\mathbf{x}|\mu_j, \Sigma_j) \\ &= \frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_j)^T \Sigma_j^{-1} (\mathbf{x}-\mu_j)} \end{aligned} \tag{3.4}$$

The finite mixture model will be expressed by extending equation (3.1) to:

$$p(\mathbf{x}|\theta) = \sum_{j=1}^{M} \alpha_j \, p(\mathbf{x}|\mu_j, \Sigma_j) \tag{3.5}$$

where α_j are the mixing coefficients, $p(\mathbf{x}|\mu_j, \Sigma_j)$ are the Gaussian density functions and where θ are the set of constraints (equations (3.2) and (3.3)) rewritten as:

$$0 \leq \alpha_j \leq 1 \text{ and } \sum_{j=1}^{M} \alpha_j = 1 \,. \tag{3.6}$$

$$\mu_j \in \mathbf{R}^d \text{ and } \Sigma_j \in \mathbf{R}^{d*d} \text{ is a positive definite matrix.} \tag{3.7}$$

In our case, the Gaussian mixture model will be used to fit the intensity histograms in a frame. The data set of the entities we observed $\mathbf{x}_1, ..., \mathbf{x}_N$ will be a vector from the size of the picture where each $\mathbf{x}_i$ is a number in the range $0 - 255$. In the one dimensional case, the Gaussian distributions can be simplified from equation (3.4) to:

$$p(\mathbf{x}|j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(\mathbf{x}-\mu_j)^2}{2\sigma_j^2}} \tag{3.8}$$

where the previous matrix covariance noted Σ_j is now the positive real number $\frac{1}{\sigma_j^2}$.

3.1.2 Model Quality

So assuming that a data set is generated by a certain Gaussian mixture, the task is to fit a model to this data, i.e. to estimate the parameters of the generating mixture. For this, it is necessary to have some notion of the quality of the model with respect to the data set. A standard way is to take the likelihood that the data set was generated by the model itself, i.e. to what extent the distribution of the data corresponds to the model. We define the log-likelihood L, given the parameters θ and the data set $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_N$, as:

$$\begin{aligned}\mathcal{L}(\theta) &= \log p(\mathbf{X}) \\ &= \log \prod_{\mathbf{x}\in\mathbf{X}} \sum_{j=1}^{M} \alpha_j\, p(\mathbf{x}|\mu_j, \Sigma_j) \\ &= \log \prod_{n=1}^{N} \sum_{j=1}^{M} \alpha_j\, p(\mathbf{x}_n|\mu_j, \Sigma_j) \\ &= \sum_{n=1}^{N} \log \sum_{j=1}^{M} \alpha_j\, p(\mathbf{x}_n|\mu_j, \Sigma_j) \end{aligned} \tag{3.9}$$

where α_j are the mixing weights. Note that independence of the data points is assumed and therefore the likelihood of the data set is equal to the product of the likelihoods of the data points. Fitting the model to a data set is now the same as maximising (3.9) with respect to the parameters θ and the mixing weights α_j. Instead of maximising the data likelihood, an equivalent computation consists in minimising the negative log-likelihood:

$$E = -\mathcal{L}(\theta) = -\sum_{n=1}^{N} \log p(\mathbf{x}_n) \tag{3.10}$$

It is important to emphasise that minimising this error function is non-trivial [3]. For example, there exist parameter values for which the likelihood goes to infinity. These arise when one of the Gaussian components collapses onto one of the data points, as it can be by setting $\mu_j = \mathbf{x}$ and then letting $\sigma_j \to 0$ (equation (3.8)).

3.1.3 The Expectation-Maximisation Algorithm

The most popular algorithm for training a Gaussian mixture is the Expectation-Maximisation (EM) algorithm ([10], [13]). The EM algorithm iteratively modifies the GMM parameters, the mean μ_j, the variance σ_j^2 and the mixing coefficients α_j for each component j , to reach a minimum of the negative log-likelihood E.
Because the component that generated each data point $\mathbf{x}_n$ is unknown, a variable $\boldsymbol{z}_n$ is introduced, which takes on integer values in the range of $1...M$, and denotes the unknown generating component.
Then using the product rule, the complete-data log-likelihood is given by :

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

where $\theta = \{\mu_j, \sigma_j, \alpha_j\}$.

The Expectation step

As indicated by the name, this step consists of taking the expectation of the log-likelihood with respect to the distribution $P(\boldsymbol{z}) = \prod_{n=1}^{N} P(\boldsymbol{z}_n|\mathbf{x}_n, \theta^{old})$. Since $\boldsymbol{z}$ is a discrete variable, the expectation over all $\boldsymbol{z}_n$ is simply a combination of N sums :

$$\begin{aligned}
\mathcal{E}^{comp}(\theta) &= \sum_{n=1}^{N}\left[\sum_{\boldsymbol{z}_1=1}^{M}\sum_{\boldsymbol{z}_2=1}^{M}\cdots\sum_{\boldsymbol{z}_N=1}^{M}\prod_{m=1}^{N}P(\boldsymbol{z}_m|\mathbf{x}_m,\theta^{old})\log\{p(\mathbf{x}_n|\boldsymbol{z}_n,\theta)P(\boldsymbol{z}_n|\theta)\}\right] \\
&= \sum_{n=1}^{N}\left[\sum_{\boldsymbol{z}_1=1}^{M}\cdots\sum_{\boldsymbol{z}_{n-1}=1}^{M}\sum_{\boldsymbol{z}_{n+1}=1}^{M}\cdots\sum_{\boldsymbol{z}_N=1}^{M}\prod_{m\neq n}^{N}P(\boldsymbol{z}_m|\mathbf{x}_m,\theta^{old})\right] \\
&\times\left[\sum_{\boldsymbol{z}_n=1}^{M}P(\boldsymbol{z}_n|\mathbf{x}_n,\theta^{old})\log\{p(\mathbf{x}_n|\boldsymbol{z}_n,\theta)P(\boldsymbol{z}_n|\theta)\}\right]
\end{aligned}$$

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

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

Considering the previous general notation for a mixture model (equation (3.1)), the expectation can be written as:

$$\mathcal{E}^{comp}(\theta^{new})=\sum_{n=1}^{N}\sum_{j=1}^{M}P(j|\mathbf{x}_n,\theta_j^{old})\log\{p(\mathbf{x}_n|j,\theta_j^{new})\alpha_j^{new}\} \tag{3.11}$$

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

The Maximisation step

In the M-step, we maximise $\mathcal{E}^{comp}(\theta^{new})$ with respect to the parameters θ^{new}. So if we differentiate equation (3.11) and set the derivatives to zero in the univariate Gaussian case, we get :

$$\begin{aligned}
\mu_j^{new} &= \frac{\sum_{n=1}^{N}P(j|\mathbf{x}_n,\theta_j^{old})\mathbf{x}_n}{\sum_{n=1}^{N}P(j|\mathbf{x}_n,\theta_j^{old})} \\
(\sigma_j^2)^{new} &= \frac{\sum_{n=1}^{N}P(j|\mathbf{x}_n,\theta_j^{old})(\mathbf{x}_n-\mu_j^{new})^2}{\sum_{n=1}^{N}P(j|\mathbf{x}_n,\theta_j^{old})} \\
\alpha_j^{new} &= \frac{1}{N}\sum_{n=1}^{N}P(j|\mathbf{x}_n,\theta_j^{old})
\end{aligned}$$

where $\theta_j^{old}=\{\mu_j^{old},(\sigma_j^2)^{old},\alpha_j^{old}\}$.
Each iteration of the Expectation and Maximisation steps is guaranteed to increase the likelihood, unless it is already at a maximum, so E and M steps are repeated until the algorithm converges.
Among the EM algorithm advantages are its easy implementation, no need to set extra user-defined parameters, and guaranteed monotone increase in model quality. There are also some down sides, the most important of which are its high dependency on initialisation and computational complexity, which is linear with respect to the size of the data set.

The K-means algorithm

The classical method to initialise the mixture before the EM algorithm is the K-means algorithm. It is a method for finding K vectors μ_j (for $j = 1, ..., K$) that represent an entire dataset. The data is considered to be partitioned into K clusters, with each cluster represented by its mean vector and each data point assigned to the cluster with the closest vector.
The K-means algorithm works iteratively. At each stage, the N data points $\mathbf{x_i}$ are partitioned into K disjoint clusters S_j each containing N_j data points. The error function that is minimised is the total within-cluster-sum-square:

$$E = \sum_{j=1}^{M} \sum_{i \in S_j} ||\mathbf{x_i} - \mu_j||^2 \tag{3.12}$$

where μ_j is the centre of the jth cluster, given by the mean of the data points belonging to the cluster:

$$\mu_j = \frac{1}{N_j} \sum_{i \in S_j} \mathbf{x_i} \tag{3.13}$$

The initial partition of the data is at random. Then the following two steps are iterated until there is no further change change to the error E:

- The mean vectors for each cluster are calculated using equation (3.13).
- Each data point is assigned to the cluster containing the closest mean vector.

The K-means algorithm converges quickly which makes it a good tool to initialise Gaussian mixture models before optimisation with the EM algorithm.

3.1.4 Implementation

Gaussian mixture model were implemented using Netlab (function `gmm`).

Estimating the number of component needed

The choice of the parameter M is very important. This parameter should reflect the number of populations in which a pixel can be labelled. It can also be seen as the number of sources that have generated the intensity of a pixel. Therefore, this setting requires an analysis of the labels we can expected in a frame. Considering a global picture, a pixel can represent part of the sea, part of the sky or part of the background. The object we tracked are oil slicks in sea. The sea label should then be divided in a minimum of two sub labels such as "oil slicks" and "clean sea". Moreover, we can use prior knowledge about sky and background, mentioned in section 1.3.3, to reduce the size of the data. The idea is to fit only the data that are part of the sea using a template image. One dilemma that appeared in the choice of parameter M was the human activity that could appear on a frame. This unstable phenomena (it can disappear from one frame to another) can be seen as an unstable label population. Using only the 2 labels defined above (slicks/non-slicks) may bring some errors when human activity is present on picture. Pixels from that population may be randomly

classified as "oil slicks" or "clean sea". On the other hand, by choosing 3 labels, the Gaussian mixture model will be trained to retrieve 3 populations on a frame that intuitively contains 2. In that case, real "oil slick" pixel may be identified as human activity pixel.
The alternative between the two possible values results in the choice of giving more importance to True negative (pixels labelled as slicks that are not) or False positive (pixels not classified that are). While True negative may be re-filtered later on, False positive appear as a real loss of information. For this reason, the number of components was set to $M = 2$.

Training the GMM

The Gaussian mixture model was initialised using the `gmminit` Netlab function. This function takes for parameters the Gaussian model to fit to the data (resulting from `gmm` function) and the data to fit (here a sub-frame taken from the sea). The mixture is then trained using the EM algorithm coded in the function `gmmem`. Documentation and details of the code of these functions are provided in the Netlab book [11].

Density Models retrieved

The 2 Gaussian mixture model was trained using the code described in section 3.1.4 on different frames.
In both case, the data used to fit the mixture model was the part of the sea in which we train the RBF network for retrieving the reflectance image.

3.2 Segmenting the image

The segmentation process consists of establishing a relevant probability mapping and use it to create and separate specific populations of pixels.
Obviously, the Gaussian mixture fitted to the data will be taken as the model for the sources that have generated the different populations. The probability mapping will be extracted from the Gaussian mixture model and processed to achieve the classification.

3.2.1 Principle

Our Gaussian mixture is composed of 2 normal components that were fitted to the data inside a sea window (see Figure 3.1). It provides us with an intensity density model of a part of the frame. If we consider a pixel $\mathbf{x}$ with an intensity $I_{\mathbf{x}}$, the probability of finding this pixel intensity in the selected part of the frame is given by the equation:

$$p(I_{\mathbf{x}}) = \sum_{j=1}^{2} \alpha_j . p(I_{\mathbf{x}} | \mu_j, \Sigma_j) \tag{3.14}$$

In our classification problem, we are more interested in finding the probabilities of the contribution of each source in the pixel intensity. These probabilities are called posteriors. They can be computed using Bayes' formula:

$$P_{post}(G_i | I_{\mathbf{x}}) = \frac{P(I_{\mathbf{x}} | G_i) P(G_i)}{P(I_{\mathbf{x}})} \tag{3.15}$$

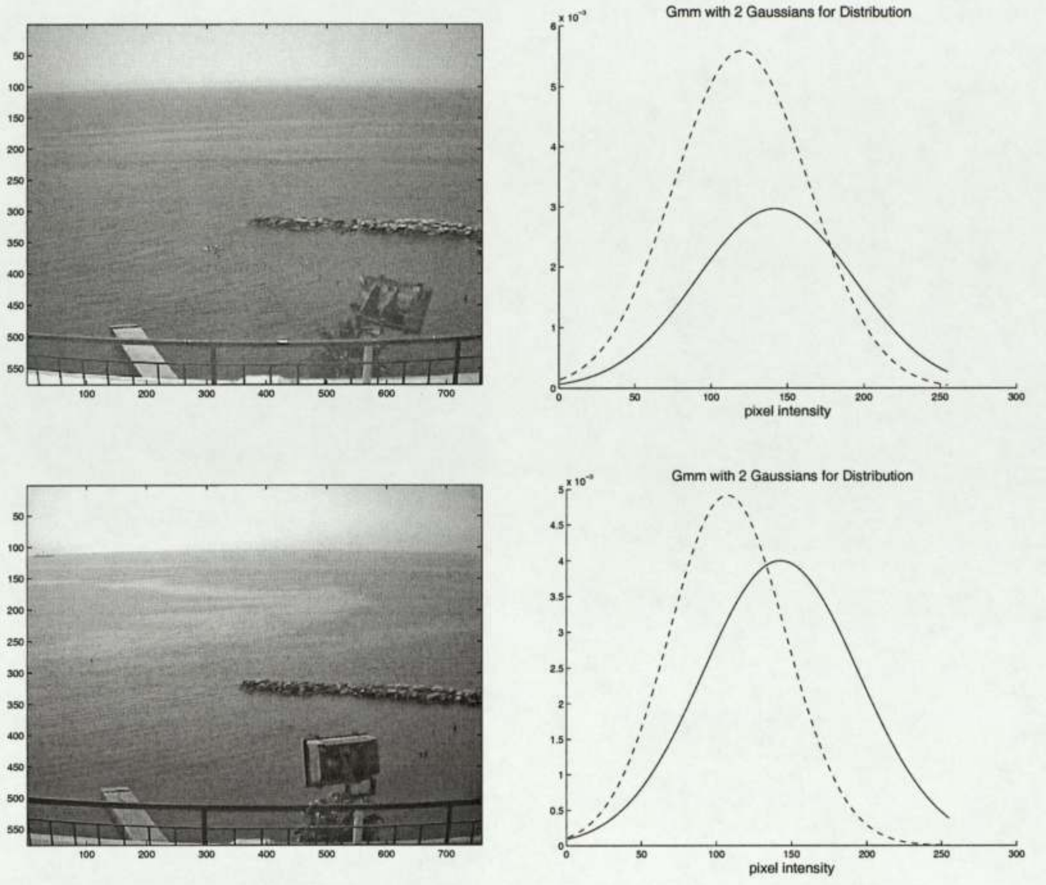


Figure 3.1: Gaussian Mixtures retrieved on different frames.

where G_i is the event "The intensity of the pixel $\mathbf{x}$ was generated by source i",and $I_\mathbf{x}$ the intensity of the pixel $\mathbf{x}$. Because the sources are Gaussian, we have:

$$P_{post}(I_\mathbf{x}|G_i) = p(\mathbf{x}|i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(\mathbf{x}-\mu_i)^2}{2\sigma_i^2}} \tag{3.16}$$

and using the definition of a mixture model

$$P(G_i) = \alpha_i \tag{3.17}$$

where α_i is the mixing coefficient associated with the source i. The posterior probability mapping can be retrieved by combining equations (3.15), (3.16) and (3.17).
The next step will be to select a Gaussian that should reflect the feature we want to identify: the "oil slicks". Once the source for our feature is selected, the classification of pixel intensity will be achieved in respect of the most probable source for every intensity value. In a mixture model with M components, a pixel will be classified as generated by the source i if:

$$\forall j \in [1, M], P_{post}(I_\mathbf{x}|G_i) \geq P_{post}(I_\mathbf{x}|G_j) \tag{3.18}$$

With 2 components, equation (3.18) is simplified. Indeed, if we only consider the 2 sources i and j, we have:

$$P_{post}(I_\mathbf{x}|G_i) = 1 - P_{post}(I_\mathbf{x}|G_j) \tag{3.19}$$

and therefore a pixel will be classified as generated by the source i if:

$$P_{post}(I_{\mathbf{x}}|G_i) \geq 0.5\,, \tag{3.20}$$

the equality from equation (3.18) being obtained from:

$$P_{post}(I_{\mathbf{x}}|G_i) = P_{post}(I_{\mathbf{x}}|G_j) = 0.5 \tag{3.21}$$

3.2.2 Selecting the right source

An important issue, to achieve a good segmentation using a mixture model, is to select correctly which component is the suspected source of the desired feature. The choice of the source for our frame feature (the oil slick) should be based on specific intensity properties of this object, properties that allow to differentiate "oil slick" pixels from sea pixels. In our case, the choice is simplified by the fact that the mixture model is constituted of only two components.
The method elaborated to select the source should be very simple, such as a method based on a single property, in order to simply its implementation.
The following paragraphs will present the two methods found to select the source. They will be compared in section 3.3.

Maximum Intensity Centre

Given the mixture model for our density, the first method to select the source was to select the one with the highest intensity value for its centre. This method is based on the principal property : the object tracked is supposed to be brighter. Therefore, we suppose it has been generated from a distribution centred around the highest intensity values.

Smallest Gaussian

The second method for selecting the source was based on the properties of "oil slicks" as a cluster. The "oil slicks" are supposed to be sparse and diffuse on a frame. Thus, they should have been generated from a small source (they are details). On the other hand, clear sea surface is usually standing in a background of uniform intensity levels. In addition, clear sea surface usually represents most of the pixel intensity space. Thus, sea surface should be covered by a sharp and strong Gaussian in the retrieved mixture. If the "oil slicks" source can be defined as a small Gaussian, the choice remains between the Gaussian with the smallest variance and the Gaussian with the smallest maximum. As said previously, "clear sea surface" can be defined by a sharp Gaussian (intensity level almost uniform). Therefore, the smallest Gaussian was chosen as the one with smallest maximum.
It is important to note from here that this maximum was compared in respect to the normalisation factor retrieved from the mixture. This normalisation factor is the mixing weight α_j (prior).

3.2.3 Implementation

The posterior probability from the 2 Gaussian mixture model was retrieved using the `gmmpost` Netlab function. This function takes for parameters the Gaussian model fitted to the data (resulting from previous training) and the interval of intensity.
The computation of Gaussian mixture modelling and the posterior probability matrix were regrouped in a Matlab function named `probmat2g`. This function returns the specific posterior probability matrix of a pixel from the sea window being generated by the selected source. This matrix is used for segmenting the image.
Segmenting is then achieved finding the pixel whose intensity probability, for belonging to the selected source, is higher than 0.5. These pixel are coloured and should represent the slicks.

3.3 Results

Segmentation on Frames containing Slicks

Our segmenting methods were tested on two different sets of pictures containing examples of slicks.

- First Set

The first set was chosen for its diffuse but visible tight prints of slicks. This set is representative of frames containing "oil slicks". The 2 Gaussian mixture was fitted to the data preceding the segmentation.
As shown by the Gaussian mixture retrieved (top left graph in figure 3.2), both methods to select the Gaussian source are equivalent. Indeed, in a usual frame containing "oil slicks" both properties for this selection are verified (the oil slicks are generated by the brightest source and are representing a small area of the picture).
In the posterior probability graph (right superior corner in Figure 3.2), each of the Gaussian sources identified dominate in different intensity levels. Indeed, the posterior probability presents only one intersection point, obtained for the intensity value $I \approx 140$, which is the boundary for the predominance of each source in each intensity level. Using that boundary to colour the frame, segmentation achieved matches with the human visual identification of slicks.
This set reveals that having identified the good source (both methods are equivalent) our model is capable of tracking small prints.

- Second Set

The second set chosen had the characteristic of containing one of the maximum prints of sea surface representing "oil slicks". The brighter surface is wide and can easily be separated from the "clear sea surface" by human eye.
The aim of using this set is to test the limits of the selection method. In this case the second method, which assumes that "oil slicks" represent small area, may fail in selecting the right Gaussian.

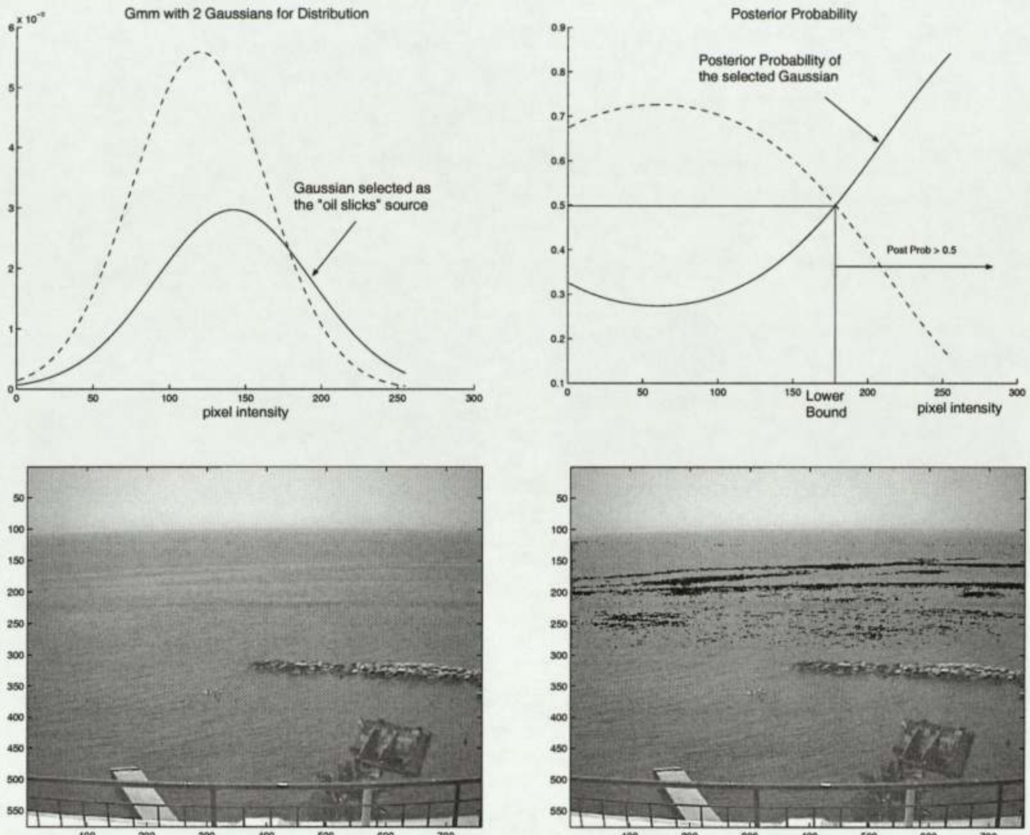


Figure 3.2: Segmentation process on set 1.

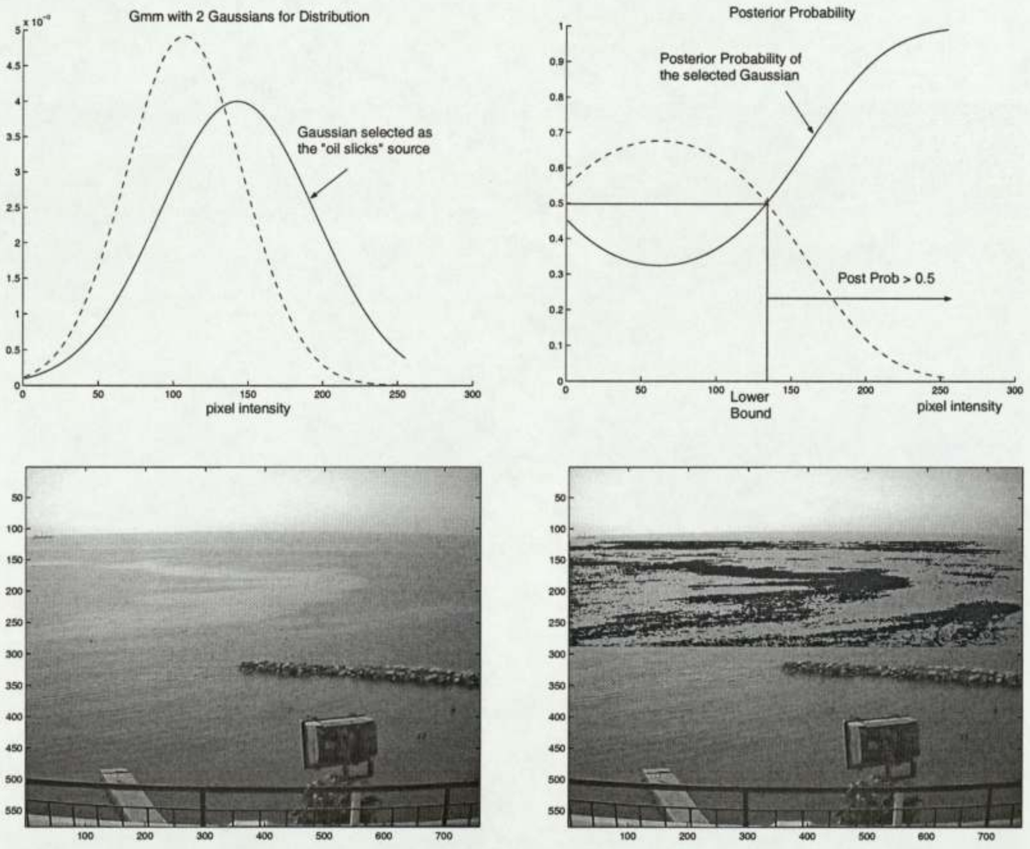


Figure 3.3: Segmentation process on set 2.

As revealed by the mixture graph, both methods lead again to select the same Gaussian. Even if the Gaussian selected has a bigger maximum value than in the previous test, it is still under the maximum of the main Gaussian representing clear sea surface which has an almost uniform intensity level.
Once again, the Gaussians retrieved dominate separate intensity intervals (one intersection point), and segmentation and coloration processes using posterior probability give very good results as can be seen by visual comparison between the original picture and the segmented picture.
The next step was testing our segmentation method on random frames.

Segmentation on a random frame

The segmentation was compared on different frames randomly chosen from the picture database.
In most of the cases, both selection method for choosing the source had the same results.
However, in frames that appeared to be empty of "oil slicks", both methods can give opposite results. These cases can be illustrated with the following set.
Using the maximum intensity centre method for selecting the source, we capture the principal Gaussian. In the posterior probability, this source is appearing to have generated pixels whose intensity is average, in the range $75 - 200$. Such a range corresponds to the uniform intensity level of clear sea level. As revealed by the segmentation, the Gaussian source selected was the wrong one. It leads to colouring the clear sea as oil slicks.
Using the second technique on the same frame, we select the smallest Gaussian which is the opposite source as the one selected previously.
The smallest Gaussian is based on not capturing the clear sea level. It captures details which could be from both high level intensity or low level (biggest variance).
This method was selected as the default one because of its higher adaptability.

Using the Posterior Probability Mapping

With its implementation, it is possible to retrieve both posterior probability mappings. Another idea could be to use these posterior probability mappings to choose the source that has generated the brightest pixel.
Tests of this method let it appear as similar to the previous except in the case of low intensity images. Indeed, in that case, it is frequent that one Gaussian completely overlaps another (See Figure 3.5).
In low intensity frames, oil slick activity is hardly traceable. By using the highest posterior method all the sea is coloured as slick which is hardly reliable. This problem can be solved with our original method.

3.3.1 Issues

As mentioned before, an issue in segmenting the image into two classes is the human activity. Human activity usually corresponds in the pixel space to sharp effects on the intensity values.
A fifth set was chosen with the characteristic of containing a significant print of human

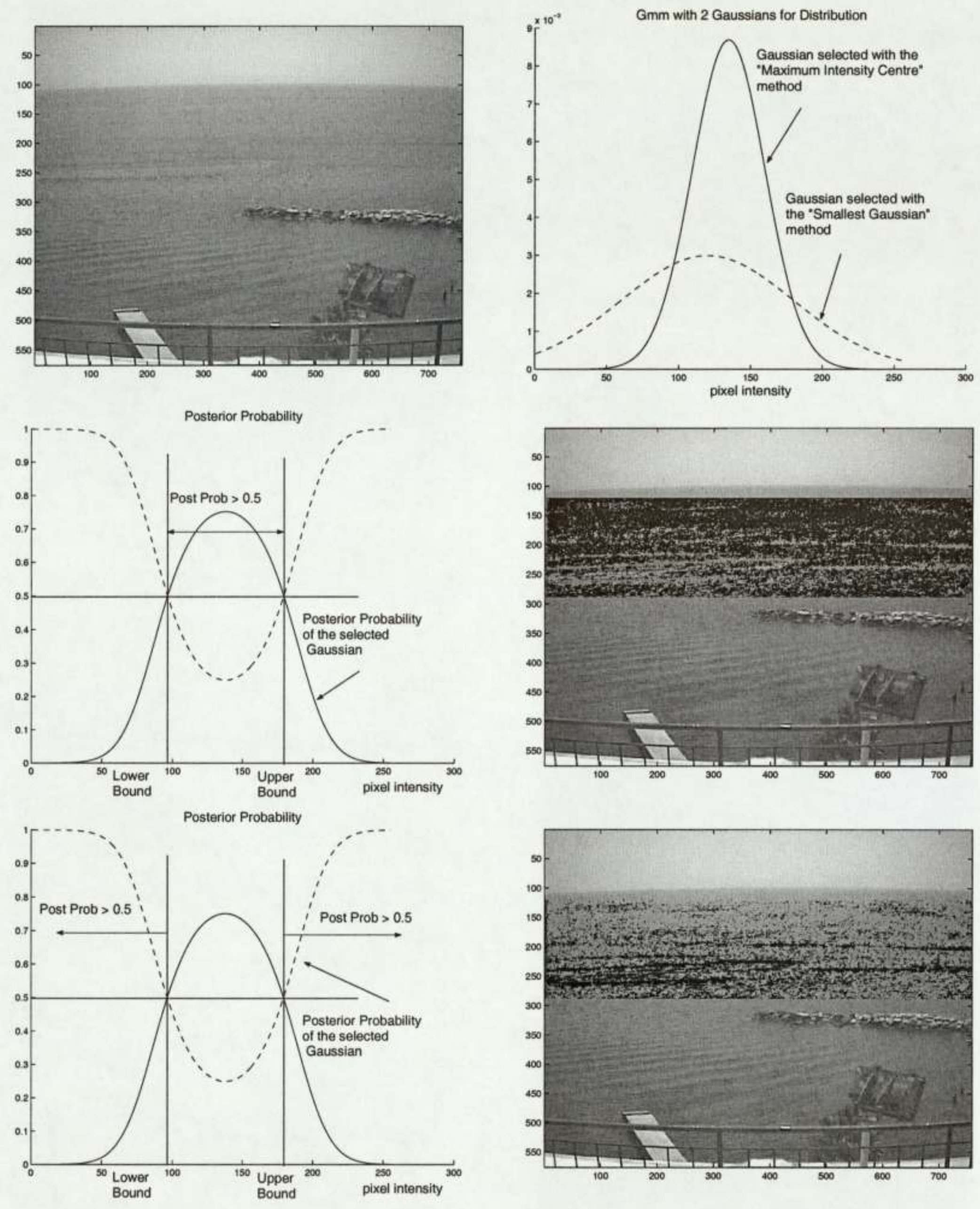
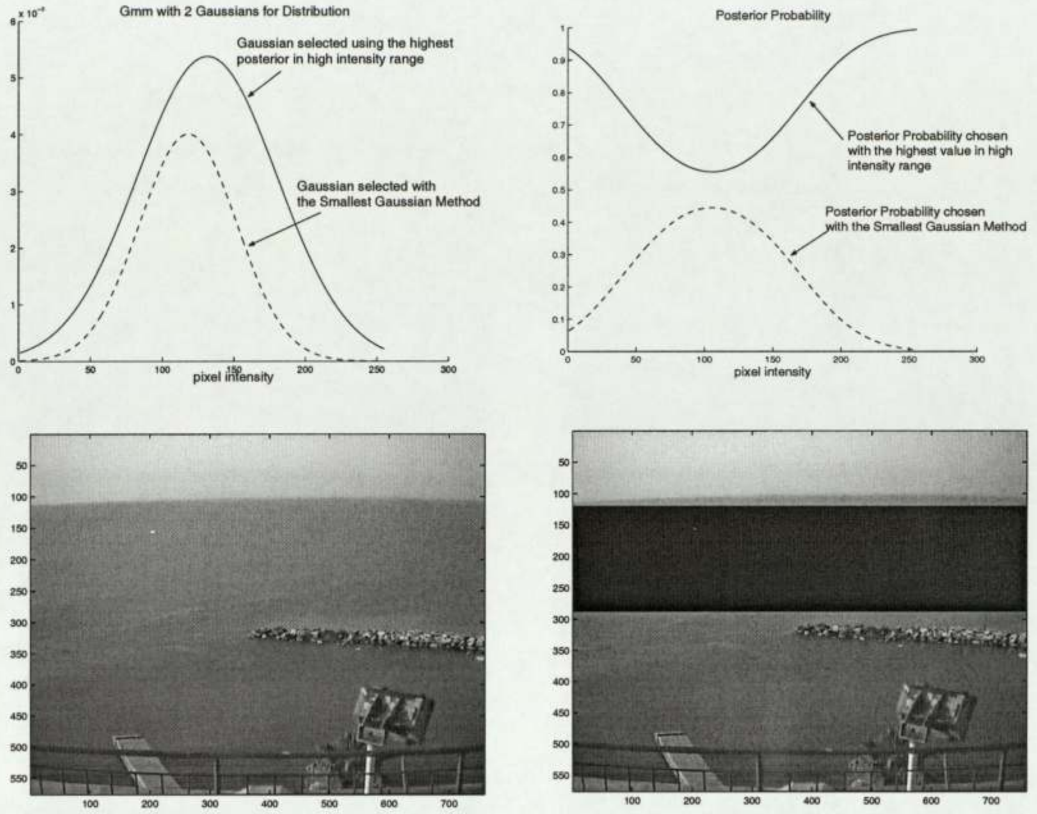


Figure 3.4: Comparing the methods on set 3.



Segmentation with the Smallest Gaussian method

With the highest posterior in high intensity levels source

Figure 3.5: Comparing previous method with a method using the posteriors.

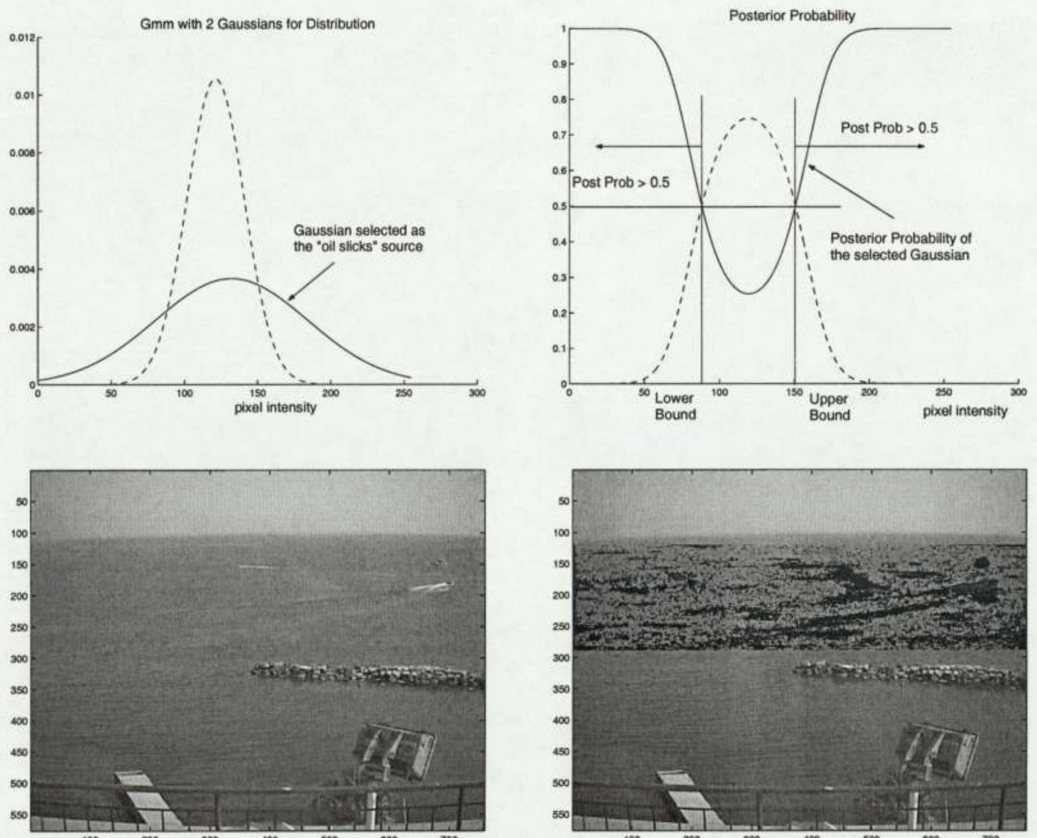
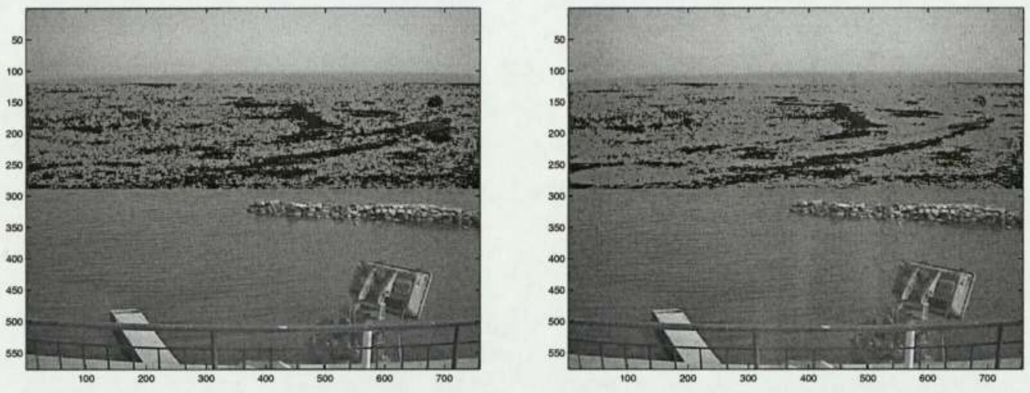


Figure 3.6: Segmentation of frame with prints of human activity.



Colouration of both ranges. Colouration of the upper range only.

Figure 3.7: Enhanced colouration cutting the lower range retrieved from the posterior probability mapping.

activity noise (2 boats). Part of the sea surface, in the range of the boat wakes, has properties similar to the bright effect of slicks on sea surface whereas the object itself stands in very low intensity level (dark objects). The segmentation process may reveal the limits of our two Gaussian mixture model.
When a third population is appearing, our 2 Gaussian mixture model will be fitted according to that data (see section 3.1.4). In the case of human activity, the prints usually represent small areas of the sea window processed.
As presented before, our method of selecting the smallest Gaussian is likely to capture the details. Using this method, we are therefore more likely to classify human activity as slick. This issue is confirmed by the graphs presented from the segmentation of the fifth set (Figure 3.6).
In the posterior probability space, segmentation will result in selecting two intensity ranges:

- A range of low values $(0 - 90)$ which corresponds to the dark pixel composing the human activity (such as boats)
- A range of high values $(150 - 255)$ which corresponds to the oil slicks and the bright pixel composing the human activity (such as the wake of boats)

The first range, corresponding to the dark part of human activity, can easily be thresholded using the upper bound retrieved from the segmentation (See Figure 3.7). One of the main problems that remains is the bright pixels composing the human activity that remains classified as oil slicks.

3.4 Summary

In this chapter, we have considered the problem of image segmentation and some of the specific problems raised. We found that a Gaussian mixture model was hard to set in order to get acceptable results. The main difficulties were:

- to estimate the number of components of the mixture needed to fit properly the population groups in an image.

- to create a method for selecting which component is the suspected source of the "oil slick" pixels.
- given the posterior probabilities and the selected Gaussian, to find the intensity boundaries that allow to classify a pixel as an oil slick pixel or a clear sea pixel.

We have not treated slicks as extended continuous features, nor have we considered the dynamics of slicks. This latter issue will be considered in the next chapter. The first issue could be part of a future project where the prior on spatial continuity is exploited as higher level knowledge, based on the lower level probabilistic pixel segmentation discussed in this chapter.

Chapter 4

Using a Dynamic Model

An interesting point of this project is the fact that the camera installed in Yermasoyia is providing an image every 10 minutes and so is indirectly giving information on sea surface dynamic activity. This chapter describes a preliminary attempt to use information across a sequence of frames to improve the segmentation achieved in the previous chapter.
The idea is to establish a model that could fit the motion of slicks across the sea. Using a sequence of observations and a reliable model of slick dynamics, we may be able to identify and track them more accurately.
From direct observations of slick activity across the ocean, the parameters involved in this evolution are numerous and belong to complex models. The slick dynamics are directly linked with the surface wind dynamics and sea dynamics. A model to define slick dynamics would appear to be very complex and will involve other sources, such as wind and waves, from which no information can be easily extracted from an image. However, the motion of slicks across the sea is a slow-time scale process compared to sea motion, wind activity or human activity. For example, it is possible to track the same slick across two consecutive frames contrary to waves and human activity.
Even if the dynamics involved are complex, it may be possible to use a discretised linear model to fit the evolution of slicks in the frame space. This model should be estimated according to observations.
A general and famous technique to fit such a model to our data is the Kalman filter ([8], [6], [4]).

4.1 The Kalman Filter

4.1.1 Theory

The idea of Kalman filtering [6] is to consider a system represented by the state vector $\mathbf{x}_t$ that we update in discrete time according to a distribution function $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. This internal state vector is not directly accessible, but rather must be inferred from measurements of observable $\mathbf{y}_t$ which are related to $\mathbf{x}_t$ by a relation $p(\mathbf{y}_t|\mathbf{x}_t)$. The Kalman filter provides a general solution to combined the measurements to estimate the system's state. Moreover, this filter allows us to update the state vector iteratively every time a new measurement is taken without the need of the previous ones or state estimations.

The variables involved in the Kalman filter are listed below and associated with figure 4.1:

$\mathbf{x}_t$	the model's parameters vector (or state vector)
$\Sigma_t^{\mathbf{x}}$	the state vector's uncertainty
$\mathbf{A}_t$	the evolution matrix from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$
η_t	the white noise involved in the evolution process (with covariance matrix Σ_t^{η})
$\mathbf{y}_t$	the observation vector
$\mathbf{B}_t$	the matrix associated to the process from which the observation is generated
$\mathbf{W}_t$	some weight matrix for η_t
ϵ_t	the white noise (i.e. Gaussian with zero mean and covariance matrix Σ_t^{ϵ}) inherent to the observation's measurement
$\mathbf{Y}_t$	the observations set from $\mathbf{y}_0$ to $\mathbf{y}_t$
$\hat{\mathbf{x}}_{t\|t-1}$	our estimation of $\mathbf{x}_t$ knowing the previous data
$\hat{\Sigma}_{t\|t-1}^{\mathbf{x}}$	our estimation of the covariance matrix for $\mathbf{x}_t$ knowing the previous data
$\hat{\mathbf{x}}_{t\|t}$	the best estimation of $\mathbf{x}_t$ knowing the previous and the new data
$\hat{\Sigma}_{t\|t}^{\mathbf{x}}$	our estimation of the covariance matrix for $\mathbf{y}_t$ knowing the previous and new data
$\hat{\mathbf{y}}_{t\|t-1}$	our estimation of $\mathbf{y}_t$ knowing the previous data
$\Sigma_t^{\mathbf{y}}$	our uncertainty about the estimation of $\mathbf{y}_t$

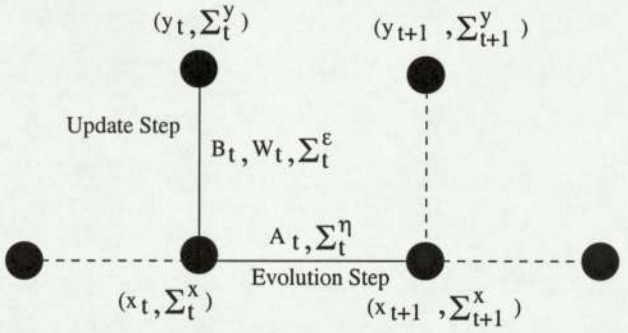

(y_t, Σ_t^y)
$(y_{t+1}, \Sigma_{t+1}^y)$
Update Step
$B_t, W_t, \Sigma_t^\varepsilon$
A_t, Σ_t^η
Evolution Step
(x_t, Σ_t^x)
$(x_{t+1}, \Sigma_{t+1}^x)$

Figure 4.1: Diagram of the Kalman filter.

The system's update is assumed to be linear, with additive noise η_t

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \eta_t \tag{4.1}$$

and that the observable $\mathbf{y_t}$ (measurements) is linked to the state by the linear relationship with additive noise ϵ_t:

$$\mathbf{y}_t = \mathbf{B}_t \mathbf{x}_t + \mathbf{W}_t \epsilon_t \tag{4.2}$$

Both noise η_t and ϵ_t are supposed to be Gaussian white noise and uncorrelated in time. They will be defined by two covariances matrices Σ_t^η and Σ_t^ϵ.
The Kalman filter is a two step iterative algorithm. It basically involves the forecast of the state vector using a dynamic rule (evolution step) and its update using the new observation (update step).
It is based on finding the maximum a posteriori (MAP) estimate of the state vector at time t given the observable and our estimate of the state at the previous time step $t-1$, i.e. $\hat{\mathbf{x}}_{t|t-1}$. If we denote all the observations received up to and including time t by $\mathbf{Y}_t$, the posterior density can be written using Bayes' Theorem as:

$$p(\mathbf{x}_t|\mathbf{Y}_t) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{Y}_{t-1})}{p(\mathbf{y}_t|\mathbf{Y}_{t-1})} \tag{4.3}$$

All these densities are assumed to be Gaussian.

- Likelihood: It contains the information from the new datum. With the noise process ϵ_t with zero mean and covariance Σ_t^ϵ, we have the likelihood

$$p(\mathbf{y}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{B}_t\mathbf{x}_t, \Sigma_t^\epsilon) \tag{4.4}$$

- Prior: The prior from equation (4.3) contains all the information obtained from previous observations. Assuming a Gaussian, this distribution will be centred on the state vector estimate at time t given all the observations up to time $t-1$, $\hat{\mathbf{x}}_{t|t-1}$. The covariance matrix of this distribution will also be an estimate of $\Sigma_t^{\mathbf{x}}$, i.e.

$$p(\mathbf{x}_t|\mathbf{Y}_{t-1}) = \mathcal{N}(\hat{\mathbf{x}}_{t|t-1}, \hat{\Sigma}_{t|t-1}^{\mathbf{x}}) \tag{4.5}$$

- Evidence: In equation (4.3), the denominator gives the probability of the new data point observed given all the previous ones. Assuming a Gaussian again, the mean of this distribution will be our best prediction of the new data vector given the previous data, i.e. $\hat{\mathbf{y}}_{t|t-1}$, and the covariance matrix, denoted $\Sigma_t^{\mathbf{y}}$, will represent our uncertainty about that prediction. Thus, evidence can be expressed as:

$$p(\mathbf{y}_t|\mathbf{Y}_{t-1}) = \mathcal{N}(\hat{\mathbf{y}}_{t|t-1}, \Sigma_t^{\mathbf{y}}) \tag{4.6}$$

- Posterior: Given the previous Gaussian, the posterior turns out to be a Gaussian centred on the best state vector estimate given all the data up to and including time t. The covariance matrix will be our updated estimate of the state vector uncertainty $\Sigma_t^{\mathbf{x}}$:

$$p(\mathbf{x}_t|\mathbf{Y}_t) = \mathcal{N}(\hat{\mathbf{x}}_{t|t}, \hat{\Sigma}_{t|t}^{\mathbf{x}}) \tag{4.7}$$

Update step

The first step is to search for the MAP estimate which is the value of $\hat{\mathbf{x}}_{t|t}$ that minimises the negative log of the posterior with respect to $\mathbf{x}_t$. For this, we take the negative log of equation (4.3), expressing Bayes' rule, and rewrite each density applying the previous assumption. Thus, we get the equation, dropping the constant term from each Gaussian:

$$\begin{aligned} -\ln p(\mathbf{x}_t|\mathbf{Y}_t) &\propto (\mathbf{x}_t - \hat{\mathbf{x}}_{t|t})^T\ (\hat{\Sigma}^{\mathbf{x}}_{t|t})^{-1}\ (\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}) && (4.8)\\ &\propto (\mathbf{y}_t - \mathbf{B}_t\mathbf{x}_t)^T\ (\Sigma^{\epsilon}_t)^{-1}\ (\mathbf{y}_t - \mathbf{B}_t\mathbf{x}_t)\\ &+\ (\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})^T\ (\hat{\Sigma}^{\mathbf{x}}_{t|t-1})^{-1}\ (\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})\\ &-\ (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})^T\ (\Sigma^{\mathbf{y}}_t)^{-1}\ (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}) && (4.9)\end{aligned}$$

Equating the quadratic terms in $\mathbf{x}_t$, we get:

$$(\hat{\Sigma}^{\mathbf{x}}_{t|t})^{-1} = \mathbf{B}_t^T\,(\Sigma^{\epsilon}_t)^{-1}\,\mathbf{B}_t + (\hat{\Sigma}^{\mathbf{x}}_{t|t-1})^{-1} \qquad (4.10)$$

By differentiating equation (4.9) with respect to $\mathbf{x}_t$ and setting it to zero in order to find the MAP we have:

$$\hat{\mathbf{x}}_{t|t} = (\mathbf{B}_t^T\,(\Sigma^{\epsilon}_t)^{-1}\,\mathbf{B}_t + (\hat{\Sigma}^{\mathbf{x}}_{t|t-1})^{-1})^{-1}\,((\hat{\Sigma}^{\mathbf{x}}_{t|t-1})^{-1}\,\hat{\mathbf{x}}_{t|t-1} + \mathbf{B}_t^T\,(\Sigma^{\epsilon}_t)^{-1}\,\hat{\mathbf{y}}_t) \qquad (4.11)$$

By combining equation (4.10) with equation (4.11), we get:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + K_t\,e_{t|t-1} \qquad (4.12)$$

with

$$K_t = \hat{\Sigma}^{\mathbf{x}}_{t|t}\,\mathbf{B}_t^T\,(\Sigma^{\epsilon}_t)^{-1}$$

called the Kalman gain, and

$$e_{t|t-1} = \mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1}$$

the estimate error.
Equation (4.12) states that the new estimate evolves by having added a certain quantity of error at each step, this quantity being the filter's gain.
These expressions involve a large number of matrix inversions to obtain the updated state covariance $\hat{\Sigma}^{\mathbf{x}}_{t|t}$. It can be shown that they can be rewritten in a more computationally tractable form such as:

$$\begin{aligned} K_t &= \hat{\Sigma}^{\mathbf{x}}_{t|t-1}\,\mathbf{B}_t^T\,(\Sigma^{\mathbf{y}}_t)^{-1} && (4.13)\\ \Sigma^{\mathbf{y}}_t &= \mathbf{B}_t\,\hat{\Sigma}^{\mathbf{x}}_{t|t-1}\,\mathbf{B}_t^T + \Sigma^{\epsilon}_t && (4.14)\\ \hat{\Sigma}^{\mathbf{x}}_{t|t} &= (I - K_t\,\mathbf{B}_t)\,\hat{\Sigma}^{\mathbf{x}}_{t|t-1} && (4.15)\end{aligned}$$

where I is identity matrix.

Evolution step

In the time interval between two observations, the state vector is evolving according to equation (4.1). Thus, the estimates of $\mathbf{x}$ and $\Sigma^{\mathbf{x}}$ have to evolve while waiting for a new observation.
We are interested in obtaining $p(\mathbf{x}_{t+1}|\mathbf{Y}_t)$ the probability of the parameters at time $t+1$ given the observation up to time t.
After processing the last data point, we have obtained the posterior distribution $p(\mathbf{x}_t|\mathbf{Y}_t)$,

which is the probability of the parameters at time t given the data up to and including time t. Since the previous step is a probability, we need to integrate the joint probability of $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$ over the whole possible values for $\mathbf{x}_t$:

$$p(\mathbf{x}_{t+1}|\mathbf{Y}_t) = \int p(\mathbf{x}_{t+1}, \mathbf{x}_t|\,\mathbf{Y}_t)\, d\mathbf{x}_t \tag{4.16}$$

Using Bayes' rule on this joint density and the fact that $\mathbf{x}_{t+1}$ only depends on the observations through $\mathbf{x}_t$ we get:

$$\begin{aligned} p(\mathbf{x}_{t+1}|\mathbf{Y}_t) &= \int p(\mathbf{x}_{t+1}, \mathbf{x}_t|\,\mathbf{Y}_t)\, d\mathbf{x}_t \\ &= \int p(\mathbf{x}_{t+1}|\,\mathbf{x}_t, \mathbf{Y}_t)\, p(\mathbf{x}_t|\,\mathbf{Y}_t)\, d\mathbf{x}_t \\ &= \int p(\mathbf{x}_{t+1}|\,\mathbf{x}_t)\, p(\mathbf{x}_t|\,\mathbf{Y}_t)\, d\mathbf{x}_t \end{aligned} \tag{4.17}$$

The posterior $p(\mathbf{x}_t|\,\mathbf{Y}_t)$ was obtained at the previous step using equation (4.7):

$$p(\mathbf{x}_t|\mathbf{Y}_{t-1}) = \mathcal{N}(\hat{\mathbf{x}}_{t|t}, \hat{\Sigma}^{\mathbf{x}}_{t|t})$$

$p(\mathbf{x}_{t+1}|\,\mathbf{x}_t)$ can be computed using the evolution equation (4.1):

$$p(\mathbf{x}_{t+1}|\,\mathbf{x}_t) = \mathcal{N}(\mathbf{A}_t\,\mathbf{x}_t, \mathbf{A}_t\,\hat{\Sigma}^{\mathbf{x}}_{t|t}\,\mathbf{A}_t^T + \mathbf{W}_t\,\Sigma^{\eta}_t\,\mathbf{W}_t^T)$$

(the mean of $\mathbf{x}_{t+1}$ is the mean of $\mathbf{x}_t$ after evolution, and the uncertainty is the covariance matrix of the weighted evolution's noise). From there, using the properties of Gaussian integration, we obtain the evolved state distribution:

$$p(\mathbf{x}_{t+1}|\,\mathbf{Y}_t) = \mathcal{N}(\hat{\mathbf{x}}_{t+1|t}, \hat{\Sigma}^{\mathbf{x}}_{t+1|t}) \tag{4.18}$$

with

$$\hat{\mathbf{x}}_{t+1|t} = \mathbf{A}_t\,\hat{\mathbf{x}}_{t+1|t} \tag{4.19}$$

$$\hat{\Sigma}^{\mathbf{x}}_{t+1|t} = \mathbf{A}_t\,\hat{\Sigma}^{\mathbf{x}}_{t|t}\,\mathbf{A}_t^T + \mathbf{W}_t\,\Sigma^{\eta}_t\,\mathbf{W}_t^T \tag{4.20}$$

4.2 Implementation

The Kalman filter has been implemented in `matlab` to enhance the tracking of slicks. Indeed, the site chosen provides us with an image every 10 minutes.

The use we made of the Kalman filter was unconventional. A Kalman filter is usually used to get a best approximation of a real value which belongs to physical phenomena such as an acceleration, a position or a velocity. In our case, we want to enhance the detection of slicks from which we have no other information than the probabilistic mapping computed from the segmentation of a frame and some general physics knowledge about the movement of oil substances on sea surfaces.

The Kalman filter was built to take as a state vector $\mathbf{x}_t$ the "true" and unaccessible probability mapping (Figure 4.2) of being a slick pixel whereas the observation $\mathbf{y}_t$ was

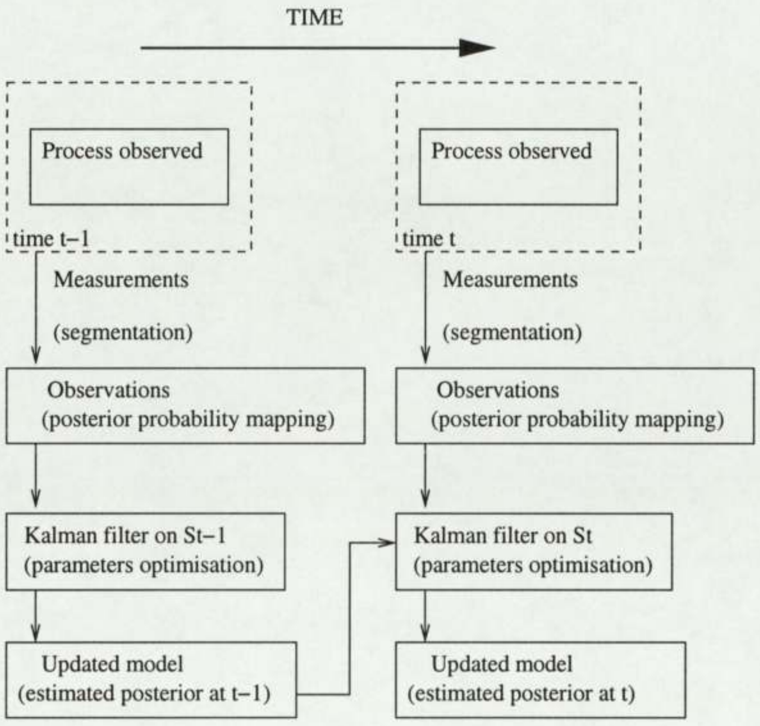


Figure 4.2: Implementation of the Kalman filter.

the posterior probability retrieved from density modelling (driven by equation (3.16)). The posterior probabilities are now labelled with time:

$$P_{post}(I_{\mathbf{x}}|G_i)_{[t]} = p(\mathbf{x}|i)_{[t]} = \frac{1}{\sqrt{2\pi\sigma_i^2}}\, e^{-\frac{(\mathbf{x}-\mu_i)^2}{2\sigma_i^2}} \tag{4.21}$$

where G_i represents the Gaussian selected as the oil slick source. The first part of the implementation was to code the Kalman filter algorithm. Two files were created to achieve the filtering:

- The function `Kstep` is representing one iteration of the filter including the two steps described in the theory.
- The function `Kfilter` is the general procedure that takes a new observation, retrieved every 10 minutes, and runs an iteration on it.

The second part of the implementation was to define the Kalman filter parameters. Indeed, in order to run a Kalman filter, we need values for evolution matrix $\mathbf{A}_t$, $\mathbf{B}_t$ and $\mathbf{W}_t$, uncertainties $\Sigma_t^{\mathbf{x}}$ and $\Sigma_t^{\mathbf{y}}$ and the noise covariances Σ_t^{ϵ} and Σ_t^{η}.
These parameters are usually obtained from models about the dynamics of the state vector. In the case of oil slicks, such models would include dynamics about sea current, dynamics about wind of the sea surfaces and the shape of the site such as 3D information about the sea bed. We do not have access to this information. The implementation of the Kalman filter that was made is a very crude version of the optimal one. Indeed, the optimal model seems far too complicated and requires information that could not be obtained.

Matrices $\mathbf{A}_t$, $\mathbf{B}_t$ and $\mathbf{W}_t$, which should refer to dynamics of the probability mapping, were set as $\mathbf{A}_t = I$, $\mathbf{B}_t = I$ and $\mathbf{W}_t = I$ where I is the identity matrix. First, this means that the filtering applied does not imply any interaction between pixels (identity matrix is diagonal). It is applied to each pixel separately. Moreover, the state vector, which is here the mean of the posterior probability, will evolve only regarding to the noise covariances Σ_t^{ϵ} and Σ_t^{η} and the initialisation specified by setting the uncertainties $\Sigma_t^{\mathbf{x}}$ and $\Sigma_t^{\mathbf{y}}$.
The uncertainties $\Sigma_t^{\mathbf{x}}$ and $\Sigma_t^{\mathbf{y}}$ about the observation and the true probability are re-estimated after each step of the Kalman filter. They were initialised so that $\Sigma_t^{\mathbf{x}} = 0.1 \times J$ and $\Sigma_t^{\mathbf{y}} = 0.1 \times J$ where J is a matrix of ones. The initialisation of the true vector $\mathbf{x}_0$ was done so that every pixel represents an equal probability of being a slick or not. Therefore, it was set with $\mathbf{x}_0 = 0.5 \times J$.
To set values for Σ_t^{ϵ} and Σ_t^{η}, we had to guess about the noise covariances of the measurements and the true probability. The probability we want to filter is mainly based on pixel intensity (chapter 3). It should evolve in respect to it. As presented in the introduction part, a pixel represents a distortion of the real space. Due to the incident angle, a pixel from the background is representing a much bigger space than a pixel present in the foreground (section 1.2.3). Moreover, the intensity captured by a pixel can be viewed as the average intensity over the real surface covered. Therefore, a bright pixel in the background, segmented as a slick due to its probabilities, should be considered more accurate in the way that it is representing an average over a bigger surface. Another way to translate this idea, is that, during the same interval of time, we expect less changes in the probability from a background pixel than one from the foreground.
Thus, noise covariances matrices were set with:

$$\Sigma_t^{\epsilon} = k_1 \times \frac{1}{Area} \tag{4.22}$$

$$\Sigma_t^{\eta} = k_2 \times \frac{1}{Area} \tag{4.23}$$

where k_1 and k_2 are scalars and $Area$ is a matrix representing the real surface of pixels, $\frac{1}{Area}$ representing a quantity proportional to the uncertainties on the observation evolution and the true state evolution.
As explained in the section 1.2.3, the main problem for retrieving the real size of the pixel is the lack of information regarding the camera field-of-view aspect ratio. The matrix $Area$ was approximated using general knowledge about distance to horizon and using some objects present in the field-of-view. A pixel was approximated to be representing a square real surface. An approximation of the real surface could then be retrieved by finding the real width and the real height of every pixel. The distortion effect due to the camera lens (horizon not appearing as a straight line) was neglected. Therefore, two pixels on the same line were considered having the same depth.

- Depth: Knowing the height of the camera ($h = 25$m) it is possible to retrieve the limit of the view defined by the horizon (figure 4.3).
  This distance to horizon can be approximated using $R_{earth} >> h$ and the Al Kashi Theorem [15] (generalised Pythagoras) to:

$$D_{Horizon} = \sqrt{2\, h\, R_{earth}}$$

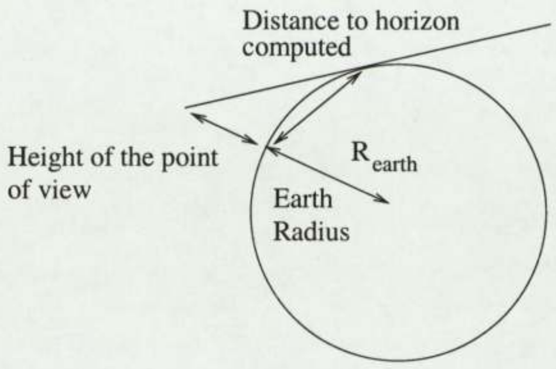


Figure 4.3: Distance to horizon.

Using the distance to horizon and the fact that every pixel is representing a surface seen under the same unitary angle $\delta\theta$ (see figure 4.4), the depth of the pixel on the i^{th} line can be approximated with:

$$\alpha = \tan^{-1}(\frac{h}{D_{First}}) \tag{4.24}$$

$$\beta = \tan^{-1}(\frac{h}{D_{Horizon}}) \tag{4.25}$$

$$\delta\theta = \frac{\alpha - \beta}{N} \tag{4.26}$$

$$Depth_i = h\ \tan(\alpha + i\delta\theta) \tag{4.27}$$

where N is the vertical resolution of the frame (number of pixels) and D_{First} the distance to the lowest pixel appearing on a frame ($D_{First} \approx 30$m).

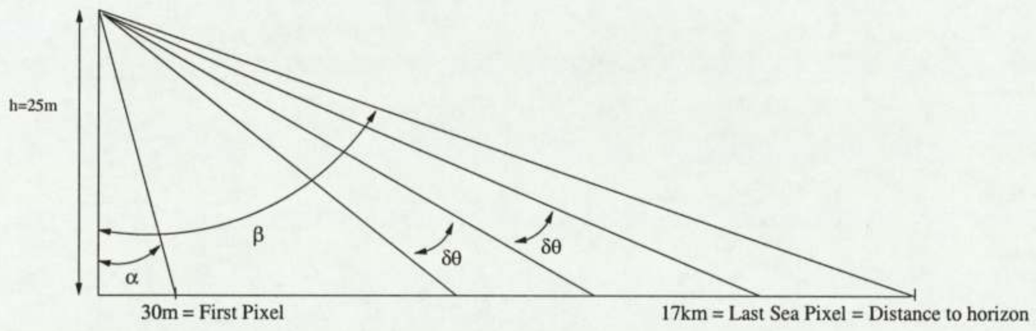


Figure 4.4: Scheme to approximate a pixel depth.

- Width: this approximation was more difficult to achieve. The technique used was to observe a few frames containing objects that we can estimate the real size, such as boats and tankers, and to compute the corresponding real width of a pixel (see table 4.1).
  With a few approximations, we extrapolate the widths of a complete frame by a linear approximation (see figure 4.5).

Object	Object Height (pixel height on a frame)	Estimated Real Size (m)	Size (in pixels)	Resolution (cm/pixel)
Tanker	476	150	26	576
Tanker	461	150	90	166
Boat	406	5	17	30
Boat	386	5	27	18
Boat	126	5	35	5

Table 4.1: Objects used to retrieve the width of a pixel depending on its height.

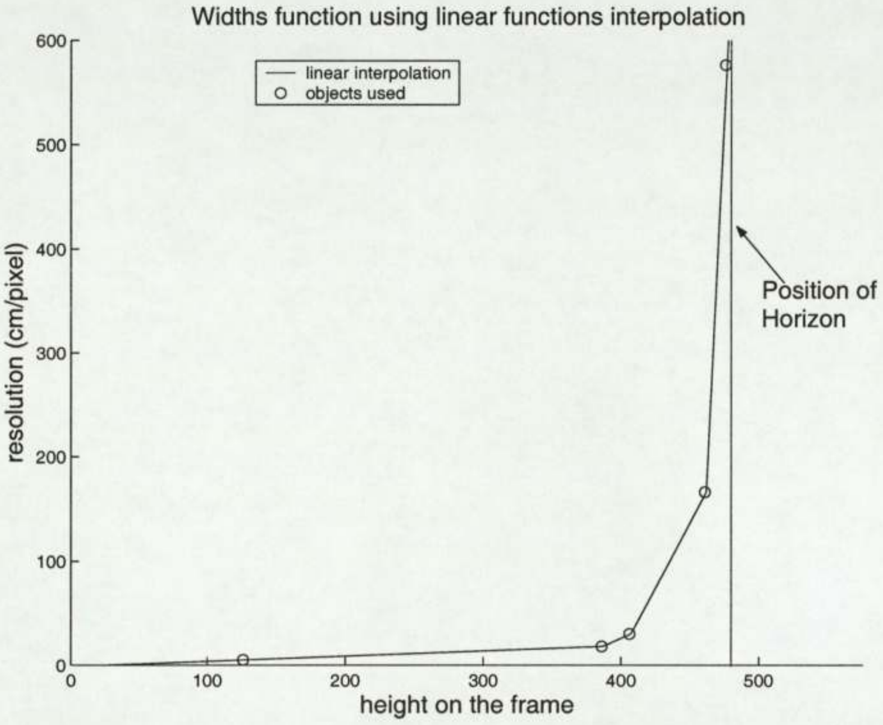


Figure 4.5: Extrapolation of the width function using linear functions.

Once the widths and depths are retrieved depending on the height of the pixel hi, the real surface S_{hi} represented by a pixel was approximated by a square:

$$S_{hi} = Depth_{hi} \times Width_{hi} \tag{4.28}$$

The matrix $Area$ is defined in terms of the coefficients $Area_{(i,j)} = S_{hi}$, so that lines are composed of identical coefficients of the surface at each height.
Giving the shape of the noise covariance matrices (in $\frac{1}{Area}$) and with respect to the models for Σ_t^ϵ and Σ_t^η (equations (4.22) and (4.23)), the coefficients k_1 and k_2 will allow us to direct the filtering in different ways:

- by choosing $k_1 = k_2$, the uncertainties on the process evolution and the observation evolution are equal. The observation is assumed to be as important as the previous estimated state in the new estimation achieved.
- by increasing k_1 so that $k_1 > k_2$, the uncertainty on the observation evolution is

similarly assumed to be more important than the one on the process evolution. We trust more the state estimation than the new observation.

- by decreasing k_1 so that $k_1 < k_2$, the new observation is considered more reliable than the state estimation.

Coefficients k_1 and k_2 were empirically chosen so that they are included in the interval $[1, 10]$.

4.3 Results

The Kalman filter was run on two different samples of 60 pictures. The results presented were taken so that the number of iterations (represented by the time t) was high enough not to be altered by the initialisation ($t > 10$).
It is important to note that the values chosen for k_1 and k_2 can be modified. In this implementation, the Kalman filter is more sensitive to the ratio between the values than the proper values. Using different values, so that $\frac{k_1}{k_2}$ remains unchanged, gives the same results.
Considering different frames segmented, the posterior probability retrieved at each time was updated using the next probability observed.
In the following pages, we present the results obtained for each of the three settings for k_1 and k_2.

4.3.1 Smoothing the posterior

The behaviour of the Kalman filter was clearly different depending on the setting applied for k_1 and k_2. Mainly, after a few iterations, the Kalman filter gives a smooth estimation of the true probability. This general behaviour is very sensitive to the chosen setting:

- with $k_1 = k_2$: the behaviour is a general smoothing of the segmentation process (Figure 4.7). The estimated posterior is mainly a smooth mapping of the posterior observed, where the isolated pixels, classified as oil slick pixels, are now classified as clear sea pixels.

- with $k_1 > k_2$: the smoothing turns out to be more important. The figure 4.8 was obtained for $k_1 = 8$ and $k_2 = 2$.

- with $k_1 < k_2$: the smoothing is lighter than with the previous settings, mainly due to the stronger importance given to posterior observed (Figure 4.9 obtained with $k_1 = 2$ and $k_2 = 8$).

Can this smoothing really be considered as an enhancement of the segmentation process?
First, the smoothing is a general effect that tends to level the posterior probability on a frame. With the segmentation technique used (select a pixel when its posterior is higher than 0.5), the smoothing implies classifying fewer pixels as slicks. For example, if a frame contains bright clusters clearly segmented (top right image in Figure 4.11),

the filter smooths the general posterior (especially with $k_1 > k_2$), making the boundaries of the clusters less accurate than before (low right image in Figure 4.11). The smoothing implies a loss of precision.
However, these effects have to be balanced considering frames where segmentation really needs enhancement (top left frame in Figure 4.7). Indeed, the smoothing removes significantly isolated pixels, making clusters appear more clearly. This effect can correspond to a real improvement of the segmentation technique as the oil slicks are usually continuous extended objects. They are mainly present on the surface as clusters (such as the layers of petrol on the sea when a tanker sinks).
Moreover, an interesting particularity of the smoothing is its disparity between pixels. Beyond the general smoothing effect, the pixels in the background seem less affected by the levelling than the ones in the foreground. This may result from the noise covariance matrices chosen. In the tests realised, the Kalman filter appeared sensitive to the matrix Σ_t^ϵ. This matrix represents the profile of the "importance" given to a posterior observed, with respect to its position on the frame. In the current implementation, the profile (in $\frac{1}{Area}$) gives a greater uncertainty for pixels standing in the background than the ones standing in the foreground. In the test using a uniform uncertainty (Figures 4.12 and 4.13), the smoothing seems uniform, making the upper pixels, classified as slicks, disappear.
Similar results were obtained with matrix Σ_t^η.
These tests reveal that the filter is sensitive to the noise covariance value, whose setting appears fairly sensible according to the results obtained. Indeed, with the distortion effects of real space in pixel space, oil slick clusters are represented with a smaller number of pixels comparing than the ones in the foreground. The setting applied to noise covariance matrices compensates efficiently for the distortion effects.
Concerning the parameters k_1 and k_2, the choice for the setting $k_1 > k_2$, or $\frac{k_1}{k_2} > 1$ relies mainly on the efficient suppression of isolated slick pixels. However, as for noise covariances matrices, the filter is very sensitive to the value chosen for both parameters.

4.3.2 Issues

The setting chosen for the filter appears difficult to calibrate. The main issue remains to find a suitable value for the ratio $\frac{k_1}{k_2}$.
By increasing it massively, the observation will be completely neglected. As already mentioned in the previous section, the new estimation will then mostly rely on the previous estimate. This appeared in the results as cases of loss of precision in the segmentation (Figure 4.11) or cases of wrong innovations appearing due to the fact that the previous estimation does not match the new frame (Figures 4.15 and 4.16).
Moreover, with the values tried for k_1 and k_2, the human activity classified as slicks could not be removed. Usually, the posterior corresponding to the human activity pixels is strong enough not to be suppressed by the smoothing (Figure 4.18). A solution could be to increase significantly the ratio $\frac{k_1}{k_2}$, which is not suitable for the reasons explained before.

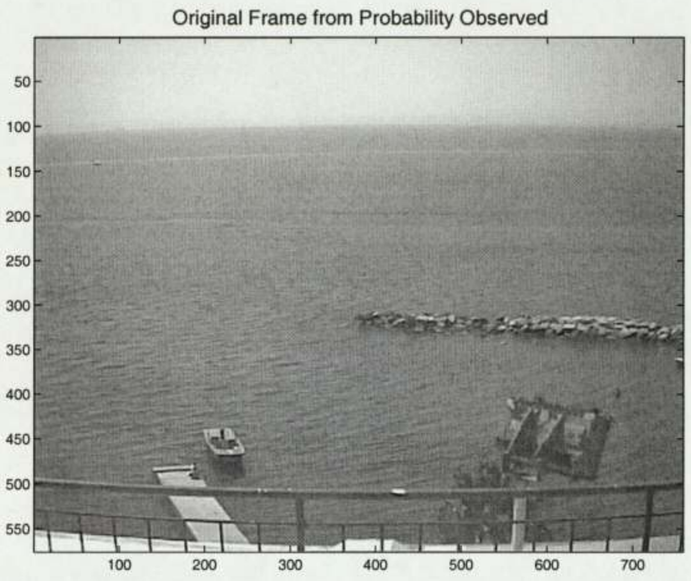


Figure 4.6: Frame observed at time $t + 1$.

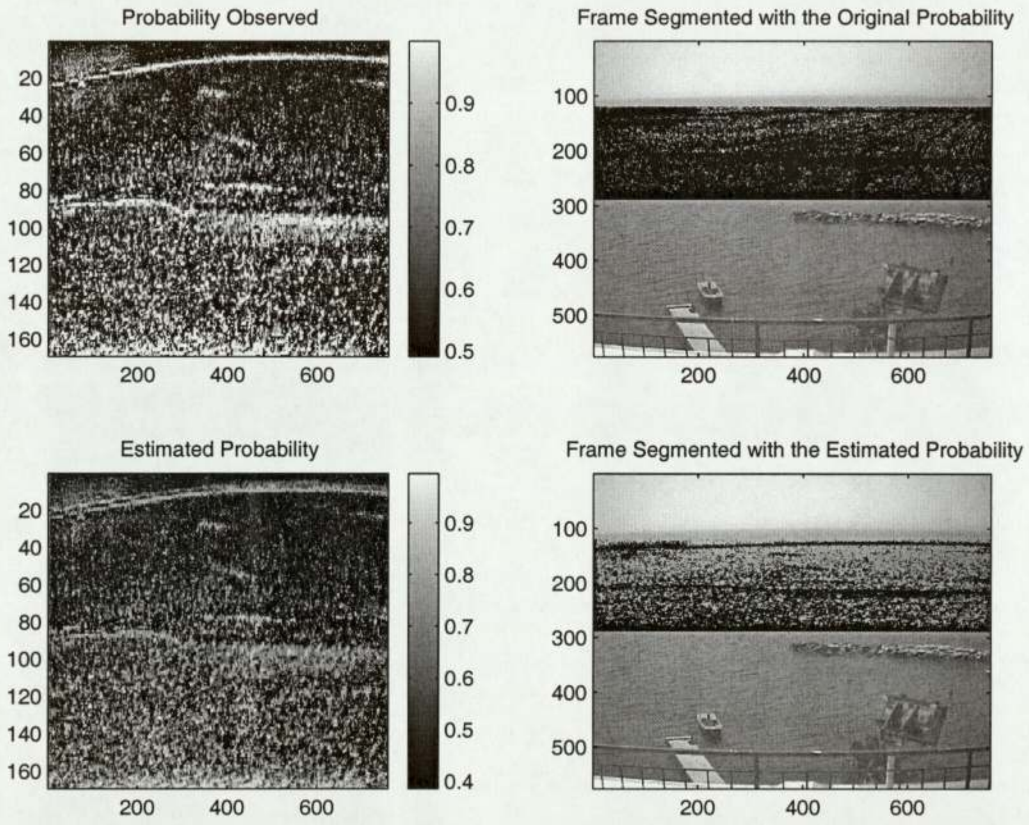


Figure 4.7: Original segmentation (up) compare to Kalman estimation (down) based on the new observation: Smoothing of the posterior probability.

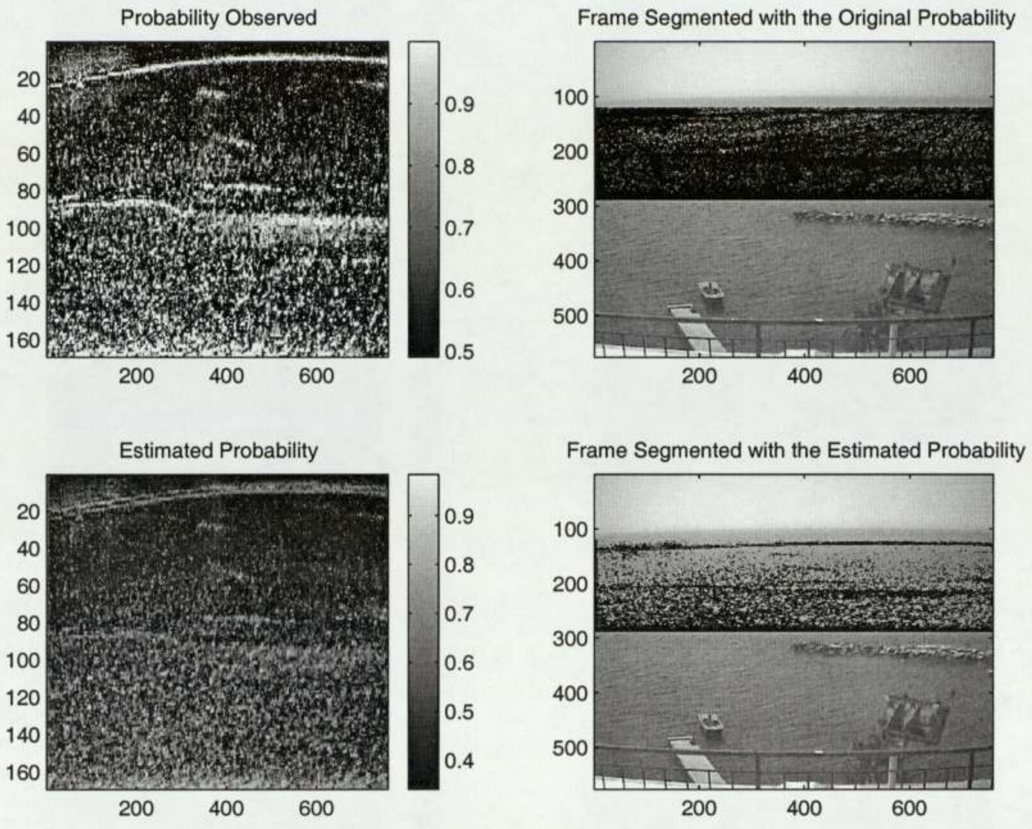


Figure 4.8: Strengthening of smoothing with $k_1 > k_2$.

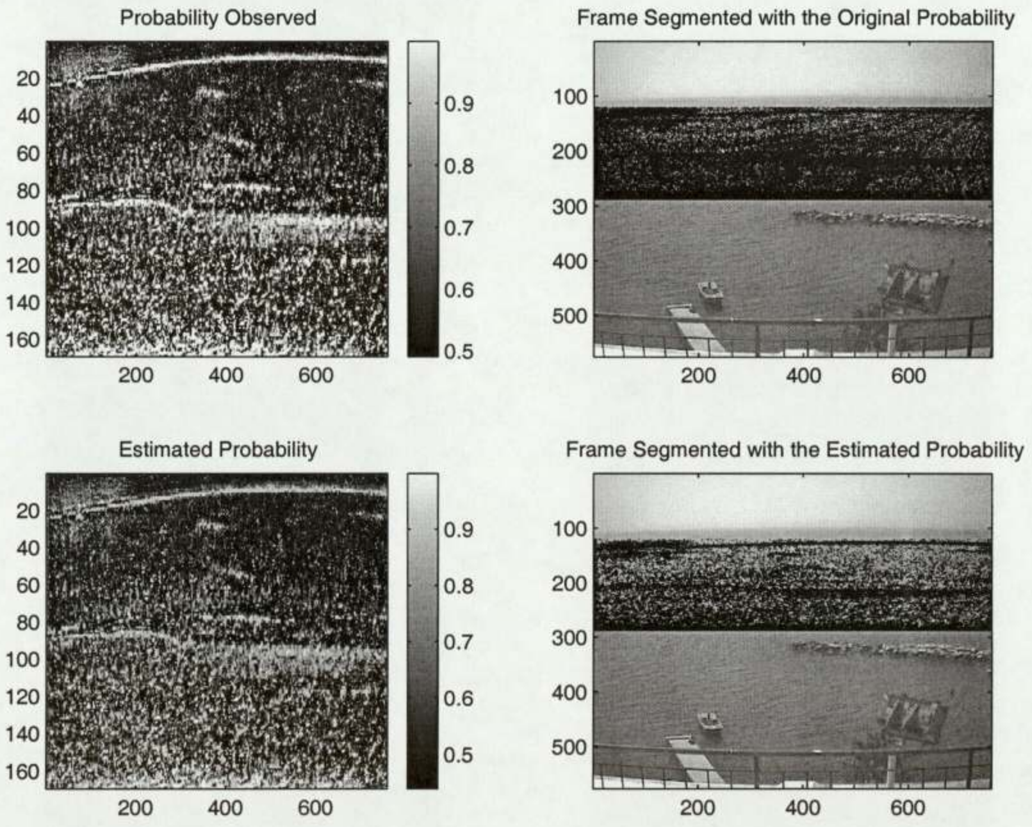


Figure 4.9: The posterior observed governs the estimation of the new state with $k_1 < k_2$.

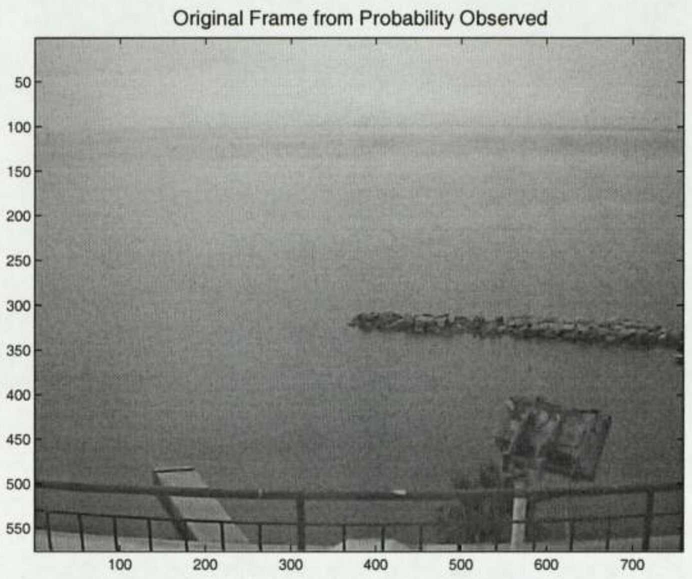


Figure 4.10: Frame observed at time $t+1$.

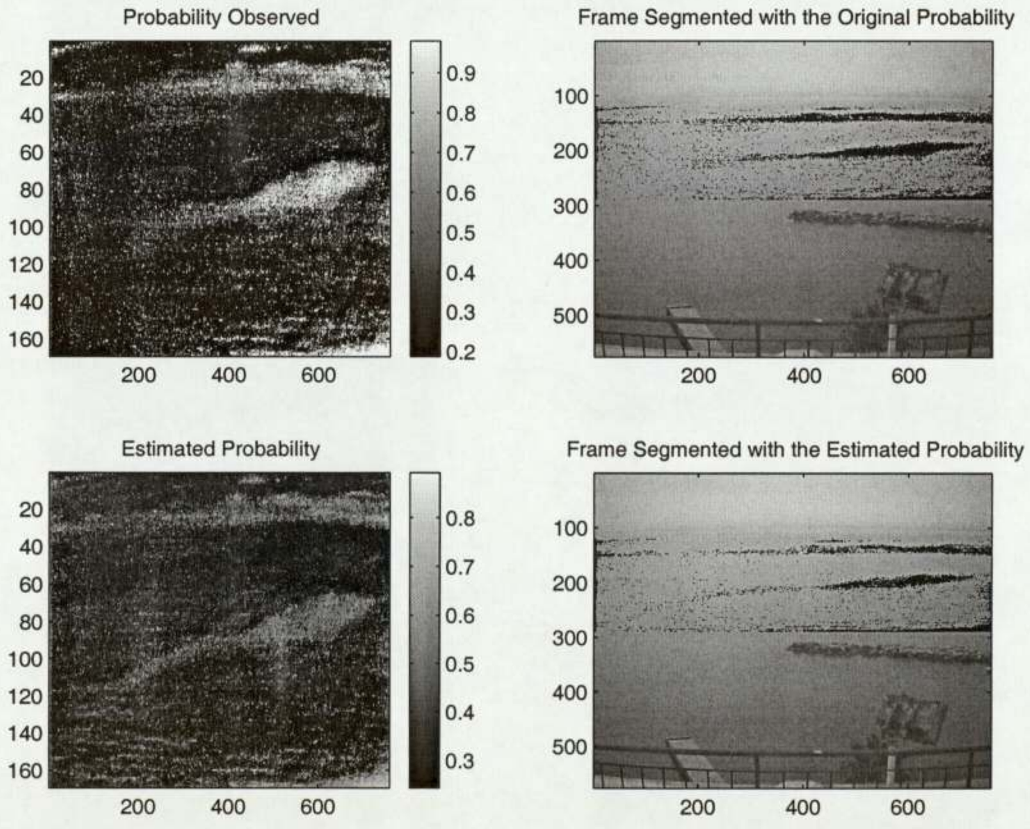


Figure 4.11: Loss of precision due to the smoothing.

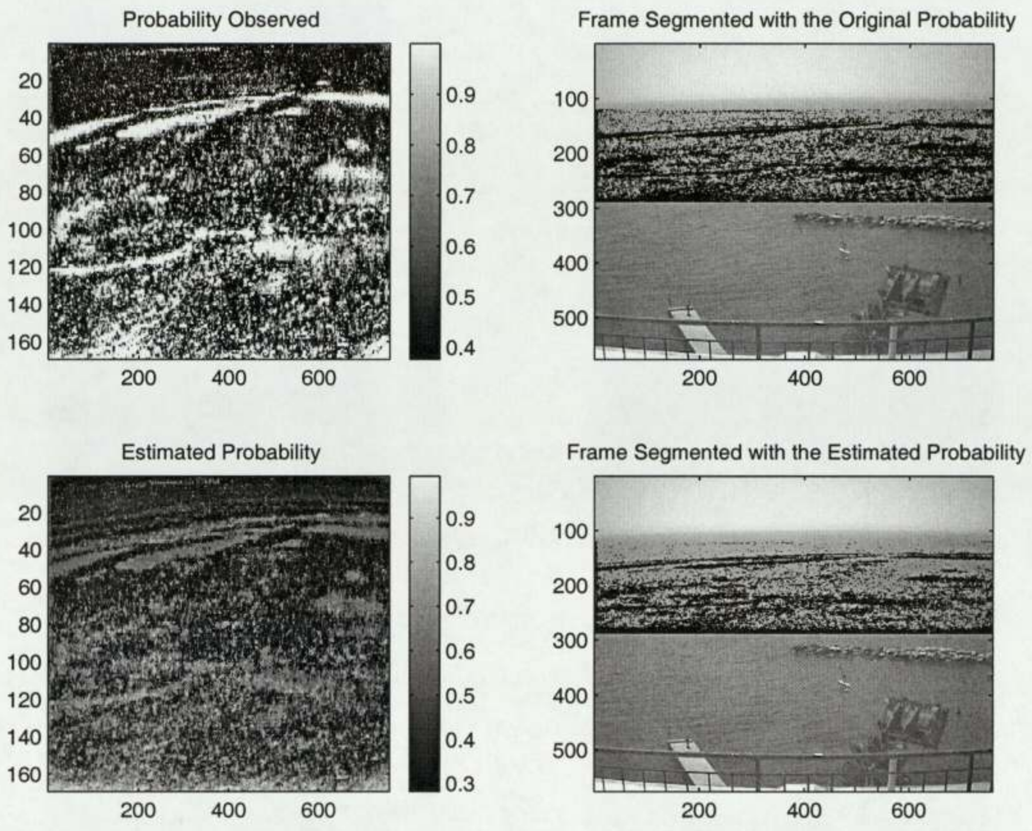


Figure 4.12: With a noise covariance based on the surface for the observation.

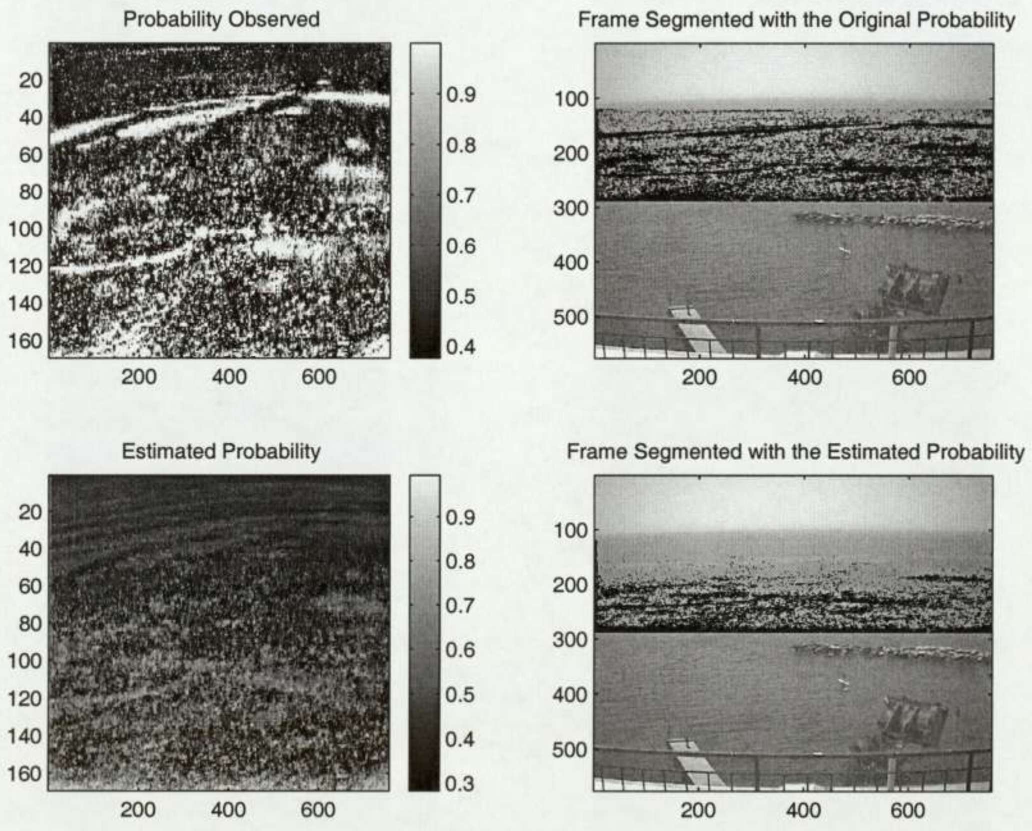


Figure 4.13: With a noise covariance uniform for the observation.

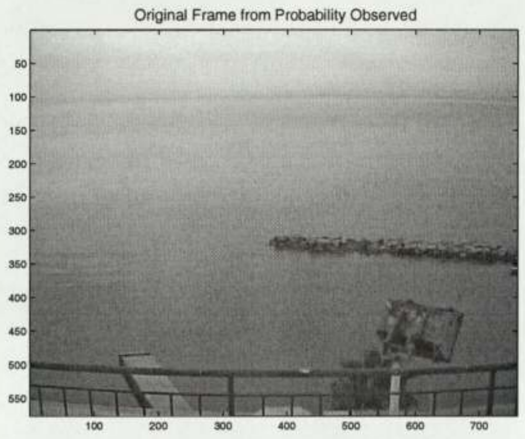


Figure 4.14: Frame observed at $t+1$.

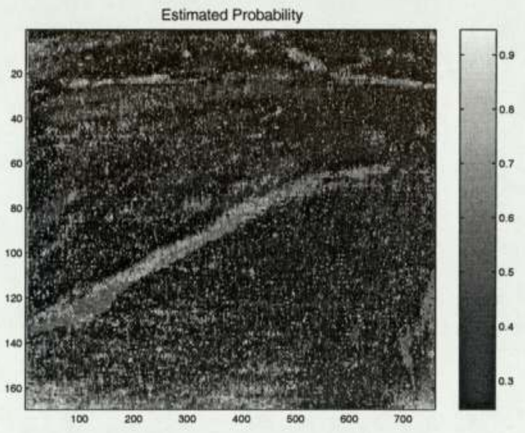


Figure 4.15: Posterior estimated at $t-1$.

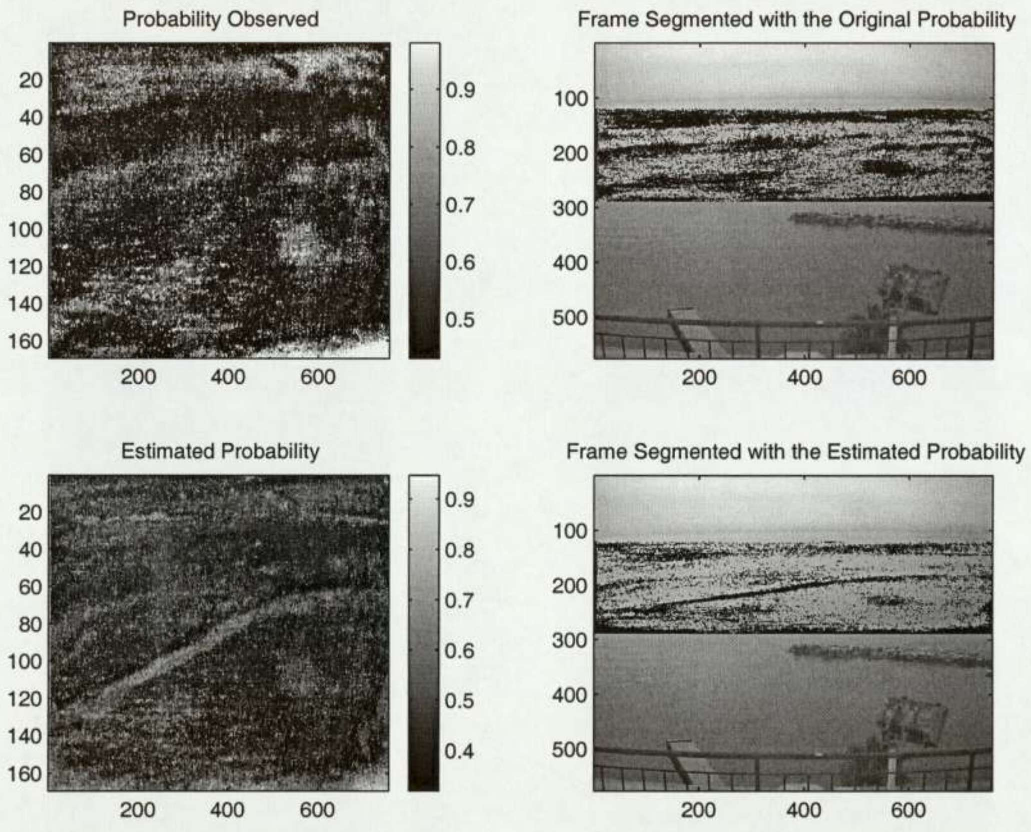


Figure 4.16: Wrong innovations.

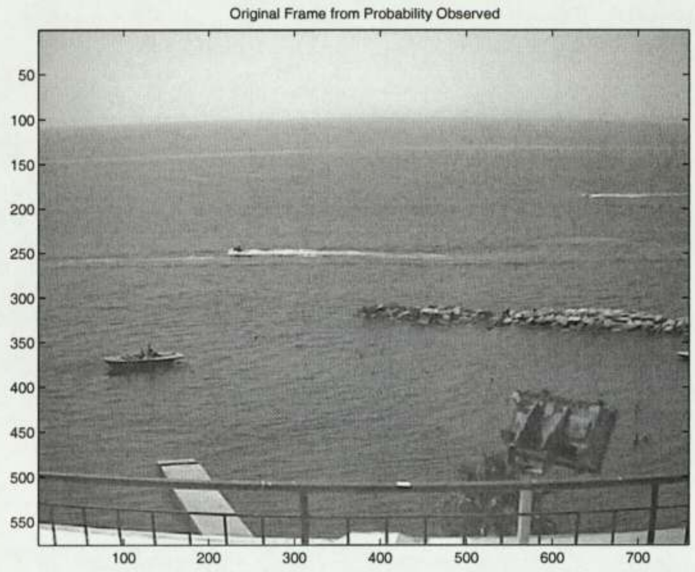


Figure 4.17: Frame observed at time $t + 1$.

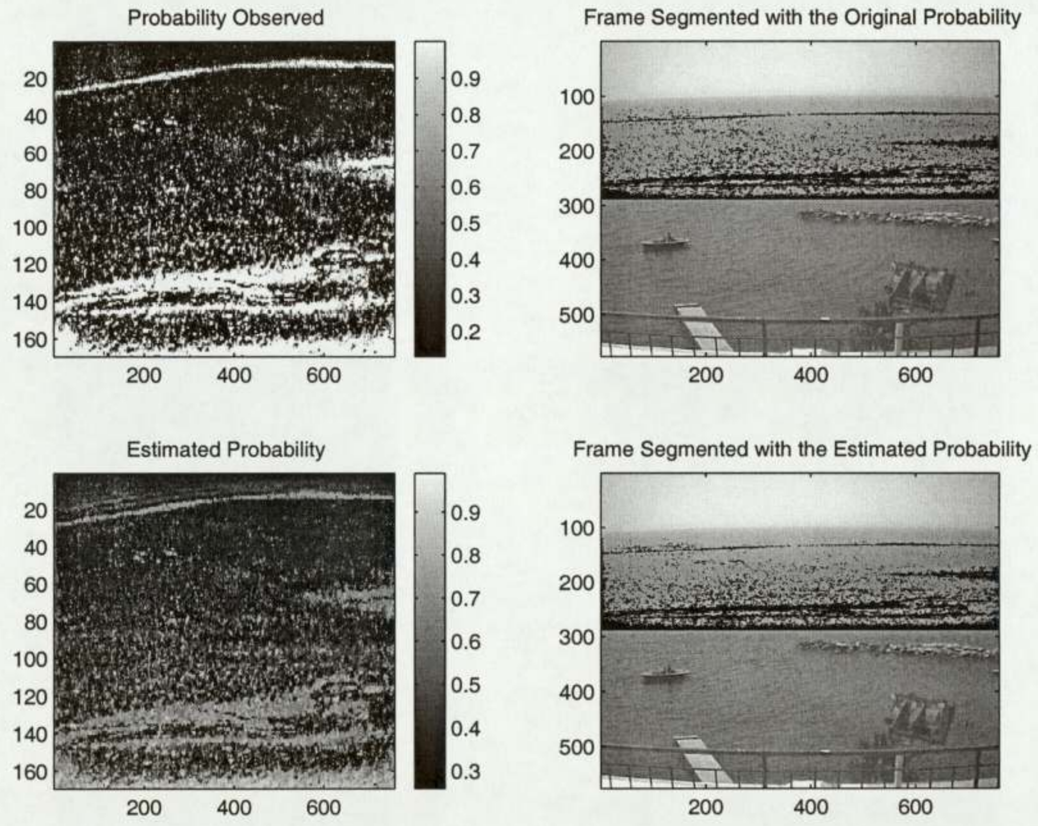


Figure 4.18: Failure to remove human activity.

4.4 Summary

The results from the Kalman filter are generally disappointing. The Kalman filter was implemented as an intention for a model to remove human activity from segmentation which failed with the implementation made.
The reasons for implementing a Kalman filter were based on assumptions of linearity in two places: linear observable and dynamics, and linear updates of the state following a new measurement. If the frame space dynamics really are close to linear, then a well-specified Kalman filter would help in removing activities which appear as nonlinear, due to the frame rate, such as human activity for example.
In the results obtained, the Kalman filter was sensitive. The choice between giving more importance to the new observation or the current estimated state (by choosing the appropriate ratio $\frac{k_1}{k_2}$) ended with a solution in which the posterior is smoothed significantly.
The main advantage of this setting is the suppression of isolated pixels classified as slicks. The main drawback is the loss of precision in the segmentation of oil slick clusters. In extreme values for the ratio, erroneous results are obtained where the new estimation corresponds to the previous estimated posterior without giving importance to the new posterior observed.
Moreover, the Kalman filter gives poor results in updating frames containing human activity. These poor results may come from the very crude model implemented. Indeed, in Kalman filtering, the best estimate given is in respect of a linear evolution of observations and dynamics. In our case, the dynamics were unknown and the filtering was driven only by the noises.
In this implementation, the noise sources were assumed to be additive and Gaussian with covariance matrices in respect of the real surface represented by a pixel. Even if the noise covariance matrices implemented appeared as sensible and suitable, these noises may not be Gaussian or additive. One step to make this model evolve would be to gather more information about dynamics and build a model for motion of slicks across the ocean. Such a model seems hard to build according to the data given, that consists only of ocean frames. Some extra information, such as meteorological information, will be needed to build and test a reliable model.
Moreover, the failure of the Kalman filter may be due to the linear assumption of a process that is known to be much more complicated. Even with good estimates of the evolution matrix, the system may be driven by non linear equations. To enhance the segmentation, the time model may be built from the basis of a complex dynamic model taking into account both the sea dynamics and surface wind dynamics.
In conclusion, although there should be useful exploitable information in the dynamics to modify the probability map to produce a more accurate segmentation, the specific model and assumptions made here do not work efficiently. On the timescale of this project, we were unable to investigate this issue further, and it should be considered as a worthwhile topic for future research.

Chapter 5

Conclusions

In this thesis, we were interested in processing real world images for detecting near-shore water-borne pollution (oil slicks).

We used a stochastic model where each pixel was allocated a probability of being generated by a "slick" model. However, first, a major problem of nonuniform illumination corruption had to be compensated for.

Every image was preprocessed in order to remove strong illumination effects due to the sun. The preprocessing of an image was achieved using two different techniques.
The basic technique, called homomorphic filtering, is based on spectral properties of the illumination. The low frequency part of the spectrum of the image is assumed to be dominated by the illumination component. The homomorphic filter is a high-pass filter whose aim is to separate illumination (low-frequencies) from reflectance (high frequencies). The homomorphic filter is really effective by empirical tuning of 3 parameters only if the spectrum is clearly disjoint. Moreover, there is no guarantee that the same filter will work across different times of day or seasons. It is not adaptive.
The second model was built following the need for an adaptive model. A RBF network is set with centres around the pictures and trained on the image intensity. The smooth fitting retrieved (mixture between number of hidden units and activation function) is taken for the illumination as the centres act as virtual light sources. The main advantage of this model to the homomorphic filter is its adaptiveness. Processing pictures from different times of day or season with the same filter still gives good results. Moreover, even if no error to the true value can be computed, this filter can be considered as more accurate, in the reflectance image retrieved, in the sense that there are no edge effects appearing as in the homomorphic filter.
The final choice for the RBF model was driven by its high adaptiveness and high flexibility (quality governed by number of hidden units).

The preprocessing ends by providing a reflectance image which enhances the "oil slicks" prints. The image was processed to separate oil slick pixels from other pixels.
The processing was achieved using a 2 Gaussian mixture model whose aim is to model two populations from a frame. By selecting the "oil slicks" source with an appropriate method, every pixel can be classified using the posterior probability. Segmentation was achieved using a constraint on the posterior probability (most probable source).

Results on images containing slicks were good, allowing us to identify and locate accurately a slick.
However, one of the main issues remaining from segmentation is the human activity noise that can be randomly fitted as a new source, our model failing naturally to model 3 different populations with only 2 clusters.

To enhance the processing, a simple Kalman filter was tried on the posterior probability. Given only mean and covariance of noises, the Kalman filter is the best linear estimator. Using the frame rate which provides us with regular observations, a Kalman filter was implemented to update the inaccessible true state of the posterior probability mapping.
Even if the slick dynamics in the ocean are governed by sea dynamics, wind dynamics and the geography of the site, the idea was to consider that the posterior probability mapping from a frame should evolve linearly with additive Gaussian noises in which covariances are determined by using surface pixel values to fit the data.
The results given by the Kalman filter trained on different sets are average. The optimal setting found achieves a smoothing of the posterior probability (filtering mostly based on the current state estimation to predict the new step). This smoothing tends to suppress isolated slick pixels, making the segmentation more accurate relatively to the spatial continuity of oil slicks (slicks appear as clusters). However, this filtering implies a loss of precision in which the boundaries of the clusters retrieved are less accurate than in the original observation. Moreover, the human activity mis-classified pixels could not be suppressed with a suitable setting (extreme values for $\frac{k_1}{k_2}$ increase dramatically the number wrong of innovations). This may be due to the very crude model implemented which did not take into account any evolution matrix. Moreover, the filter relies on assumptions of linearities that may be wrong.

Even if the dynamic filter implemented did not give good results, we still believe that the time information can help in enhancing the segmentation process. Future work could be orientated in gathering information about slick dynamics in order to build evolution matrices that may enhance the Kalman filter results. Other fields of research could be the implementation of non linear models, such as hidden Markov model using the discrete indicator matrix for coloration of a frame, or to build a full model for slick dynamics.
Moreover, in the thesis, we have considered the slicks as a group of pixels individually segmented using their intensity. Due to their oil constitution, the slicks appear as spatially continuous objects. This information should be included in the future work. Filters based on clusters may enhance significantly the segmentation.

Bibliography

[1] Blue Water: Final Report (draft). Technical Report IST-1999-10388, November 2002.

[2] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3–26. Academic Press, 1978.

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

[4] N. Gershenfeld. *The Nature of Mathematical Modeling.* Cambridge University Press, 1999.

[5] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Prentice Hall International Editions, 2002.

[6] D. Lowe. Theory and applications of Kalman filtering, 2002.

[7] D. Lowe and I. Stainvas. Toward sea surface pollution detection from visible band images. *IEICE Transactions on Electronics*, E84-C(12), 2001.

[8] P. S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering.* Academic Press, 1979.

[9] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering.* Dekker, 1988.

[10] G.J. McLachlan and T. Krishnan. *The EM algorithm and Extensions.* Wiley, 1997.

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

[12] M. Petrou and P. Bosdogianni. *Image Processing, The Fundamentals.* John Wiley & Sons, 1999.

[13] P. Tino and L. Csato. The em algorithm, 2002.

[14] A. Webb. *Statistical Pattern Recognition.* Arnold, 1999.

[15] A. P. Youschkevitch. *Les Mathématiques Arabes (du VIIIè au XVè siècle).* Editions Vrin, 1976.

Index