Automated Oil Spill Detection with Ship Borne Radar

Nasser Mostafa Saleh
March, 2004

Automated oil spill detection with ship borne radar

By

Nasser Mostafa Saleh

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in Water Resources and Management specialising in Environmental System Analysis and Management.

Thesis Assessment Board

Chairman:	Prof. A.M.J. Meijerink
External examiner:	Ir. J.C.M. Kleijweg (TNO Physics and Electronics Laboratory)
First supervisor:	Dr. A.S.M. Gieske
Member:	Prof. J.L. van Genderen
Second supervisor:	Dr. Tjeerd Hobma

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION
ENSCHEDE, THE NETHERLANDS

I certify that although I may have conferred with others in preparing for this assignment, and drawn upon a range of sources cited in this work, the content of this thesis report is my original work.
Signed …….

Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

ABSTRACT

This research outlines the need for a real-time, a ship borne radar facility SHIRA for the tracking of nearby oil spills during oil containment and cleaning operations at sea with high quality imagery. SHIRA is imaging digital X-band radar, developed by TNO for the measurement of oceanographic features. It was especially adapted for the imaging of oil slicks by implementing a number of improvements, based on the outcome of a previous oil detection experiment. The main aspects that make SHIRA suitable for the current purpose are its high sensitivity and dynamic range, and its ability to integrate a large number of images of the same scene, after correction for platform motions.

Quantitative comparison has been made between various filters, which are able to reduce variance in homogeneous areas, preserve edges and lines, suppress point scatter, and preserve spatial variability, while avoiding artefacts. Gamma Filter yields the best smoothing and despeckling. Moreover, it seems to keep the edges without blurring the minimum. Eight textures are applied, based on the co-occurrence matrix. These textures include mean, variance, homogeneity, contrast, dissimilarity, entropy, second moment, and correlation. Co-occurrence measures use a gray-tone spatial dependence matrix to calculate texture values. This is a matrix of relative frequencies with which pixel values occur in two neighbouring processing windows separated by a specified distance and direction. It shows the number of occurrences of the relationship between a pixel and its specified neighbour.

The use of the correlation texture implies a scaling of the axes so that the features receive a unit variance. It prevents certain features from dominating the image because of their large value. Correlation texture performs best, in terms of the probability of oil occurrence because it allows clear discrimination between synthetic oil (oval-round shaped), natural oil (striped), water waves and water for data with high waves (dataset Prestige event, Spain) or with low-moderate waves (dataset with oil simulations in North Sea).

Non-Standardized Principal Component Analysis can discriminate between oil and water regardless of wind speed. To avoid the disadvantage of the use of linear combinations that lead to a high correlation of the resulting textures and loosing textural information, Principal Component Analysis has been applied. The first principal component of covariance matrix, i.e. the eigenvector with the largest Eigen value corresponds to an identifiable physical property, which is most important for interpretation of the patterns. The higher order Non-Standardized Principal Component Analysis and Correlation methods produce the same results in this case.

ACKNOWLEDGEMENT

I am grateful to Dr. A.S.M. Gieske (first supervisor), Dr. Tjeerd Hobma (second supervisor) Department of Water Resources and Environmental Management, Ir. J.C.M. Kleijweg, at TNO Physics and Electronics Laboratory for supervising my thesis. I acknowledge their encouragement and support in my project. The discussions with them at various stages of the work proved to be of exceptional help. I sincerely thank them for their suggestions and critical comments to bring further improvement in this project and in the presentation of the information in the best form of visualization; this has definitely given an enriching blend to the whole work.

I would like to extend my gratitude to my parents, my aunt Nadia, my wife and her family.

LIST OF FIGURES

LIST OF TABLES

TABLE OF APPENDICES

TABLE OF CONTENTS

1. Introduction

1.1. Background

SHIRA is imaging digital X-band radar, developed by TNO for the measurement of oceanographic features and oil spill detection, which is the most common environmental problem in the sea. This study aims of building real-time capacity of SHIRA to detect oil spill as early warning system with improving the quality of the images without more stacking to avoid false detection. Many experiments have been carried out for testing and developing its system. A new algorithm is proposed in this work to reach our goal with high sensitivity. Three data sets are considered with different locations and wind speeds, 0955hh(low moderate waves- dataset North Sea event, Holland), Vlek3 and Vlek10 with high waves -dataset Prestige event, Spain.

The images are characterized by high speckle/noise. The gray-tone variation between features is not wide in all the three data sets 0955hh,Vlek3 and Vlek10, so according to (Haralick et al., 1973) if there's no fine texture result then filters should be applied first. We applied six spatial filters, to determine which one gives the best smoothing and despeckling and it seems to keep the edges best without blurring the minimum. The selection of best filter is done based on the results obtained through SNR.

One important property of tone-texture is the spatial pattern of the resolution cells composing each discrete tonal feature. When there is no spatial pattern and the gray-tone variation between features is wide, a fine texture results. As the spatial pattern becomes store definite and involves more and more resolution cells, a coarser texture results (Haralick et al., 1973).

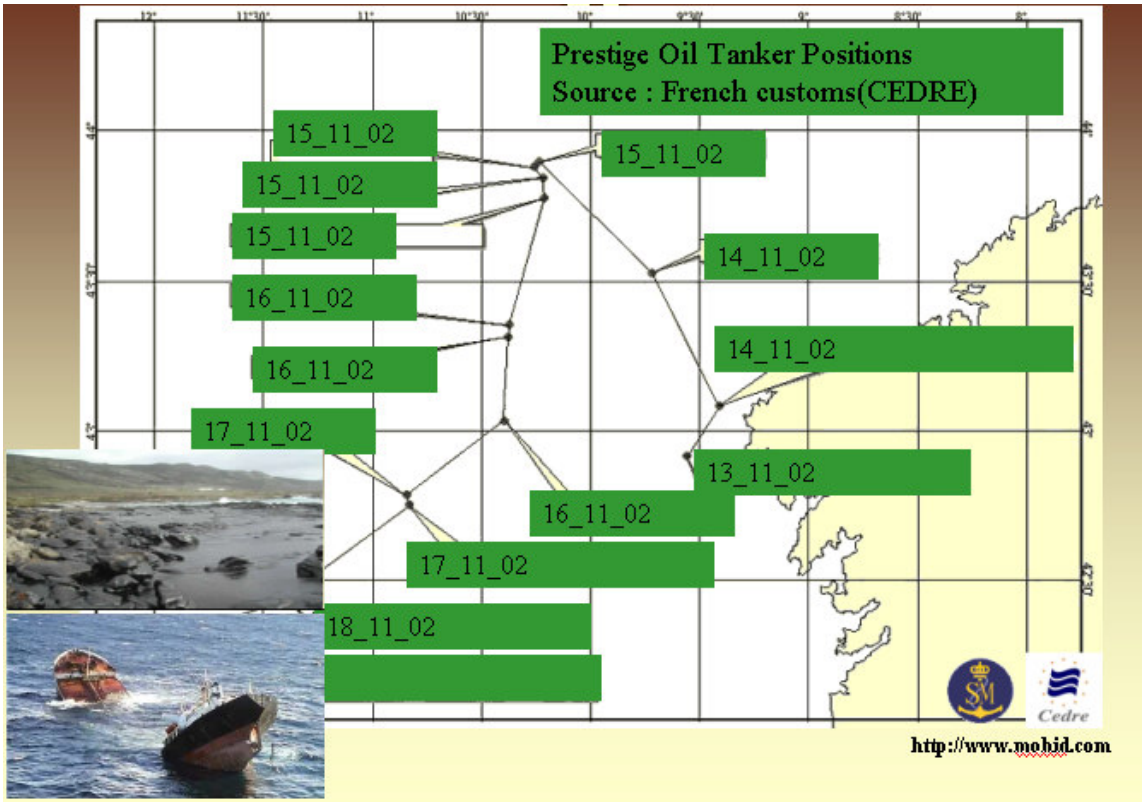


Figure (1-1) Prestige oil tanker positions accident Nov 2002 .

1.2. Research Problem

Three experiments have been carried out in 1993 in which SHIRA's performance was tested on artificial oil spills. The following recommendations were made:

- Improve the low-noise logarithmic amplifier to increase sensitivity and dynamic range.
- Digitization of backscatter values below the noise level.
- Improve the contrast optimisation and quality of the real-time display. the images are characterized with high speckle/noise because of the fact that the quality is poor at times depending on weather conditions and boat movement.

The first two of these improvements were implemented immediately (Kleijweg et al., 1994).

1.3. Objectives

Image quality can be improved by averaging images. However because of boat movement this doesn't always lead to a better result. The images are characterized with high speckle/noise because of the fact that the quality is poor at times depending on weather conditions and boat movement. There are several ways to reduce speckle, these include increasing the bandwidth of the radar system or by reconstructing pixels through the averaging of range-line returns in the along track direction (known as multi-looking), or stacking, several spatial filters designed to reduce the speckle in order to make data

more interpretable. Each one has its own unique algorithm. Then texture algorithm must be applied. The main objectives are:

- Improve the contrast optimisation and quality of the real-time display.
- Choose the best filter of several spatial filters to reduce the speckle.
- Identifying objects (oil) of interest in the image.
- Choose the best texture of the eight co-occurrence matrix that yield the best suitable method for oil spill detection using ship borne radar.
- Propose new method to extract all the information from the eight co-occurrence matrix.

1.4. Research Questions

- Which texture algorithm performs best, in terms of the probability of oil occurrence?

- Does the new algorithm allow clear discrimination between synthetic oil (oval-round shaped) and natural oil (striped)?

- How does the analysis method perform in a setting with high waves (dataset Prestige event, Spain)?

- How does the new algorithm perform in a setting with low-moderate waves (dataset with oil simulations in North Sea)?

1.5. Research Methodology

Figure (1-2) shows the main algorithm with the following steps:

- Step1: Image extraction and conversion it to polar then Range and angle correction for the ship motion.

- Step2: Conversion from polar to Cartesian images and average of 10 images or less.

- Step3: Adaptive filters were applied then Gamma filter was selected according to Signal to Noise Ratio.

- Step4: Co-occurrence matrix with eight textures was applied then correlation texture was selected according to (Haralick et al., 1973) to identify, describe and extract the oil, after which image features can be categorized (classified).

- Step5: Principal component analysis for texture fusion and evaluation of correlation texture as the best suitable method for oil spill detection using ship borne radar.

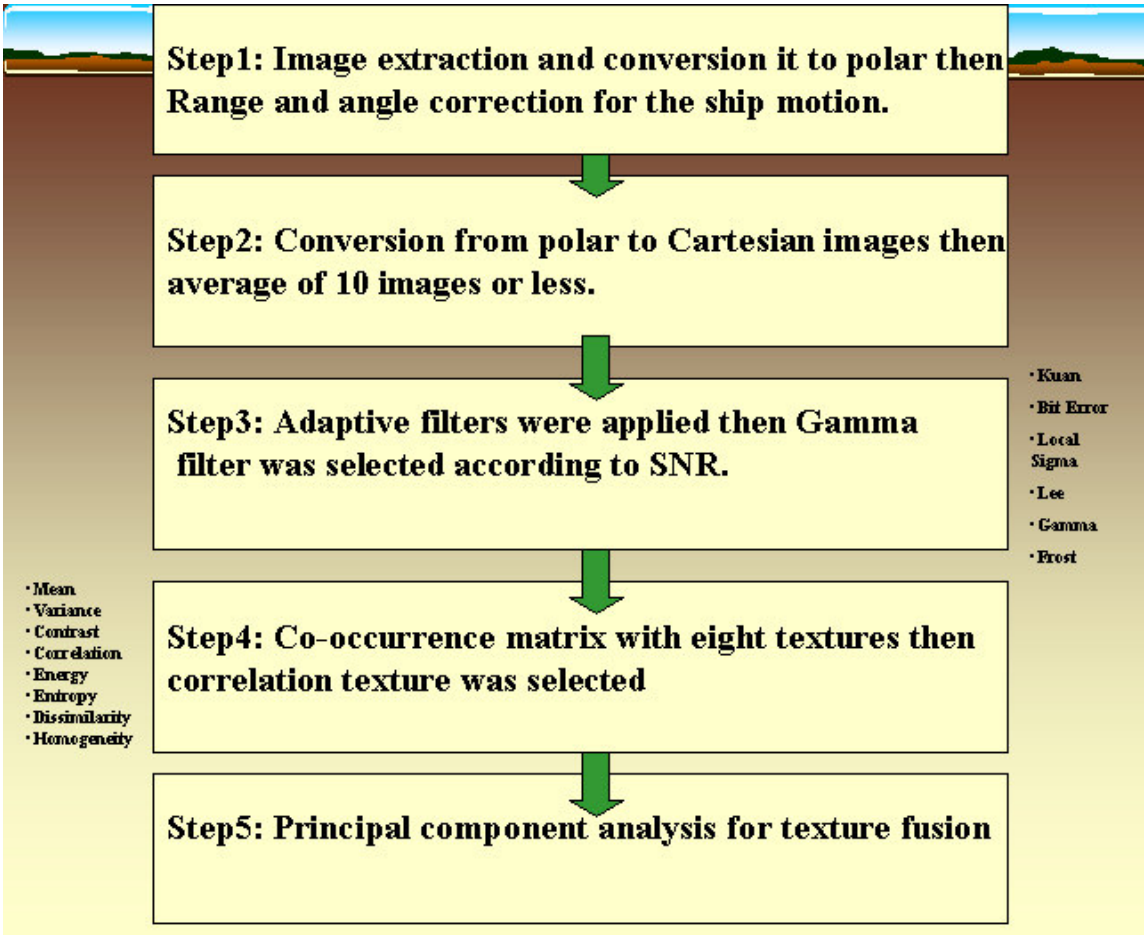


Figure (1-2) Flow chart showing summary of the algorithm

1.6. Previous Work

- Three experiments have been done in 1993 in which SHIRA's performance was tested on artificial oil spills. (Kleijweg et al., 1994).
- (Misset, 2003) proposed an algorithm consists of five steps:

Step 1: Data input
The angles of the shadows of the masts have to be indicated.

Step 2: Data to polar
In the second step the polar image is isolated from the data. This is rather tricky because the images do not all have the same number of lines. The angles range from 0 to 4096. The number of lines in an image ranges from 1800 to 1900. Here the lines have to be looked at, but some adjustments are needed. The raw data lines are put in the image and the data line is copied if the angles do not follow one another with a minimal difference of intensity. Through this the lines can be assigned to a matrix of 4096 by 1024.

Step 3: Polar to Cartesian, range and angle correction and Stacking

Polar images are converted to Cartesian images while range and angle corrections are made for the ship movement. After that the necessary adjustments are carried out and stored one by one. An average of images is calculated to improve the image quality and be able to detect oil.

Step 4: Detection Data

A threshold value is determined such that the objects in the binary image are classified in 'objects-of-interest' and 'irrelevant' objects.

Step 5: Detection dataF

In this final step the function "frequention" looks at the frequency with which an object returns when one looks back at a number of images. A logical number of images that is looked back at are for instance the same number that has been averaged. When an object has been detected these data are stored and forwarded to the function plotting. This function plots the image under study.

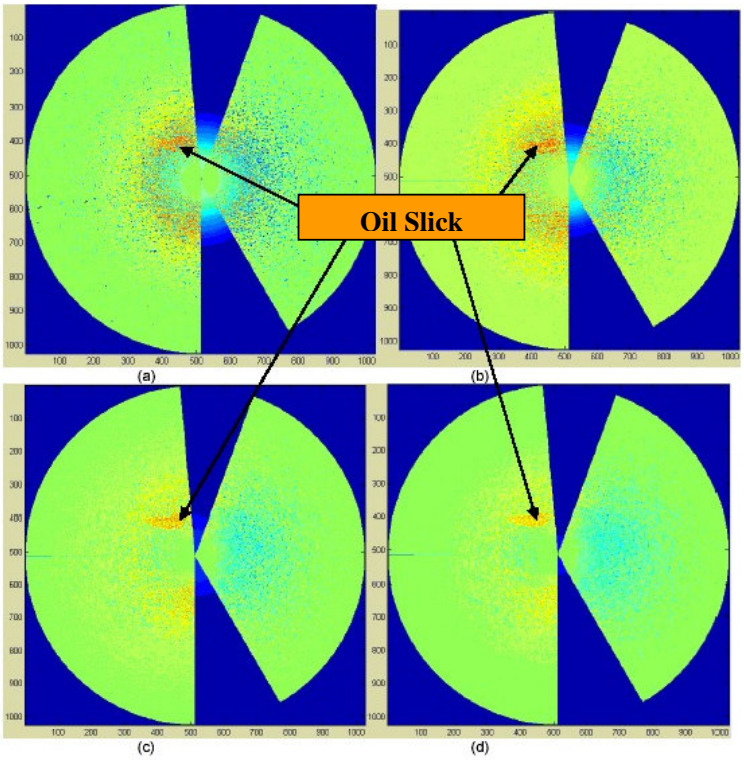


Figure (1-3) (a) Single image (b) two images averaged (c) average of 5 images (c) average of 10 images.

Note how the average of stacking influences the clarity of appearance.

1.7. About this thesis

In five chapters:

- In chapter 1, we have presented an introduction as a brief description of the work.

- In chapter 2, we present summary for SHIRA system.

- In chapter 3, we have presented the perspective and motivation for digital image processing and scene analysis and the requirement for different stages during this study.

- In chapter 4, the new algorithm and its data processing have been presented.

- In chapter 5, results and conclusions have been presented with both quantitative and qualitative analysis.

2. SHIRA Radar Shipborne as PPI Radar system

2.1. SHIRA Ship's Radar

The instrument is used to detect and monitor the following: sea surface features, Oil slicks, wave spectra, currents, water depth, and small targets at the surface. (TNO Physics and Electronics Laboratory, Internet site: http://www.tno.nl/instit/fel/os/fac/ra_fac_shira.html).

Figure (2-1) The SHIRA antenna located in the Netherlands. The antenna with a length of 2.7 m completes one revolution in 1.2 sec in imaging mode.

2.1.1. Monitoring the sea surface

Radar waves reflected from the sea surface are a valuable source of information. They are governed by the sea surface roughness, which, in turn, is determined by the wind, waves, oil-slick contamination, and even by topographic features of the sea bottom. SHIRA is a radar sensor that extracts this information. With SHIRA, it is possible to conduct continuous surveillance, producing time-series of images. SHIRA can be operated where any other form of continuous surveillance would be impossible or extremely costly. Time series of a fixed area are particularly useful for feature tracking and monitoring, for example for the evolution of oil slicks. Time series also contain valuable information about the statistical properties of the sea surface, making it possible to enhance the detection capability for small targets in the clutter contaminated environment of the sea surface.

2.1.2. The SHIRA system

SHIRA (Ship's Radar) is digital X-band radar that images the sea surface, with a rotating antenna and PC-based data processing capability. It is not only a relatively low-cost instrument, but also extremely versatile and cheap to run. It is a remote sensing instrument developed by the TNO Physics and Electronics Laboratory on the basis of a conventional ship's navigation radar. SHIRA, however, extracts information from the sea clutter, which in navigation radar is normally suppressed. SHIRA can be operated from a ship, a (production) platform or an elevated coastal site. At each antenna revolution, i.e. about once a second, SHIRA digitises and stores a radar image of an operator-specified area of the sea surface. Compensating for possible ship movements, SHIRA can aggregate many co-located images taken during successive antenna revolutions, in this way producing a time series of images.

Frequency	9.44 GHz X-Band
Antenna length	9 feet (2.74 m)
Polarization	HH / VV
Resolution in Range	7.5 / 3.75 m *
Resolution in Azimuth	0.7^{0} (12 m at 1 km)
Field of view in Range	< 2500 m
Field of view in Azimuth	$< 360^{o}$
Pulse width	1 revolution / 3.3 s 50 ns / 250 ns / 1 µs*
Peak power	25 kW
Spatial sampling grid	Polar
Digitization	8 bit
Rotation speed	1 revolution / 3.3 s

* Adjustable (Kleijweg et al., 1994).

Table (2-1) required navigation SHIRA radar system specifications

2.1.3. Applications of SHIRA

Information about waves, currents, slicks, sea bed form and the detection of small objects are obtained by applying different processing techniques on the time series of co-located images. The individual images contain information about the two-dimensional spatial structure of the sea surface, and the time series about its temporal behaviour.

SHIRA was originally developed to measure ocean directional wave spectra, and derived quantities of current speed and water depth. The latter are possible because the water depth and current velocity determines the local velocity of the waves. In many respects, SHIRA actually performs much better in this regard than conventional wave and current instruments like buoys.

Because oil slicks produce a lower radar echo, they show up in radar images as dark patches. This effect is already exploited by airborne and satellite radar (SLAR and SAR), such as carried by the Netherlands Coast Guard aircraft and the European ERS satellites. The lower sensitivity of radar for shal-

lower aspect angle is, in SHIRA, more than compensated for by its higher integrating capabilities. SHIRA has proved itself to be a useful tool for monitoring the evolution of oil slicks. With SHIRA it is also possible to measure sea bottom topography. Bottom topography in shallow seas is, under certain hydro-meteo conditions, visible on radar images. At the TNO Physics and Electronics Laboratory this phenomenon is already exploited to derive bathymetric maps from space- and airborne SAR. Placed on a shore-based platform, SHIRA can be used for studying the dynamics of wave breaking in the surf zone. Detailed knowledge of processes in the surf zone is relevant for morphological modelling, and therefore of interest for coastal management and protection. Finally, SHIRA can be used to detect small nearby objects, the backscatter signature of which is normally lost in the clutter. The application of filtering techniques, based on information derived from prior images, results in a greatly increased detection capability for small, nearby targets.

2.1.4. Research and experiments

In 1990 and 1991 SHIRA participated in ESA campaigns for calibration of ERS-1 wave products. Furthermore, SHIRA participated in the NATO MILOC Rocky Road survey in 1993. During these experiments SHIRA proved to deliver reliable wave spectra for winds force 2 to 12 Beaufort, and also demonstrated that its derived wave direction is independent of the radar look angle with respect of the wind or wave direction. In 1992 and 1993 SHIRA participated in the NETHGER Oil Spill Experiments.

Development of SHIRA at the TNO Physics and Electronics Laboratory is currently concentrating on expanding the hardware and storage capabilities of the system. The increased data rate enables surveillance of larger areas and will lead to an increased detection possibility for small targets in highly clutter contaminated environments.

Further research is going on to distinguish oil slicks from uncontaminated water using the spatio-temporal behaviour of the radar sea clutter. This must lead to automation of oil spill detection in order to minimise the workload on human operators.

Another development is towards integrating wavelength and direction measurements from SHIRA with wave height measurements from Doppler radar and with ship motion measurements from accelerometers. This synergy of sensors will result in an accurate description of the wave spectrum including absolute wave height. Such information is of great importance during the transportation of large structures.

3. Digital Image processing

3.1. Radar image speckle noise and a adaptive filters

Adaptive filtering uses the standard deviation of those pixels within a local box surrounding each pixel to calculate a new pixel value. Typically, the original pixel value is replaced with a new value calculated based on the surrounding valid pixels (those that satisfy the standard deviation criteria). Unlike a typical low-pass smoothing filter, the adaptive filters preserve image sharpness and detail while suppressing noise.

There are several ways to reduce speckle, these include increasing the bandwidth of the radar system or by reconstructing pixels through the averaging of range-line returns in the along track direction (known as multi-looking). Unfortunately, these methods are applied during the image formation from signal data, and cannot be controlled by the analyst. But the analyst is provided with several spatial filters designed to reduce the speckle in order to make data more interpretable. Each of these filters has unique algorithm. One should be aware of the fact that filtering tends to not only remove the speckle but also reduces the textural information inherent in radar data.

3.1.1. LEE FILTER (Lee, 1980)

Use Lee filters to smooth noisy (speckled) data that have intensity related to the image scene and that also have an additive and/or multiplicative component. Lee filtering is a standard deviation based (sigma) filter that filters data based on statistics calculated within individual filter windows. Unlike a typical low-pass smoothing filter, the Lee filter and other similar sigma filters preserve image sharpness and detail while suppressing noise. The pixel being filtered is replaced by a value calculated using the surrounding pixels. Speckle noise in radar images is generally assumed a multiplicative error model. Then

$$\bar{i}(x, y) = I(x, y)\omega(x, y) + \bar{I}(x, y)(1 - \omega(x, y) \quad (1)$$

Where ω is the weighting function.

$$\omega(x, y) = 1 - \frac{C_{si}^2}{C_i^2(x, y)} \quad (2)$$

$$C(x, y) = 1 - \frac{\sigma(x, y)}{\bar{I}(x, y)} \quad (3)$$

Where $C_{si,}$ is the speckle index and C_I is the ratio of the square root of local variance over the local mean, where I (x,y) is the pixel which will be filtered and $\bar{I}$ (x , y) is the output pixel,
σ = Standard deviation of intensity within window.
ω = weight for each pixel.

3.1.2. FROST FILTER (Frost et al., 1982)

Performs Frost adaptive filtering to any type of image data. The Frost filter is an adaptive filtering algorithm. It uses an exponentially damped convolution kernel, which adapts itself to features based on local statistics. The Frost filter differs from the Lee and Kuan filters with respect that the scene reflectivity is estimated by convolving the observed image with the impulse response of the SAR system. The impulse response of the SAR system is obtained by minimizing the mean square error between the observed image and the scene reflectivity model, which is assumed to be an autoregressive process.

Use Frost filters to reduce speckle while preserving edges in radar images. The Frost filter is an exponentially damped circularly symmetric filter that uses local statistics. The pixel being filtered is replaced with a value calculated based on the distance from the filter centre, the damping factor, and the local variance.

The resulting value R for the smoothed pixel is:

$$R = \sum_{i=1}^{n} P_i \times \omega_i \; / \; \sum_{i=1}^{n} \omega_i \qquad (4)$$

The implementation of this filter consists of defining a circularly symmetric filter with a set of weighting values M for each pixel:

$$\omega_i = EXP \quad (- A \times T_i) \qquad (5)$$

Where:

$$A = D \times (\sigma / \mu)^2 \qquad (6)$$

A=exponential damping factor.
μ = Mean value of intensity within window.
σ = standard deviation of intensity within window.
T_i = the absolute value of the pixel distance from the centre pixel to its neighbours in the filter window.
P_i = grey levels of each pixel in filter window.
ω = Weight for each pixel .

3.1.3. GAMMA FILTER (Lopes et al., 1990)

Kuan first proposed the Gamma filter. The scene reflectivity was assumed to be Gaussian distributed, however this is not quite realistic since it implicitly assumes a negative reflectivity. Lopes modified the Kuan Map Filter by assuming a gamma distributed scene and setting up two thresholds. To apply the MAP (Maximum A Posteriori) approach to speckle reduction, the a priori knowledge of the probability density function of the scene is required. With the assumption of a gamma distributed scene.
Use Gamma filters to reduce speckle while preserving edges in radar images. The Gamma filter is similar to the Kuan filter but assumes that the data is gamma distributed. The pixel being filtered is replaced with a value calculated based on the local statistics.

$$\bar{I} = \frac{K\bar{I} + \sqrt{I^{-2}K + 4\alpha N\bar{I}}}{2((1 + C_u^2)/(C_i^2 - C_u^2))} \quad (7)$$

Where K is damping factor and

$$Cu = \sqrt{1/N_{Look}} \quad (8)$$

$$C_i = \sigma / \mu \quad (9)$$

Ci is the ratio of the standard deviation of intensity within window over the local mean,

Which I (x , y) is the will be filtered and $\bar{I}$ (x , y) is the output pixel,
μ = Mean value of intensity within window,
σ = standard deviation of intensity within window,
$N_{look=}$ Number Looks
Cu = The root square of the reciprocal of the Number Looks.

3.1.4. KUAN FILTER (Kuan et al., 1985)

The Kuan filter is used primarily to filter speckled radar data. The Kuan filter smooths the image data, without removing edges or sharp features in the images. It is only applicable for radar intensity image. Kuan filter first transforms the multiplicative noise model into a signal-dependent additive noise model. Then the minimum mean square error criterion is applied to the model. The resulting filter has the same form as the Lee filter but with a different weighting function. Because Kuan filter made no approximation to the original model, it can be considered to be superior to the Lee filter.

Kuan filter first transforms the multiplicative noise model into a signal-dependent additive noise model. Then the minimum mean square error criterion is applied to the model. The resulting filter has the same form as the Lee filter but with a different weighting function. Because Kuan filter made no approximation to the original model, it can be considered to be superior to the Lee filter.
The resulting grey-level value R for the smoothed pixel is:

$$R = I_c \times \omega + \mu \times (1 - \omega) \quad (10)$$

where:

$$\omega = (1 - C_u^2 / C_i^2)/(1 + C_u^2) \quad (11)$$

$$Cu = \sqrt{1/N_{Look}} \quad (12)$$

$$C_i = \sigma / \mu \quad (13)$$

C_i is the ratio of the standard deviation of intensity within window over the local mean,
$N_{look=}$ Number Looks,
I_c = canter pixel in filter window,
μ = Mean value of intensity within window,
σ = Standard deviation of intensity within window,
ω = Weight for each pixel,
Cu = the root square of the reciprocal of the Number Looks.

3.1.5. Local Sigma filter (Eliason et al., 1990)

Use Local Sigma filters to preserve fine detail (even in low contrast areas) and to reduce speckle significantly. The Local Sigma filter uses the local standard deviation computed for the filter box to determine valid pixels within the filter window. It replaces the pixel being filtered with the mean calculated using only the valid pixels within the filter box.

3.1.6. Bit Error filters (Eliason et al., 1990)

Use Bit Error filters to remove bit-error noise, which is usually the result of spikes in the data caused by isolated pixels that have extreme values unrelated to the image scene. The noise typically gives the image a salt-and-pepper appearance. Bit-error removal uses an adaptive algorithm to replace spike pixels with the average of neighbouring pixels. The local statistics (mean and standard deviation) within the filter box are used to set a threshold for valid pixels.

3.2. Texture Features and Co-occurrence Matrix

The Gray Level Co-occurrence Matrix is proposed and used by (Haralick et al., 1973). The power of features extracted made it popular among other algorithms used for texture analysis. Obtaining textural features of an image is based on the assumption that the texture information on an image *I* is contained in the overall or average spatial relationship which the gray tones in the image I have with one another. More specifically, this texture information is adequately specified by a set of gray tone spatial dependence matrices; that are computed for various angular relationships and distances between neighboring resolution cell pairs on the image. The features are derived from these gray tone spatial dependence matrices.

(Haralick et al., 1973) proposed and show that:

- Over the last few years significant progress has been made in applying methods using distributions of feature values to texture analysis. Very good performance has been obtained in various texture classification and segmentation problems (Haralick et al., 1973).

- Many images contain regions characterized by variation in brightness rather than any unique value of brightness. Texture refers to the spatial variation of image tone as a function of scale. To be defined as a distinct textural area, the gray levels within the area must be more homogeneous as a unit than areas having a different texture (Haralick et al., 1973).

- Occurrence Measures are used to apply any of different texture filters that are based on occurrence measures. Occurrence measures use the number of occurrences of each gray level within the processing window for the texture calculations. Use Co-Occurrence Measures to apply any of eight texture filters that are based on the co-occurrence matrix. These filters include mean, variance, homogeneity, contrast, dissimilarity, entropy, second moment, and correlation. Co-occurrence measures use a gray-tone spatial dependence matrix to calculate texture values. This is a matrix of relative frequencies with which pixel values occur in two neighbouring processing windows separated by a specified distance and direction. It shows the number of occurrences of the relationship between a pixel and its specified neighbour. For example, the co-occurrence matrix shown below figure (3-2) was produced from original image as shown below in figure (3-1) using each pixel and its horizontal neighbour (shift values of X=1, Y=0) for a 4 x 4 window. The pixels in the 4 x 4 base window and the pixels in a 4 x 4 window that was shifted by 1 pixel are used to create the co-occurrence matrix (Haralick et al., 1973).

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

Figure (3-1) Original matrix

$$p_{0^0,1=}\begin{bmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

Figure (3-2) co-occurrence matrix with zero angle

- Texture is an important characteristics used in identifying objects or regions of interest in an image. Textural features probably have a general applicability for a wide variety of image classification. The most difficult step in categorizing pictorial information front a large block of resolution cells is that of defining a set of meaningful features to describe the pictorial information from the block of resolution cells. Once these features are defined, image blocks can be categorized using any one of a multitude of pattern-recognition techniques (Haralick et al., 1973).

- Texture is an important cue, which is used by human beings in interpreting Pictorial information. Despite its importance in image analysis, a precise definition of texture does not exist. A dictionary definition of texture is:Something composed of closely interwoven elements. The structural approach to texture analysis is suitable for describing textures where there is much regularity in the placement of texture elements. The statistical approach generates features to characterize the stochastic properties of the spatial distribution of grey levels in an image. They have been proved to be more powerful and useful than the structural features (Haralick et al., 1973).

- One of the most popular approaches to image analysis is based on its Grey-Level Co-occurrence Matrix: A second order statistical measure of image variation. It gives the joint probability of occurrence of grey levels of two pixels separated spatially in a fixed manner. Division by the total number of co occurrences would give the corresponding probability values. Smooth texture gives co-occurrence matrix with high values along diagonals if d is small compared to texture spatial variation. If texture is large compared to d, values of the marginal probability of gray-level co-occurrence matrix in the direction of the pixels and lines, Pd, are spread more uniformly throughout. Pd does not directly provide a single feature for texture discrimination (Haralick et al., 1973).

- Texture is an important characteristic for the analysis of many types of images. It can be seen in all images from multispectral scanner images obtained from aircraft or satellite platforms (which the remote sensing community analyzes) to microscopic images of cell cultures or tissue samples (which the biomedical community analyzes). Despite its importance and ubiquity, in image data, a formal approach or precise definition of texture does not exist (Haralick et al., 1973).

- Think of this kind of texture as an organized area phenomenon. When it is decomposable, it has two basic dimensions on which it may be described. The first dimension is for describing the primitives out of whom the image texture is composed, and the second dimension is for the description of the spatial dependence or interaction between the primitives of an image texture. The first dimension is concerned with tonal primitives or local properties and the second dimension is concerned with, the spatial organization of the tonal primitives. Tonal primitives are regions with tonal properties. The tonal primitive can be described in terms such as the average tone, or maximum and minimum tone of its region. The region is a maximally connected set of pixels having a given tonal property. The tonal region can be evaluated in

terms of its area and shape. The tonal primitive includes both its gray tone and tonal region properties (Haralick et al., 1973).

- The number and types of its primitives and the spatial organization or layout of its primitives describe an image texture. The spatial organization may be random, may have a pairwise dependence of one primitive on a neighbouring primitive, or may have a dependence of n primitives at a time. The dependence may be structural, probabilistic, or functional (like a linear dependence) (Haralick et al., 1973).

- To objectively use the tone and textural pattern elements, the concepts of tonal and textural feature must be explicitly defined. With an explicit definition, we discover that tone and texture are hot independent concepts. They bear an inextricable relationship to one another very much like the relation between a particle and a wave. There really is nothing that is solely particle or solely wave. Whatever exists has both particle and wave properties and depending on the situation, the particle or wave properties may predominate. Similarly, in the image context, tone and texture are always there, although at times one property can dominate the other and we tend to speak of only tone or only texture. Hence, when we make an explicit definition of tone and texture, we are not defining two concepts: we are defining one tone-texture concept (Haralick et al., 1973).

- The basic interrelationships in the tone-texture concept are the following. When a small-area patch of an image has little variation of tonal primitives, the dominant property of that area is tone. When a small-area patch has wide variation of tonal primitives, the dominant property of that area is texture. So that there is only one discrete feature, the only property present is simple gray tone. As the number of distinguishable tonal primitives increases within the small-area patch, the texture property will dominate. **When the spatial pattern in the tonal primitives is random and the gray tone variation between primitives is wide, a fine texture results. As the spatial pattern becomes more definite and the tonal regions involve more and more resolution cells, a coarser texture results** (Haralick et al., 1973).

(Haralick, 1979) proposed and show that:

- One aspect of texture is concerned with the spatial distribution and spatial dependence among the gray tones in a local area. (Jules, 1962) first used gray tone spatial dependence co occurrence statistics in texture discrimination experiments. (Darling and Joseph, 1968) used statistics obtained from the nearest neighbour gray tone transition matrix to measure this dependence for satellite images of clouds and was able to identify cloud types on the basis of their texture. (Bartels et al., 1976) and (Weid et al., 191970) used one-dimensional concurrence in a medical application. (Rosenfeld and Troy, 1970) and (Haralick, 1971) suggested two-dimensional spatial dependence of the gray tones in a co occurrence matrix for each fixed distance and/or angular spatial relationship; (Haralick et al., 1972, 1973) used statistics of this matrix as measures of texture in satellite imagery, (Haralick et al., 1973, 1974) aerial, and microscopic imagery. (Chien and Fu, 1974) **showed the application of gray tone co occurrence to automated chest X-ray analysis**. (Pressman, 1976) showed the application to cervical cell discrimination. (Chen and Pavlidis, 1978) **used co occurrence in conjunction with a**

split and merge procedure to segment an image on the basis of texture. All these studies achieved reasonable results on different textures using gray tone co occurrence. The gray tone co occurrence can be specified in a matrix of relative frequencies P_i with which two neighbouring resolution cells separated by distance (d) occur on the image, one with gray tone i and the other with gray tone j. Such matrices of spatial gray tone dependence frequencies are symmetric and a function of the angular relationship between the neighbouring resolution cells as well as a function of the distance between them. For a 0° angular relationship, they explicitly average the probability of a left-right transition of gray tone i to gray tone j within the right-left transition probability (c.f Haralick, 1979).

- Texture could be defined as a structure composed of a large number of more or less ordered similar elements or patterns without one of this drawing special attention. We could think of a strictly ordered array of identical sub-patterns, like a checkerboard for instance. Such a texture is called deterministic. It can be described by the characteristics of one such sub-pattern or "primitive" and by the "placement rules" defining the spatial distribution of the primitives. We could also have in mind a pattern merely obeying some statistical laws. The resulting structure might resemble noise on a television screen. Such a texture is said to be stochastic. We have to point out, however, that deterministic textures can be heavily disturbed in their repetitiveness and their primitives might be similar but not identical at all. Then the texture is no longer an ideal one but is referred to as an "observable texture". In computer vision literature there are many different definitions of texture (Haralick, 1979).

- Texture is one of the important characteristics used in identifying objects or regions of interest in an Image whether the image be an aerial photograph or a satellite image. Textural features probably have a general applicability for a wide variety of image-classification applications. These set of features for classifying or categorizing pictorial data. The classification of pictorial data can be done on a resolution cell basis (such as in identifying the crop category of a resolution cell on satellite imagery) or on a block of contiguous resolution cells (such as in identifying the crop category of an entire agricultural field extending over a large number of resolution cells) (Haralick, 1979).

- The most difficult step in categorizing pictorial information from a large block of resolution cells is that of defining a set of meaningful features to describe the pictorial information from the block of resolution cells. Once these features are defined, image blocks can be categorized using any one of a multitude of pattern-recognition techniques (Haralick, 1979).

- In a search for meaningful features for describing pictorial information, it is only natural to look toward the types of features, which human beings use in interpreting pictorial information. Spectral, textural, and contextual features arc three fundamental pattern elements used in human interpretation of colour photographs. Spectral features describe the average tonal variations in various bands of the visible and/or infrared portion of an electromagnetic spectrum, whereas textural features contain information about the spatial distribution of tonal variations within a hand (Haralick, 1979).

- Contextual features contain information derived from blocks of pictorial data surrounding the area being analysed. When small image areas from black and white photographs are independently processed by a machine, then texture and tone arc most important. The concept of tone is based on the varying shade, of gray of resolution cells in a photographic image, while texture is concerned with the spatial (statistical) distribution of gray tones (Haralick, 1979).

- Texture and tone are not independent concepts; rather, they bear an inextricable relationship to one another very much like the relationship between a particle and a wave. Context, texture, and tone are always present in the image, although at times one property can dominate the other. Texture is an innate property of virtually all surfaces the grain of wood, tile weaves of a fabric, the pattern of crops in a field, etc. It contains important information about the structural arrangement of surfaces and their relationship to the surrounding environment. Texture has been extremely refractory to precise definition and to analysis by digital computers. The textural properties of images appear to carry useful information for discrimination purposes (Haralick, 1979).

Based on repeated occurrence of some gray-level configuration in the texture this configuration varies rapidly in fine textures, more slowly in coarse textures occurrence of gray-level configuration may be described by matrices of relative frequencies, called **co-occurrence matrices** these matrices are symmetric but asymmetric definition is also possible. Texture classification can be based on criteria (features) derived from the co-occurrence matrices (Sonka, 1998).

Texture features were computed by the following equations.

Energy:

$$\sum_i \sum_j \mathrm{P}_{\phi,d}(i,j)^2 (i,j) \tag{14}$$

Entropy:

$$\sum_i \sum_j \mathrm{P}_{\phi,d}(i,j)^2 (i,j) \log \mathrm{P}_{\phi,d}(i,j) \tag{15}$$

Contrast:

$$\sum_i \sum_j \left| i-j \right|^k \mathrm{P}_{\phi,d}(i,j)^\lambda (i,j) \tag{16}$$

Homogeneity:

$$\sum_i \sum_j \frac{\mathrm{P}_{\phi,d}(i,j)(i,j)}{\left|i-j\right|} \tag{17}$$

Correlation

$$\frac{\sum_{i,j}(i,j)\mathrm{P}_{\phi,d}(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{18}$$

where means and standard deviations are defined as

$$\begin{aligned}
\mu_x &= \sum_i i \sum_j \mathrm{P}_{\phi,d}(i,j) \\
\mu_y &= \sum_j j \sum_i \mathrm{P}_{\phi,d}(i,j) \\
\sigma_x &= \sum_i (i-\mu_x)^2 \sum_j \mathrm{P}_{\phi,d}(i,j) \\
\sigma_y &= \sum_j (j-\mu_x)^2 \sum_i \mathrm{P}_{\phi,d}(i,j)
\end{aligned} \tag{19}$$

Variance

$$\sum_i \sum_j (i-\mu)^2 p(i,j) \tag{20}$$

Mean

$$\sum_{i=2}^{2N} i p_{x+y}(i) \tag{21}$$

3.3. Image Fusion and Principal Component analysis

(van Genderen, et al. 1998) show that:

- (Ehlers, 1991) **show that the linear band combinations lead to a high correlation of the resulting bands with loosing data**. (van Genderen and Pohl, 1994) show that, image fusion is performed at three different processing levels according to the stage at which the fusion takes place:
  1. Pixel
  2. Feature
  3. Decision level.

- (van Genderen, et al. 1994) show that similar objects with similar features from multiple sources are assigned to each other and then fused for further assessment using statistical approach. This will enhance the information apparent in the image as well as increase the reliability of the interpretation.

- (Rogers and Wood, 1990) show that image fusion increased confidence, reduce ambiguity, improve reliability, increased utility and improve classification.

Image fusion is applied to digital imagery in order to:

- Sharpen images (Chaves et al., 1991).
- Improve geometric corrections (Strobl et al., 1990).
- Provide stereo-viewing capabilities for stereo photogrammetry (Bloom et al., 1988).
- Enhance certain features not visible in either of the single data alone (Leckie, 1990).
- Complement data sets for improved classification (Schistad-Solberg et al., 1994).
- Detect changes using multitemporal data (Duguay et al., 1987).
- Substitute missing information (e.g., clouds-VIR, shadows-SAR) in one image with signals from another sensor (Aschbacher and Lichtenegger, 1990) replace defective data (Suits et al., 1988).

3.4. Principal Component analysis

Principal component analysis refers mathematically to the Characteristic Equation; the characteristic equation of a system is based upon the transfer function that models the system. It contains information needed to determine the response of a dynamic system. There is only one characteristic equation for a given system. PCA is not a statistical procedure but is merely a mathematical manipulation to recast n variables as n factors. This linear transform has been widely used in data analysis and compression. The characteristic equation is the equation, which is solved to find a matrix's eigenvalues, also called the characteristic polynomial. For a general k*k matrix A, the characteristic equation in variable λ is defined by

$$\det(A - \lambda I) = 0 \tag{22}$$

where I is the identity matrix and det (B) is the determinant of the matrix B. Writing A out explicitly gives

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{1k} \\ a_{21} & a_{21} & a_{21} & a_{21} \\ \dots & \dots & \dots & \dots \\ a_{k1} & a_{k1} & a_{k1} & a_{kk} \end{bmatrix} \tag{23}$$

So the characteristic equation is given by

$$\begin{bmatrix} a_{11}-\lambda & a_{12} & a_{13} & a_{1k} \\ a_{21} & a_{21}-\lambda & a_{21} & a_{21} \\ \dots & \dots & \dots & \dots \\ a_{k1} & a_{k1} & a_{k1} & a_{kk}-\lambda \end{bmatrix} = 0 \qquad (24)$$

The solutions of the characteristic equation are called eigenvalues, and are extremely important in the analysis of many problems in mathematics, physics and also in remote sensing. The polynomial left-hand side of the characteristic equation is known as the characteristic polynomial.

As summarised in Ilwis 3.11 Academic help:
The Principal component analysis operation results in a linear transformation of a set of raster maps. The map values are transformed into raster map values, called components.
The following steps apply:

- The covariance matrix of the input raster maps is computed.
- The orientation of the components is computed based on the eigenvectors and Eigenvalues of the covariance matrix.
- The vectors in each column are normalized. The normalized values are the principal component analysis coefficients.

With two input raster maps the principal components analysis coefficients matrix looks like:

$$\begin{bmatrix} \alpha 1, \alpha 2 \\ \beta 1, \beta 2 \end{bmatrix} \qquad (25)$$

Where:
$\alpha 1$, $\alpha 2$ is coefficients of the first component.
$\beta 1$, $\beta 2$ is coefficients of the second component.
With M bands an MxM matrix of principal component analysis coefficients is calculated.

The Principal Component Analysis operation is a mathematical method to uncover relationships among many variables (as found in a set of raster maps in a map list) and to reduce the amount of data needed to define the relationships. With Principal Component Analysis each variable is transformed into a linear combination of orthogonal common components (output raster maps) with decreasing variation. The linear transformation assumes the components will explain all of the variance in each variable. Hence each component (output raster map) carries different information, which is uncorrelated with other components. The input for Principal components analysis consists of a map list with raster maps from which the covariance matrix will be calculated. The output raster maps are listed in decreasing order of variance. This enables a reduction of output maps because the last number of transformed maps has little or no variation left (may be virtually constant maps). The 'last' components thus do not add significance and may hence be discarded. During the operation, specifying a parameter can directly reduce the number of output bands. The additional information of the output matrix contains furthermore the amount of variance accounted for by each component.

Principal components analysis can be used for several purposes, e.g.:

- **Data compression**
- **Data Classification.**
- **Finding targets of interest**.

PCA is the most popular instance of second main class of unsupervised learning methods, projection methods aim to find small number of "directions" in input space that explain correlations in input data; re-represent data by projecting along those directions data is assumed to be continuous, linear relationship between data and learned representation

4. Shape Algorithm and data processing

4.1. Introduction

One of the most common environmental problems in the last decades is the appearance of oil slicks in marine, coastal or river waters. The main sources are accidents due to boat mishaps, fires or oil rig explosions/blowouts, illegal oil product discharges and accidents during fuel transportation.

4.2. Data Sets

Three data sets have been analysed, 0955hh with low moderate waves (dataset North Sea event, Holland 128 images) and Vlek3, Vlek10 with high waves (dataset Prestige event, Spain 65,170 images respectively).

4.3. Algorithm Methodology

Figure (1) shows the main algorithm with all its steps as follows:

- Step1: Image extraction and conversion it to polar then Range and angle correction for the ship motion.
- Step2: Conversion from polar to Cartesian images then average of the first 10 images.
- Step3: Adaptive filters were applied then Gamma filter was selected according to Signal-to-Noise Ratio.
- Step4: Co-occurrence matrix with eight textures was applied then correlation texture was selected According to (Haralick, 1973) to identify, describe and extract the oil from the block of resolution cells, then image features can be categorized (classified).
- Step5: We propose using of Principal component analysis for texture fusion.

4.3.1. Step1: Image extraction and conversion it to polar then Range and angle correction for the ship motion

Running Matlab code to read data file and to convert it to raw data and then convert the raw data to polar images. The masts and cabin on a ship are not transparent for the radar; the radar cannot detect what happens behind the mast. Indeed, an image will arise but nothing can be said about the pixels, which are situated behind the mast. It is therefore imperative that the shadows are not taken into account in further processing of the picture quality.

Processing of the data and the intensity images is a more complicated matter because of the fact that the quality is poor at times depending on weather conditions. A number of operations make it still possible to detect the oil in the pictures. First an adjustment has to be made for the range. This is necessary because in the range intensity decreases. This decrease can be described as the surface of a

cone, which becomes larger with height. The height is the range. However, the base is not a flat plane but a spherical segment with a radius equal to the range. After adjustment of the range there is a second intensity shift depending on direction and speed of the wind is far more complicated than for the range as shown in figures (4-1a), (4-2) and (4-3) (Misset, 2003), see Appendix 2 and Appendix 3.

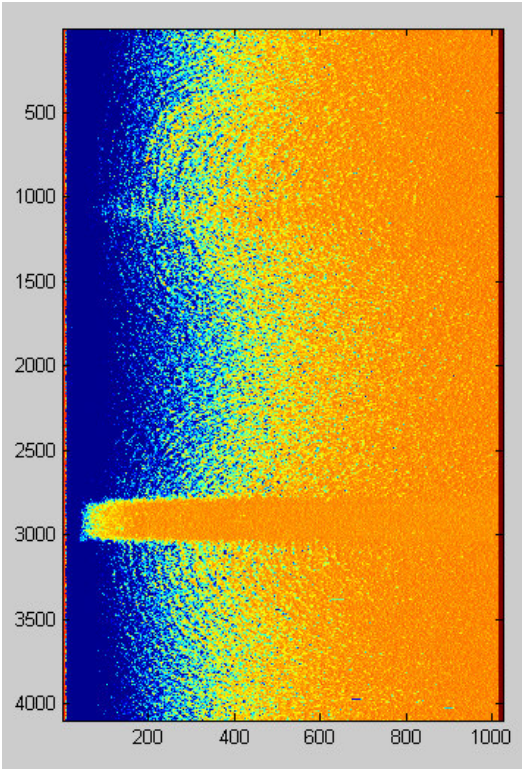


Figure (4-1a) first image of (data set 0955hh) 4096*1024 polar image without any corrections.

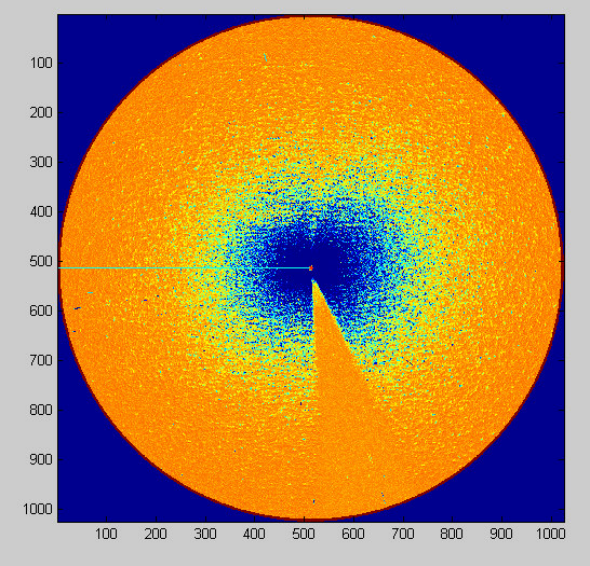


Figure (4-1b) first image of (data set 0955hh) 1024*1024 Cartesian image without any corrections.

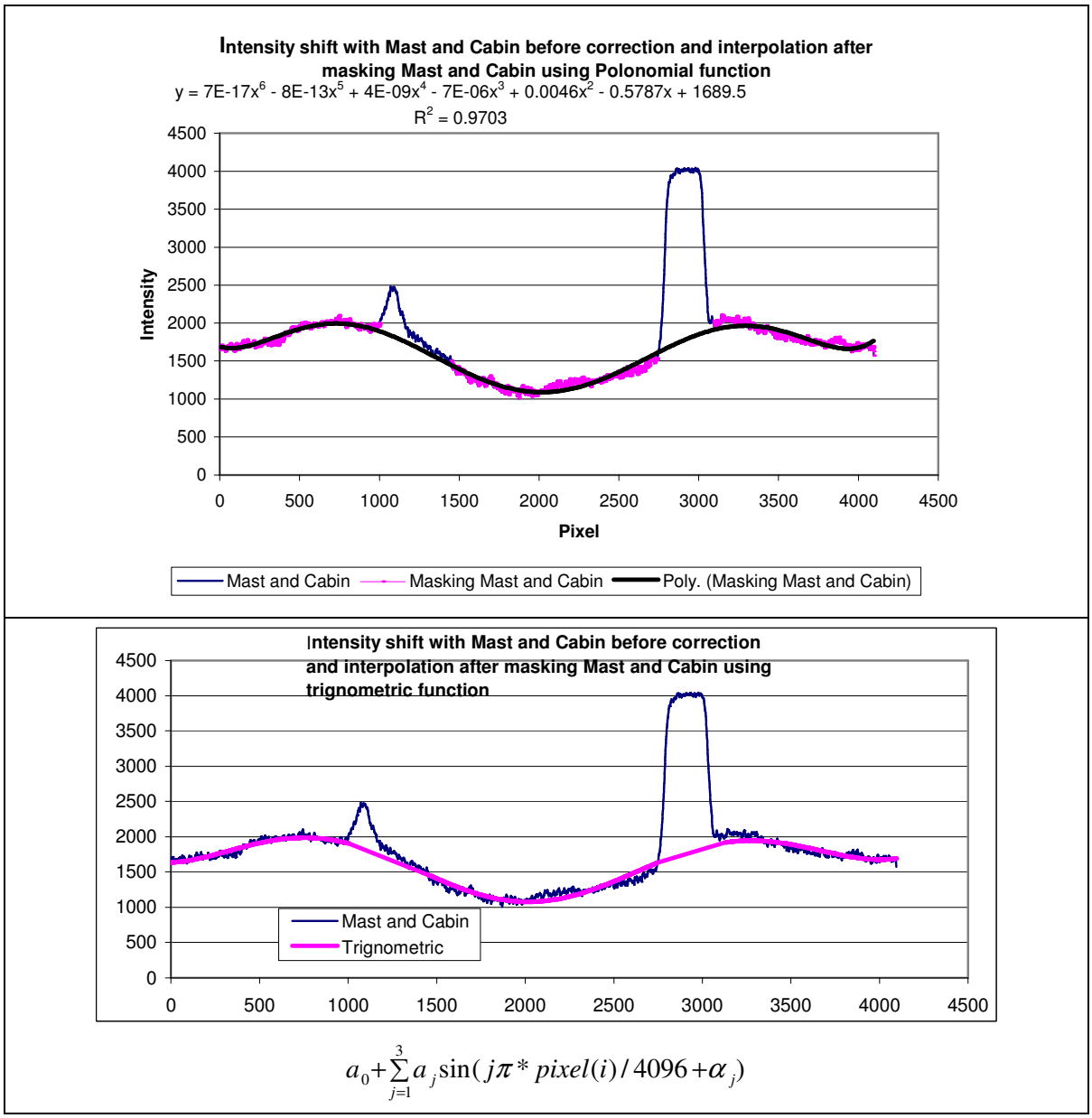


Figure (4-2) Intensity shift with Mast and Cabin before correction and interpolation after masking Mast and Cabin.

Figure (4-2) shows interpolation using two different methods.

4.3.2. Step2: Conversion from polar to Cartesian images then average of the first 10 images

Another code to convert images from polar to Cartesian images using function pol2cart figure (4-1b) and (4-4). To avoid having a wrong detection we average only the first 10 images as shown in figures (4-5) and figure (4-6) show the average of first 50images, see Appendix 4.

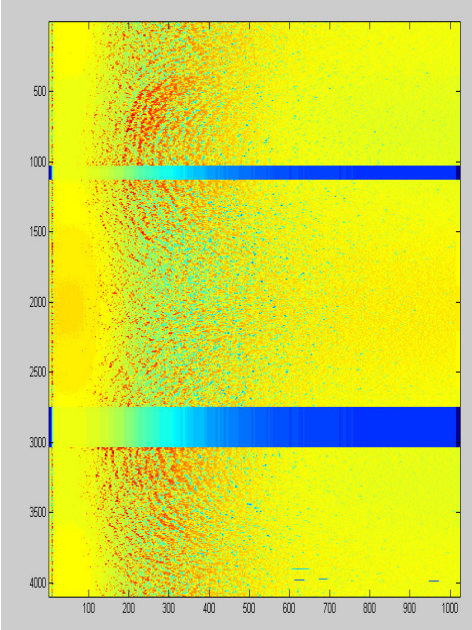


Figure (4-3) Second Polar image of data set 0955hh after masking Mast, Cabin and range correction.

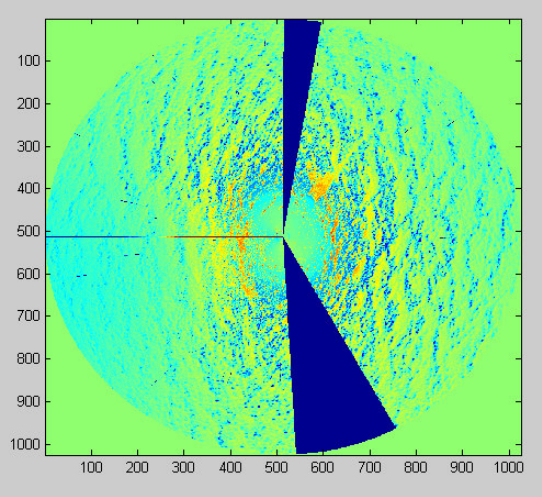


Figure (4-4) Second Cartesian image of data set 0955hh after masking Mast, Cabin and range correction.

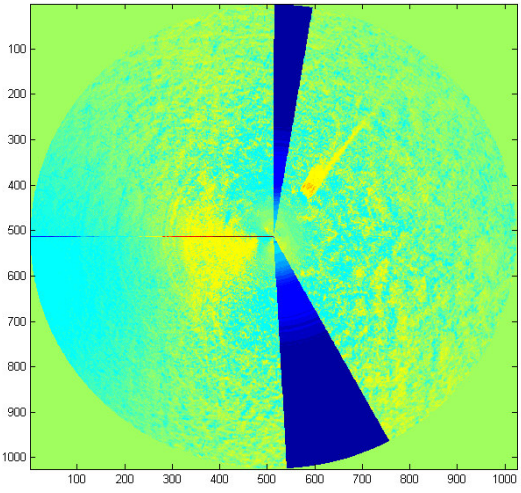


Figure (4-5) Average of first 10 Cartesian images after masking Mast, Cabin and range correction data set Vlek10.

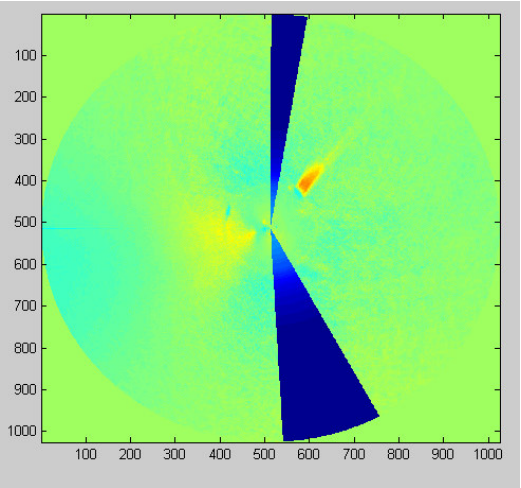


Figure (4-6) Average of 50 Cartesian images after masking Mast, Cabin and range correction data set Vlek10.

4.3.3. Step3: Adaptive filters were applied then Gamma filter was selected according to Signal to Noise Ratio.

The images are characterized with high speckle/noise. The gray-tone variation between features is not wide in all the three data sets 0955hh,Vlek3 and Vlek10, so according to (Haralick et al., 1973) no fine texture result, so filter should be applied first. One important property of tone-texture is the spatial pattern of the resolution cells composing each discrete tonal feature. When there is no spatial pattern and the gray-tone variation between features is wide, a line texture results. As the spatial pattern becomes store definite and involves more and more resolution cells, a coarser texture results (Haralick et al., 1973).

Six spatial filters are applied to the image and the selection of the best filter is done based on the results obtained through SNR which gives the best smoothing, despeckling and it seems to keep the edges best without blurring the minimum.

4.3.4. Step4: Co-occurrence matrix with eight textures was applied then correlation texture was selected

Co-occurrence matrix with eight textures was applied then correlation texture was selected According to (Haralick et al., 1973) to identify, describe and extract the oil from the block of resolution cells, then image features can be categorized (classified).

Co- occurrence is applied to categorize the image to "Oil slick" and water. Eight textures with 0° angular relationship and d=1 were employed such as: Energy; Homogeneity; Mean; Variance; contrast dissimilarity; Entropy; Correlation.

We can describe gray tone as microstructure of an image and texture as the macrostructure see Appendix 5.

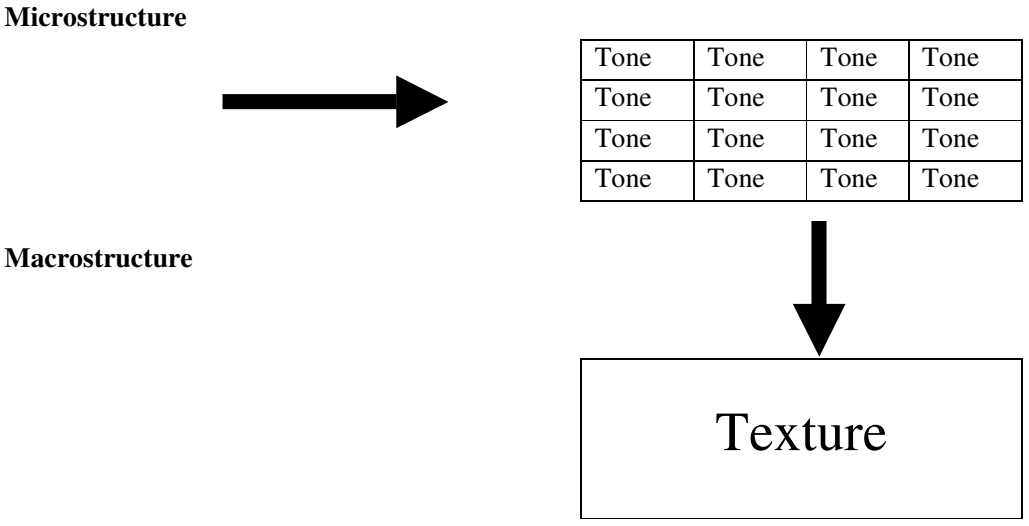


Figure (4-7) relation between Tone and Texture

4.3.5. Step5: New method by using Principal component analysis for texture fusion.

The use of linear combinations has the disadvantage that these band combinations lead to a high correlation of the resulting textures. Principal Component analysis has been proposed in this thesis to fuse eight textures to extract one that has the highest variance.

To avoid the disadvantage of the use of linear combinations that lead to a high correlation of the resulting textures and loosing textural information, Non-Standardized Principal Component Analysis (the mean not equal to zero and variance not equal to unity) was applied to extract all the information from the eight textures and to discriminate between oil and water. The first principal component of covariance matrix, i.e. the eigenvector with the largest eigenvalue corresponds to an identifiable physical property, which is most important for interpretation of the patterns. The higher order Non-Standardized Principal Component Analysis and Correlation method produce the same results in this case and were able to account for oil spill detection, see Appendix 6.

5. Results & Conclusions

5.1. Intoduction

SHIRA (Ship's Radar) is digital X-band radar that images the sea surface, with a rotating antenna and PC-based data processing capability. It is not only a relatively low-cost instrument, but also extremely versatile and cheap to run. It is a remote sensing instrument developed by the TNO Physics and Electronics Laboratory on the basis of a conventional ship's navigation radar. SHIRA, however, extracts information from the sea clutter, which in navigation radar is normally suppressed. SHIRA can be operated from a ship, a (production) platform or an elevated coastal site. At each antenna revolution, i.e. about once a second, SHIRA digitises and stores a radar image of an operator-specified area of the sea surface. Compensating for possible ship movements, SHIRA can aggregate many co-located images taken during successive antenna revolutions, in this way producing a time series of images.

Three experiments have been carried out in 1993 in which SHIRA's performance was tested on artificial oil spills. The following recommendations were made:

- Improve the low-noise logarithmic amplifier to increase sensitivity and dynamic range.
- Digitisation of backscatter values below the noise level.
- Improve the contrast optimisation and quality of the real-time display.

The first two of these improvements were implemented immediately (Kleijweg, 1994).

Misset (2003) proposed a method to improve the third one (the image quality) applying a moving average through the images (stacking). He found that the more stacking the more having a wrong image and a wrong detection.

The main objective of my work is to improve the 3 rd aspect. A new algorithm based on texture analysis has been applied. Quantitative comparison has been made between various filters, which are able to reduce variance in homogeneous areas, preserve edges and lines, suppress point scatter, and preserve spatial variability, while avoiding artefacts. Gamma Filter yields the best smoothing and despeckling. Moreover, it seems to preserve the edges without blurring the minimum. Eight textures are applied, based on the co-occurrence matrix. These textures include mean, variance, homogeneity, contrast, dissimilarity, entropy, second moment, and correlation. Co-occurrence measures use a gray-tone spatial dependence matrix to calculate texture values. This is a matrix of relative frequencies with which pixel values occur in two neighbouring processing windows separated by a specified distance and direction. It shows the number of occurrences of the relationship between a pixel and its specified neighbour.

To extract all the information from the eight textures and avoiding the disadvantage of the linear combination, we proposed using Non-Standardized Principal component Analysis.

5.2. Filters

In this study, we examine the effect of using different types of filters to improve image quality and quantitative comparison has been made. (Schwan et al., 1995) show that best speckle filtering must reduce variance in homogeneous areas with preservation of edges, lines, and spatial variability.

Tables (5-1), (5-2) and (5-3) show SNR evaluates the performance of filters to describe its quality. Gamma Filter yields to give the best smoothing and despeckling. Moreover it seems to preserve the edges, lines and spatial variability best without blurring the minimum by preserving the mean (the average intensity of the images) and keep the images normally fine.

Signal to Noise Ratio (SNR) is able to relate between the mean and the degree of deviation between the gray level and its mean value for the overall image as shown in Tables (5-1), (5-2) and (5-3) and figures (5-1), (5-2) and (5-3). Although oil slicks are often not homogeneous, because they are broken up by the waves, gamma filter shows still the best method in this case where we find no difference in SNR for average from 1 to 10, from 51 to 55 or from 151 to 160 images and also better than stacking.

$$SNR = \mu / \sigma \tag{26}$$

μ = the mean, σ = the standard deviation.

Filter type		Data set 0955hh	
Window 7*7	Mean	Standard deviation	SNR
Avg10	254	13	20
Avg50	228	52	4
Lee	254	9	28
Gamma	**254**	**8**	**32**
Frost	161	18	9
Kuan	254	11	23
Local Sigma	22	71	0
Bit Error	254	12	21

Table (5-1) SNR (data set 0955hh average of first 10 images)

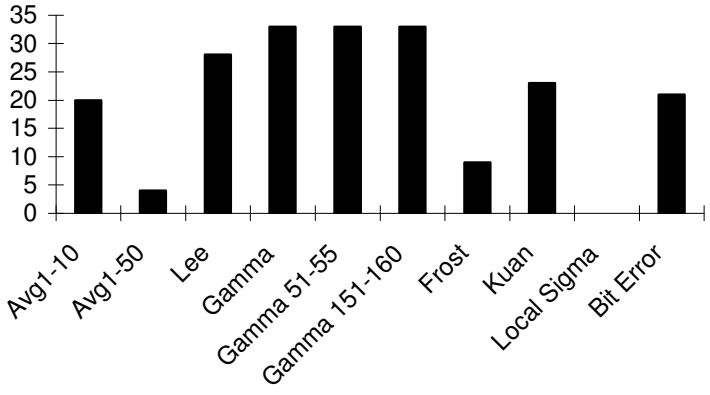


Fig (5-1) Signal to Noise Ratio (data set 0955hh)

Filter type	Data set Vlek3		
Window 7*7	Mean	Standard deviation	SNR
Avg10	227	52	4
Lee	227	46	5
Gamma	**227**	**43**	**5**
Frost	210	52	4
Kuan	227	49	5
Local Sigma	89	106	1
Bit Error	228	52	4

Table (5-2) SNR (data set Vlek3 average of first 10 images)

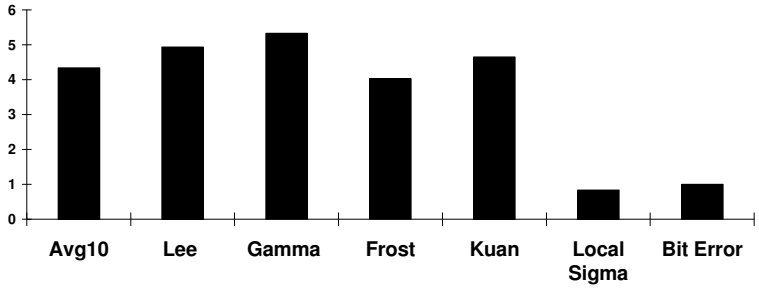


Fig (5-2) Signal to Noise Ratio (data set Vlek3)

Filter type	Data set Vlek10		
Window 7*7	Mean	Standard deviation	SNR
Avg10	228	52	4
Lee	228	42	5
Gamma	228	37	6
Frost	146	29	5
Kuan	228	47	5
Local Sigma	113	112	1
Bit Error	228	52	4

Table (5-3) SNR (data set Vlek10 average of first 10 images)

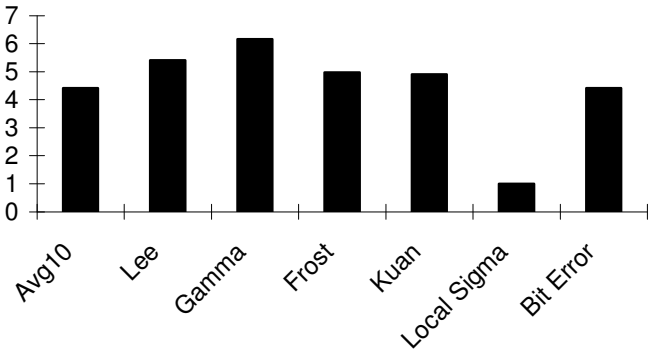


Fig (5-3) Signal to Noise Ratio (data set Vlek10)

5.3. Texture

(Haralick et al., 1973) show the correlation features for the water-body image have lower values. High discrimination between water body with low correlation and the oil with high correlation and different unit variance for each feature was proved in this study regardless the wind speed.

Window 7*7 with 0 angle and shift = one pixel, correlation texture performs best in terms of the probability of oil occurrence because it allows clear discrimination between synthetic oil (oval-round shaped), natural oil (striped), water waves and water for data with high waves (dataset Prestige event, Spain) and with low-moderate waves (dataset with oil simulations in North Sea).

Correlation texture was able to discriminate between oil, look alike phenomena and other organic pollutants as shown in figures (5-4), (5-5) and (5-6).

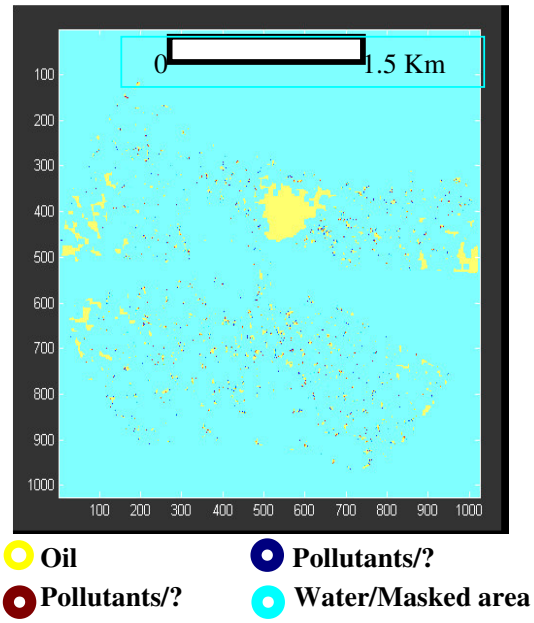


Figure 5-4 Correlation applied to data set Vlek3.

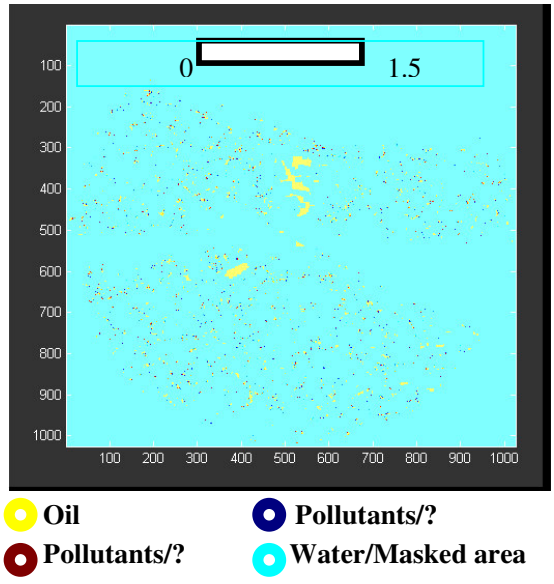


Figure 5-5 Correlation applied to data set Vlek10.

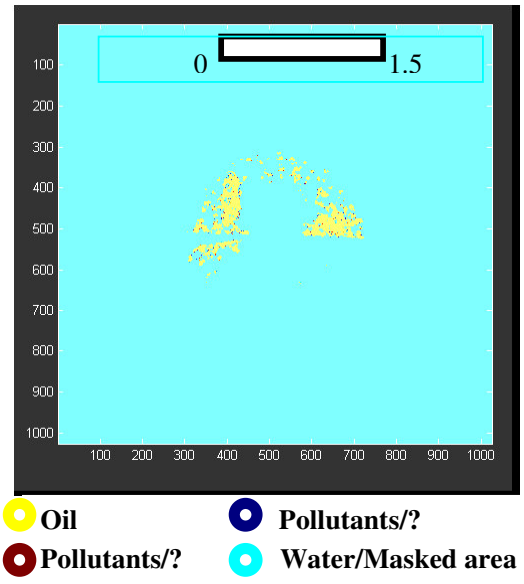
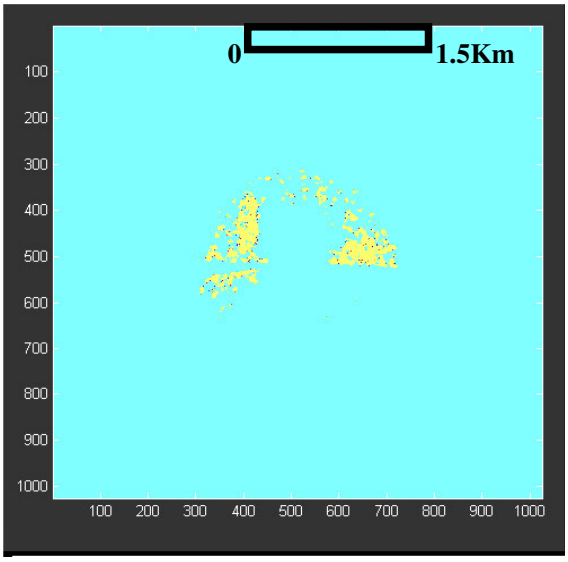


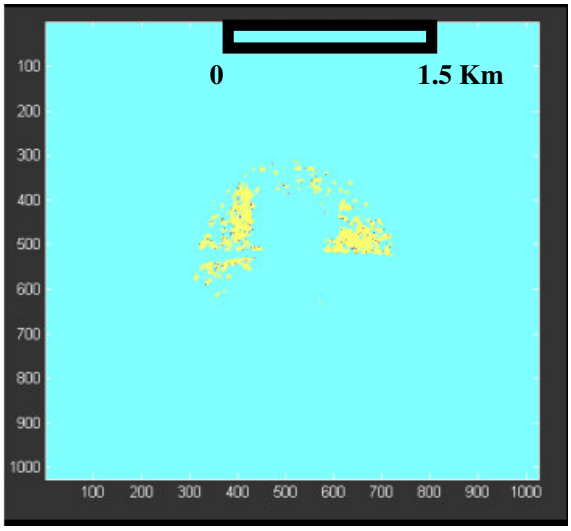
Figure 5-6 Correlation applied to data set0955hh.

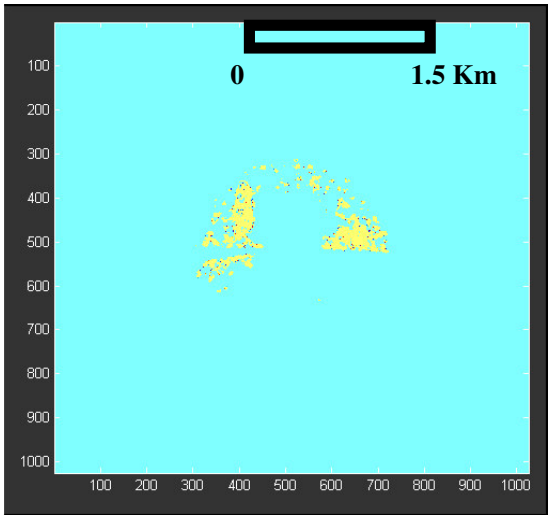
Note from figures (5-1) and (5-7) that we can't average any number of images because the movement of the ship, the more average the more blurring the minimum as shown in table (5-1). We proposed average of any 10 images or less for the reasons:

- Reduce the running time.
- Maximize SNR.

Note that neither the oil shape nor its size was changed when we average any 10 images or less. The ship didn't move very fast in this series.







Oil **Pollutants/?** **Pollutants/?** **Water/Masked area**

Figure (5-7) Correlation applied to data set0955hh average from 1 to 10, from 51 to 55 and from 151 to 160 images respectively.

5.4. Image fusion and Principal Component Analysis

To avoid the disadvantage of using linear combinations, which lead to a high correlation of the resulting texture principal component analysis, was applied for the following:

- To fuse the eight textures co-occurrence matrix.
- To evaluate which texture is the best method to discriminate between water body and oil.

In this study, the results obtained from the higher order Non-Standardized Principal Component Analysis and correlation texture were able to account for oil spill detection and established similarity with the results of (Haralick et al., 1973) hence Correlation is the best method to discriminate between water body and oil regardless wind speed. In the computation of principal components, the eigenvectors used for transformation can be derived from a covariance matrix eq.24 (Non-Standardized data) which equal to zero, i.e. uncorrelated as shown in Table (5-4).

The more eigenvectors are used in the combination the lower the error becomes. Selecting the eigenvectors associated with the largest eigenvalues can also minimize error. The result is optimum in the sense that it minimizes the mean square error between the textures and their approximations due to using the eigenvectors corresponding to the largest eigenvalues.

As shown in Table (5-4) the eigenvectors corresponding to the largest eigenvalue in the covariance matrix of the population, and this eigenvalue gives the variance of the gray levels of the transformed image. Thus based on the numbers shown in these tables the higher eigenvalue the higher contrast because of the first three images account for about 98.57 %, 97.13 % and 96.88 % of the total variance in the three data sets respectively as shown in figures (5-12), (5-13) and (5-14), the fact that the other five principal component images have low contrast is not unexpected. This means describing the content of eight images with three.

Note how PCA1 and Correlation are similar to each other.

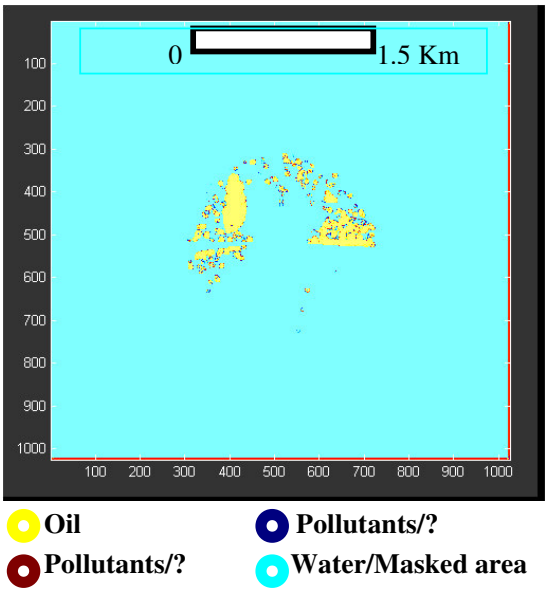


Figure (5-8) PCA1 with Eigenvalue percentage 66.91% data set 0955hh

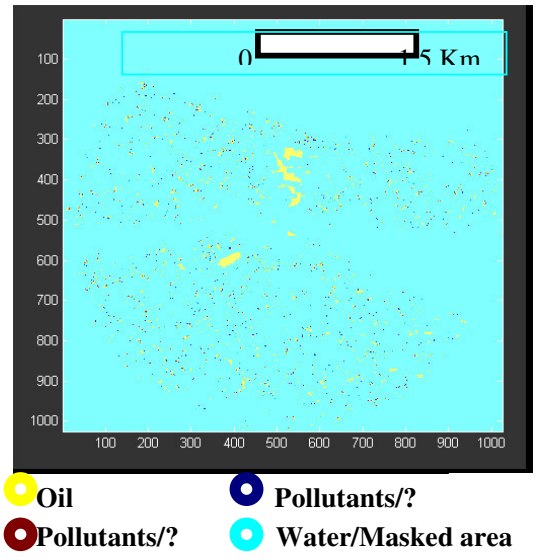


Figure (5-9) PCA1 applied to Co-occurrence Matrix with Eigenvalue percentage 85.98 % data set Vlek10.

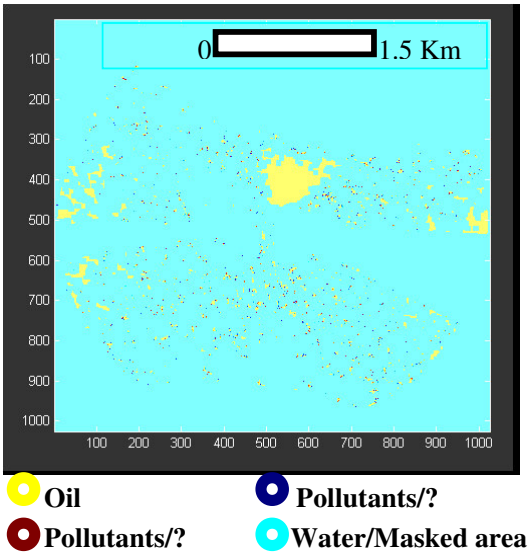


Figure (5-10) PCA1 applied to Co-occurrence Matrix with Eigenvalue percentage 86.27 % data set Vlek3.

Eigenvalue (Variance)/ band: 0955hh							
PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
118	52	4	2	0	0	0	0
Eigenvalue (Variance) percentages per band:							
67 %	29 %	2%	1 %	0	0	0	0

Eigenvalue (Variance)/ band: Vlek10							
PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
3222	247	163	114	2	0	0	0
Eigenvalue (Variance) percentages per band:							
86%	7 %	4 %	3 %	0	0	0	0

Eigenvalue (Variance)/ band: Vlek3							
PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
3757	310	163	124	1	0	0	0
Eigenvalue (Variance) percentages per band:							
86 %	7 %	4 %	3 %	0	0	0	0

Table (5-4) The eight Principal Components Coefficients (data sets 0955hh, Vlek10 and VLEK3) respectively.

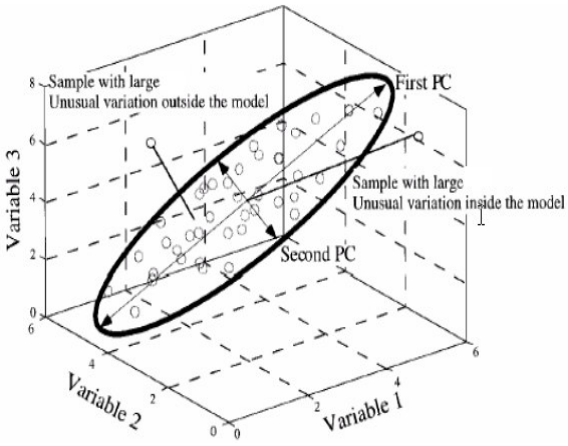
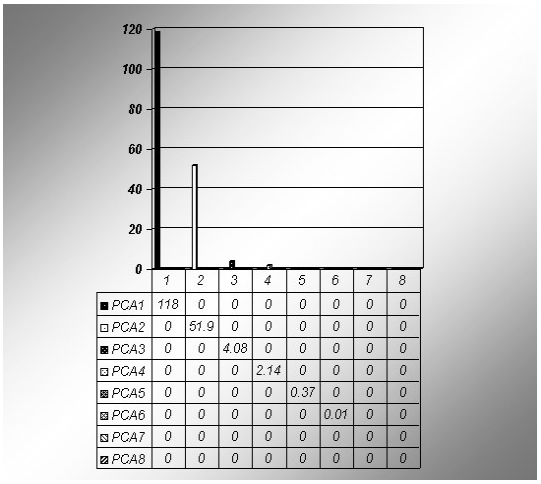
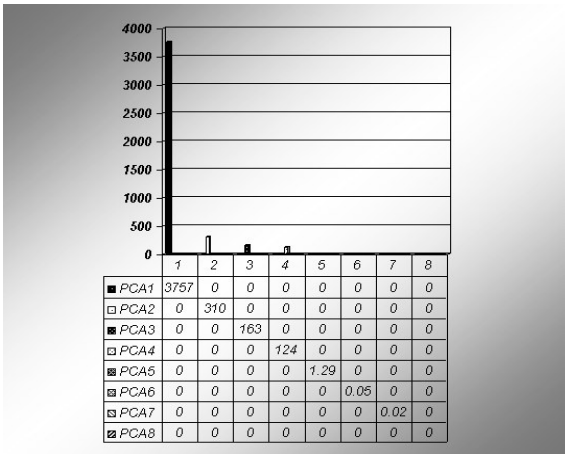


Figure (5-11) first three PCAs are perpendiculars to each other.



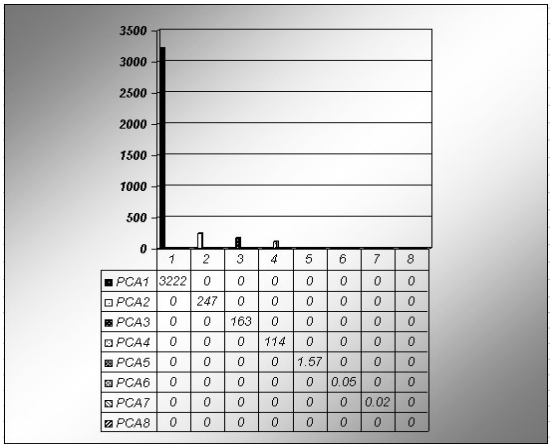
	1	2	3	4	5	6	7	8
PCA1	118	0	0	0	0	0	0	0
PCA2	0	51.9	0	0	0	0	0	0
PCA3	0	0	4.08	0	0	0	0	0
PCA4	0	0	0	2.14	0	0	0	0
PCA5	0	0	0	0	0.37	0	0	0
PCA6	0	0	0	0	0	0.01	0	0
PCA7	0	0	0	0	0	0	0	0
PCA8	0	0	0	0	0	0	0	0

Figure (5-12) Principal component analysis and co-variance matrix (data set 0955hh).



	1	2	3	4	5	6	7	8
PCA1	3757	0	0	0	0	0	0	0
PCA2	0	310	0	0	0	0	0	0
PCA3	0	0	163	0	0	0	0	0
PCA4	0	0	0	124	0	0	0	0
PCA5	0	0	0	0	1.29	0	0	0
PCA6	0	0	0	0	0	0.05	0	0
PCA7	0	0	0	0	0	0	0.02	0
PCA8	0	0	0	0	0	0	0	0

Figure (5-13) Principal component analysis and co-variance matrix (data set Vlek10).



	1	2	3	4	5	6	7	8
PCA1	3222	0	0	0	0	0	0	0
PCA2	0	247	0	0	0	0	0	0
PCA3	0	0	163	0	0	0	0	0
PCA4	0	0	0	114	0	0	0	0
PCA5	0	0	0	0	1.57	0	0	0
PCA6	0	0	0	0	0	0.05	0	0
PCA7	0	0	0	0	0	0	0.02	0
PCA8	0	0	0	0	0	0	0	0

Figure (5-14) Principal component analysis and co-variance matrix (data set Vlek3).

Note how the first three PCAs are perpendiculars to each other and most of the is around PCA1. The covariance matrix is symmetric and the diagonal represents PCAs as shown in Figure (5-11). The combination between PCA1, PCA2 and PCA3 yield lower error between the textures and their approximations due to using the eigenvectors corresponding to the largest eigenvalues.

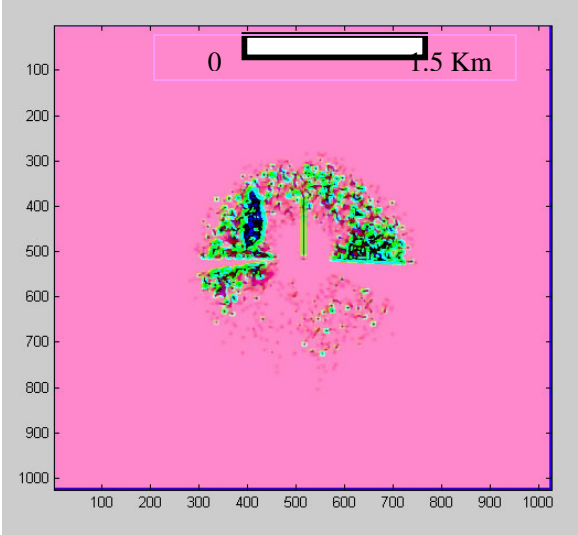


Figure (5-15) Combination of PCA1, PCA 2 and PCA 3 applied to Co-occurrence Matrix with Eigenvalue percentage 98.57 % data 0955hh

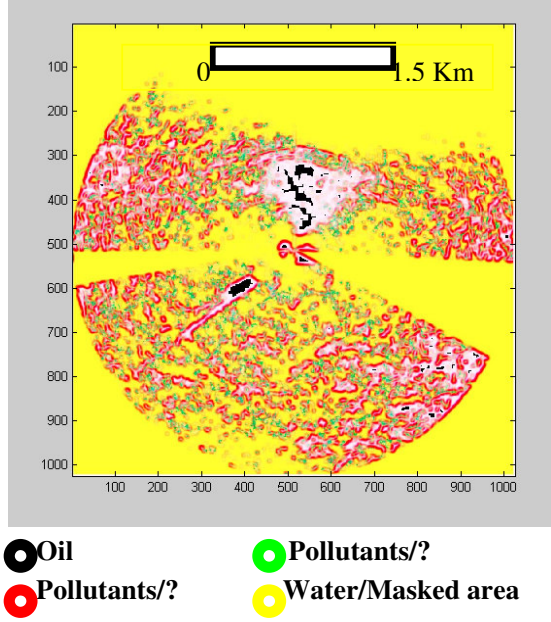


Figure (5-16) Combination of PCA1, PCA 2 and PCA3 applied to Co-occurrence Matrix with eigenvalue percentage 96.88 % data Vlek10

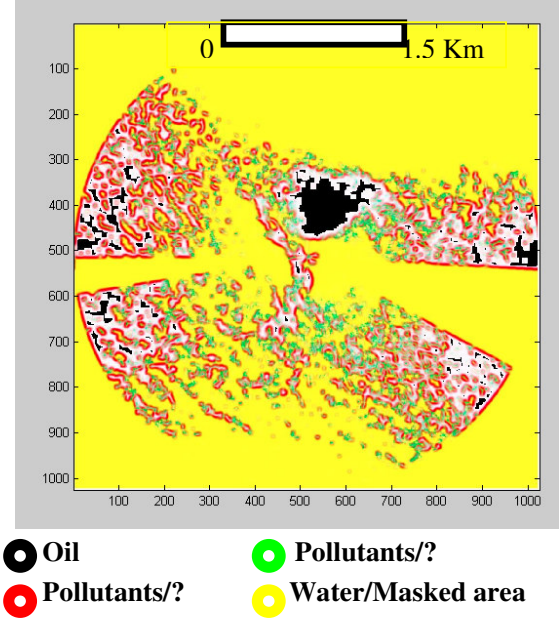


Figure (5-17) Combination of PCA1, PCA 2 and PCA3 applied to Co-occurrence Matrix with Eigenvalue percentage 97.13 % data Vlek3

5.5. Conclusions

To evaluate the performance of filters and describe their quality SNR was used. In all cases it has been found that Gamma Filter yields the best smoothing and despeckling results and it seems to preserve the edges best without blurring the minimum. Through the use of SNR the performance of different filters can be evaluated and the best one can be selected.

We can't average any number of images because the movement of the ship, the more average the more blurring the minimum. We proposed average of any 10, 5 images or less for the reasons:

- Reduce the running time.
- Maximize SNR.

Neither the oil shape nor its size was changed when we average any 10, 5 images or less. No influence of the number of images in the average of any 10, 5 images or less on the results but we found average of more than that like average of 20, 30, 40 or more blur the minimum and reduce SNR.

In this study although Oil slicks are often not homogeneous, because they are broken up by the waves Gamma Filter method still the best in this case and it was selected because of its high SNR and it preserves the images normally fine. Signal to Noise Ratio (SNR) is able to relate between the mean and the degree of deviation between gray levels and mean values for the entire image.

Gamma filter able to reduce the number of average of images to the minimum, even one image is enough except the first one of the series.

High discrimination between water body with low correlation and the oil with high correlation and different unit variance for each feature was proved in this study regardless of the wind speed. Correlation texture performs best, in terms of the probability of oil occurrence because it allows clear discrimination between synthetic oil (oval-round shaped) , natural oil (striped), water waves and water for data set with high waves (dataset Prestige event, Spain) and with low-moderate waves (dataset with oil simulations in North Sea). Correlation texture was able to discriminate between oil, look alike phenomena, and other organic pollutants.

In this study, the results obtained from the first PCA were similar to those obtained with correlation texture and were also similar to the results of (Haralick et al., 1973) hence correlation is the best method to discriminate between water body and oil regardless of wind speed. Uncorrelated linearly transformed components are derived from the co-occurrence matrix with eight textures such that the first principal component accounts for the maximum possible proportion of the variance of the eight textures. The result are similar to those of correlation texture which then account for the maximum proportion of the unexplained residual variance, and so forth. The higher order Non-Standardized Principal Component Analysis and Correlation method produce the same results in this case and were able to account for oil spill detection.

Non-Standardized Principal Component Analysis can discriminate between oil and water regardless of wind speed. To avoid the disadvantage of the use of linear combinations that lead to a high correlation of the resulting textures and loosing textural information, Principal Component Analysis has been applied. The first principal component of covariance matrix, i.e. the eigenvector with the largest Eigen value corresponds to an identifiable physical property such as the size or the width of the oil, which is most important for interpretation of the patterns.

Non-Standardized Principal Component Analysis preserves the total variance and minimises the mean square approximate errors that avoid the main defect or linear fusion that produce high correlation between the fused bands.

The more eigenvectors are used in the combination the lower the error. Selecting the eigenvectors associated with the largest eigenvalues can also minimize error. The result is optimum in the sense that it minimizes the mean square error between the textures and their approximations due to using the eigenvectors corresponding to the largest eigenvalues.

The attractive result is that the first PCA method yields similar results to those of the correlation texture method, which comply with our assumption that correlation is the best method, can discriminate between water body and oil slick.

The first three Eigenvalues account for

- (File: VLEK0955h) 98.57 %
- (File: VLEK3) 97.13 %
- (File: VLEK10) 96.88 %

of the total variance in the three data sets, This means that the content of eight images from the co-occurrence matrix can be expressed by three texture matrices (images) fused in one image.

The new algorithm able to:

1. Discriminate between water body and oil regardless of wind speed.

2. Discriminate between Oil, water waves and looks alike phenomena.

3. Discriminate between Oil and other Organic Pollutants.

4. Shape the oil.

5.6. Recommendations

We recommend for applying our shape algorithm regardless the wind speed as follows:

- Step1: Image extraction and conversion it to polar then Range and angle correction for the ship motion.
- Step2: Conversion from polar to Cartesian images then average of any 10, 5 images or less.
- Step3: Adaptive filters were applied then Gamma filter was selected according to Signal to Noise Ratio.
- Step4: Co-occurrence matrix with eight textures was applied then correlation texture was selected according to (Haralick et al., 1973) to identify, describe and extract the oil, after which image features can be categorized (classified).
- Step5: Principal component analysis for texture fusion and evaluation of correlation texture as the best suitable method for oil spill detection using ship borne radar.

For any data set gamma filter is able to reduce the number of average of images to the minimum, even two images is enough except the first one of the series, so the stacking is recommended to be less, because of the power of the current algorithm SHIRA can move with any velocity.

Less than average of two images not recommended in data set 0955hh with low wind speed. Fitting using Trigonometric method is recommended.

For data set with moderate and high wind speed, ignoring step 5 is recommended because of the correlation yield excellent discrimination between Oil, water and other organic pollutants.

For data set with low wind speed, step 5 is important to improve the oil detection.

References

- Aschbacher, J., and Lichtenegger, J., 1990. Complementary nature of SAR and optical data: a case study in the Tropics. Earth Observation Quarterly, 31**,** 4- 8.

- Bloom, A., Fielding, E., and Fu, X., 1988. A demonstration of stereophotogrammetry with combined SIR-B and Landsat-TM images. International Journal of Remote Sensing, 9**,** 1023-1038.

- Chavez, P. S., Guptill, S. C., and Bowell, J. H., 1984, Image Processing Techniques for TM data. Technical Papers, Proceedings 50th Annual Meeting of ASPRS, Washington DC, U.S.A., 2**,** 728-742.

- Duguay, C.R., Holder, G. , LeDrew, E.F. ,and Howarth, P. ,1987. Integrating remotely sensed data from different sensors for change detection. Proceedings of the International Geoscience and Remote Sensing Symposium, IGARSS, 87, 567.

- Ehlers, M., 1991. Multisensor image fusion techniques in remote sensing, ISPRS Journal of Photogrammetry and Remote Sensing, 46, 19-30.

- Eliason, E.M., McEwen, A. S., 1990. Adaptive Box Filters for Removal of Random Noise from Digital Images, Photogrammetric Engineering & Remote Sensing, 56, 4, 453.

- Frost, V.S. , Stiles, J.A. , Shanmugan, K.S. , and Holtzman, J.C., 1982. A model for radar images and its application to adaptive digital filtering of multiplicative noise, IEEE Trans. Pattern Analysis and Machine Intelligence, 4, 2, 157-166.

- Genderen, J. L. van, and Pohl, C., 1994. Image fusion: Issues, techniques and applications.Intelligent Image Fusion, Proceedings EARSeL Workshop, Strasbourg, France, edited by J. L. van Genderen and V. Cappellini (Enschede: ITC),18- 26.

- Genderen, J. L. van, and Pohl, C., 1998. Multisensor image fusion in remote sensing: concepts, methods and applications, International Journal of Remote Sensing, 19, 5, 823- 854.

- Haralick, R.M., Shanmugam, K., and Dinstein, I.H., 1973.Textural Features for Image Classification, IEEE Trans. SMC, 3 (6), 610-621.

- Haralick,R.M., 1979. Statistical and Structural Approaches to Texture, Proc. IEEE, 67 (5), 786-804.

- ILWIS 3.11, 2002. International Institute for Geoinformation Science and Earth Observation user manual, 254-259.

- Kleijweg, J.C.M., Greidanus, H., Teeuwen, J.H.M., 1994. Oil spill tracking with SHIRA ship borne radar, NRSP-2, 94-20.

- Kuan, A.A. Sawchuk, T.C. Strand, and P. Chavel, 1985. Adaptive restoration of images with speckle, IEEE Trans. ASSP, 35, 3, 373-383.

- Leckie, D. G., 1990. Synergism of SAR and visible/infrared data for forest type discrimination Photogrammetric Engineering and Remote Sensing, 56, 1237- 1246.

- Lee, J.S., 1980. Digital Image Enhancement and Noise Filtering by Use of Local Statistics, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2, 2, 165-168.

- Lopes, A., Nezry, E., Touzi, 1990. Adaptive speckle filters and Scene heterogeneity, IEEE Transaction on Geoscience and Remote Sensing, Vol. 28, 6, 992-1000.

- Misset, P., 2003. ”SeaDarq Olie-detectie, ” Sea dark oil detection **“** TNO-FEL, translated by Scholte, H., 2003, ITC.

- Rogers, R. H., and Wood, L., 1990, The history and status of merging multiple sensor data an overview, ACSM-ASPRS Annual Convention, Image Processing and Remote Sensing, 4, 352-360.

- Schistad-Solberg, A. H., Jain, A. K., and Taxt, T., 1994, Multisource classification of remotely sensed data: fusion of Landsat TM and SAR images. I.E.E.E. Transactions on Geoscience and Remote Sensing, 32, 768- 778.

- Schwan, H., Wunderle, S., Desnos, Y. L., 1997. Evaluation of Speckle-Filtered ERS-1 SAR Images in Patagonia and Antarctica, ESA EOQ Nr.45, September 1995 EASI/PACE (PCI software)Version 6.2 Manuals.

- Sonka, M., Hlavac,V., and Boyle, R., 1998. Image Processing, Analysis and Machine Vision,2nd Edition was published by PWS - an Imprint of Brooks and Cole.

- Strobl, D., Raggam, J., and Buchroithner, M. F., 1990. Terrain correction geocoding of a multi-sensor image data set. Proceedings 10th EARSeL Symposium, Toulouse, France (Paris: European Space Agency), 98-107.

- Suits, G., Malila, W., and Weller, T., 1988, Procedures for using signals from one sensoras substitutes for signals of another. Remote Sensing of Environment, 25, 395- 408.

APPENDICES

APPENDIX 1

```
function General algorithm
%This is the main function
clear all;
warning off;
Location = 'D:\MATLAB6p5\work\0955hh\Data\0955hh.drq';
Angle = 'no';
RANGE = 'yes';
BNOI = 1;
ENOI = 10;
NPix = 1024;
NoIS=10;
L = -30000;
H = -17000;
min = 2;
max = 80;
```

%Step1: Image extraction and conversion it to polar then Range and angle correction for the ship motion.

%Step2: Conversion from polar to Cartesian images then average of 10 images or less.

%Step3: Adaptive filters were applied then Gamma filter was selected according to Signal to Noise Ratio.

%Step4: Co-occurrence matrix with eight textures was applied then correlation texture was selected according to (Haralick et al., 1973) to identify, describe and extract the oil, after which image features can be categorized (classified).

%Step5: Principal component analysis for texture fusion and evaluation of correlation texture as the best suitable method for oil spill detection using ship borne radar.

APPENDIX 2

```
%stap 1:data_to_polar(Location, ENOI, NoIS);
fid=fopen('D:\Matlab6p5\work\0955hh\data\0955hh.drq','r');
FIH = FIH_load(fid);
clear FIH;
ILN = 1;
LIH(ILN) = LIH_load(fid)
A_New = LIH(1).AI.AA - LIH(1).AI.NoOS;
A_Old = -1;
NPix = LIH(1).SI.NuOS;
ILN = 1;
COUNTER = 0;
IN = 1;
CH = 2;
LAI = 1:4096;
IP = zeros(4096,NPix);
while IN <= ENOI
   if A_New >= A_Old
      Line1 = fread(fid,(2*NPix),'int16');
      Line2 = Line1(CH:2:(2*NPix));
      LIH(1) = LIH_load(fid);
      while (A_New-A_Old) >= 1
         A_Old = A_Old+1;
         IP(ILN,1:(NPix)) = Line2';
         ILN = ILN + 1;
         d(1).A_New = A_New;
         d(1).A_Old = A_Old;
         d(1).ILN = ILN;
         d(1).COUNTER = COUNTER;
         d(1).IN = IN
      end
      COUNTER = COUNTER + 1;
      A_New = LIH(1).AI.AA - LIH(1).AI.NoOS;
   else
      A_Old = A_New;
      ILN;
      [OUT1, OUT2] = number (IN,NoIS);
      Location_save =['D:\MATLAB6p5\work\0955hh\polar\raw\',OUT1,'\polar_raw_'];
      save_image_IP(IP,NPix,ILN,LAI,Location_save,IN);
```

APPENDIX 3

```
function [IP] = range_angle_correction(IP,NPix,ILN,Location);
Location_Load =['D:\MATLAB6p5\work\0955hh\polar\raw\IN_\polar_raw_'];
HvG = (2*pi)/(4096);
B=ones(4096,NPix);
L1 = 1025 ;   % round((Pm1/(360/(2*pi)))/HvG);
L2 = 1168 ;   % round((Pm2/(360/(2*pi)))/HvG);
L3 = 2749 ;   % round((Pc1/(360/(2*pi)))/HvG);
L4 =3100;     % round((Pc2/(360/(2*pi)))/HvG);
V1 = 0 ;      %round((Pv1/(360/(2*pi)))/HvG);
V2 =0  ;       % round((Pv2/(360/(2*pi)))/HvG);
R1 =0 ;        % Rv1;
R2 =0 ;        % Rv2;
B(1026:1129,:)=zeros(1:(L2-L1),1:NPix);
B(2749:3036,:)=zeros(1:(L4-L3),1:NPix);
if V1==0
else
   B(349:936,135:402) =zeros(1:(V2-V1+1),1:(R2-R1+1));
end
B = (A + abs(ones(4096,NPix)*min(min(A)))).*B;
AS = sum(B')/NPix;
AS1 = AS(1,1);
AS4 = AS(1,L1-2);
AS5 = AS(1,L2+2);
AS6 = AS(1,L3-2);
AS7 = AS(1,L4+2);
AS8 = AS(1,4096);
for x=1:(L2-L1)
   AS(1,((L1-2)+x+2)) = (((AS5-AS4)/(L2-L1))*x)+AS4;
end
for x=1:(L4-L3)
   AS(1,((L3-2)+x+2)) = (((AS7-AS6)/(L4-L3))*x)+AS6;
end
x2 = 1:4096;
y2 = AS;
p2 = polyfit(x2,y2,6);
meanvecAS=mean(AS);
f2 = polyval(p2,x2);
mask_AS(1,1:4096) = f2;
for X=1:10
   x = 2^(X);
   x1 = 2^(X-1);
   mask_AS(x1+1:x,:) = mask_AS(1:x1,:);
end
```

```
IP = B - mask_AS';
A=IP ;
HvG = (2*pi)/(4096);
B=ones(4096,NPix);
L1 = 1025 ;   % round((Pm1/(360/(2*pi)))/HvG);
L2 = 1168 ;   % round((Pm2/(360/(2*pi)))/HvG);
L3 = 2749 ;   % round((Pc1/(360/(2*pi)))/HvG);
L4 =3100;     % round((Pc2/(360/(2*pi)))/HvG);
V1 = 0 ; %round((Pv1/(360/(2*pi)))/HvG);
V2 =0  ; % round((Pv2/(360/(2*pi)))/HvG);
R1 =0 ; % Rv1;
R2 =0 ;% Rv2;
B(1026:1129,:)=zeros(1:(L2-L1),1:NPix);
B(2749:3036,:)=zeros(1:(L4-L3),1:NPix);
if V1==0
else
   B(349:936,135:402) =zeros(1:(V2-V1+1),1:(R2-R1+1));
end
B = (A + abs(ones(4096,NPix)*min(min(A)))).*B;
mask_LS = sum(B)/((4096)-((L2-L1)+(L4-L3)));
for X=1:12
   x = 2^(X);
   x1 = 2^(X-1);
   mask_LS(x1+1:x,:) = mask_LS(1:x1,:);
end
IP = B - mask_LS;
```

Appendix 4

```
function summing(Location, NOI, NPix, NoIS, RANGE,HOEK)
  DB=zeros(NPix+1,NPix+1,NoIS);

NOI=NoIS;

for IN=1:NOI
   IN
   [OUT1, OUT2] = number(IN,NoIS);

   Location_load = ['D:\Matlab6p5\work\0955hh\polair\hoek&range-
corr\',OUT1,'\polair_hoek&range-corr_'];
   [IP,NPix,ILN,LAI] = load_image_IP(Location_load,IN);
   [IXY,NPix] = pol2cart(IP,NPix,LAI,3.75);

   Location_save = ['D:\Matlab6p5\work\0955hh\xy\hoek&range-corr\',OUT1,'\xy_hoek&range-
corr_'];
   save_image_IXY(NPix,IXY,Location_save,IN)
   for tel=1:(NoIS-1)
      layer = (NoIS+1)-tel;
      DB(:,:,layer) = DB(:,:,(layer-1));
   end
   DB(:,:,1) = IXY;
   DBS = zeros(NPix+1,NPix+1);
   for layer=1:NoIS
      DBS = DBS + DB(:,:,layer);
   end


   IXY = DBS/ NoIS;

   Location_save = ['D:\Matlab6p5\work\0955hh\xy\hoek&range-
corr\sum_Of_\Sum_of',num2str(NoIS),'_images51_55_'];
   save_image_IXY(NPix,IXY,Location_save,IN)

end
end
```

APPENDIX 5

```
/*function co-occurrence matrix*/
%
{
 d = distance;

  /* Determine the number of different gray scales (not maxval) */
 for (row = PGM_MAXMAXVAL; row >= 0; --row)
  tone[row] = -1;
 for (row = rows - 1; row >= 0; --row)
  for (col = 0; col < cols; ++col)
    {
 /*   if (grays[row][col])   If gray value equal 0 don't include */
      tone[grays[row][col]] = grays[row][col];
    }

for (row = PGM_MAXMAXVAL, tones = 0; row >= 0; --row)
  if (tone[row] != -1)
    tones++;
/* fprintf (stderr, "(Image has %d graylevels.)\n", tones); */

 /* Collapse array, taking out all zero values */
 for (row = 0, itone = 0; row <= PGM_MAXMAXVAL; row++)
  if (tone[row] != -1)
    tone[itone++] = tone[row];
 /* Now array contains only the gray levels present (in ascending order) */

 /* Allocate memory for gray-tone spatial dependence matrix */
 P_matrix0 = pgm_matrix (0, tones, 0, tones);
 P_matrix45 = pgm_matrix (0, tones, 0, tones);

 for (row = 0; row < tones; ++row)
  for (col = 0; col < tones; ++col)
  {
    P_matrix0[row][col] = P_matrix45[row][col] = 0;
  }

  R0 = 0;
 /* Find gray-tone spatial dependence matrix */
/* fprintf (stderr, "(Computing spatial dependence matrix..."); */

 for (row = 0; row < rows; ++row)
  for (col = 0; col < cols; ++col)
    if (grays[row][col])  /* if value anything other than zero */
```

```
      for (x = 0, angle = 0)
      {
            while (tone[x] != grays[row][col])
              x++;
            /* M. Boland if (angle == 0 && col + d < cols)  */
            /* M. Boland - include neighbor only if != 0 */
            if (angle == 0 && col + d < cols && grays[row][col + d])
            {
              y = 0;
              while (tone[y] != grays[row][col + d])
                y++;
              P_matrix0[x][y]++;
              P_matrix0[y][x]++;
              /* R0++;  M. Boland 25 Nov 98 */
              R0+=2 ;
            }

            /* M. Boland - include neighbor only if != 0 */
                        {
              y = 0;
              while (tone[y] != grays[row + d][col])
                y++;

            }

  /* Gray-tone spatial dependence matrices are complete */

  /* Find normalizing constants */
/* R0 = 2 * rows * (cols - 1)*/

  /* Normalize gray-tone spatial dependence matrix */
  for (itone = 0; itone < tones; ++itone)
    for (jtone = 0; jtone < tones; ++jtone)
    {
      P_matrix0[itone][jtone] /= R0;

    }

/*  fprintf (stderr, " done.)\n"); */
/*  fprintf (stderr, "(Computing textural features"); */
/*  fprintf (stdout, "\n"); */
/*  DOT; */
/*  fprintf (stdout,
"%s        0        Avg       Range\n",
```

```
            BL);
*/
  if (feature_usage->ASM)
            {
            ASM[0] = f1_asm (P_matrix0, tones);

            }
  Tp = &Texture->ASM[0];
  results (Tp, F1, ASM);

  if (feature_usage->contrast)
            {
            contrast[0] = f2_contrast (P_matrix0, tones);

            }
  Tp = &Texture->contrast[0];
  results (Tp, F2, contrast);

  if (feature_usage->correlation)
            {
            corr[0] = f3_corr (P_matrix0, tones);

            }
  Tp = &Texture->correlation[0];
  results (Tp, F3, corr);

  if (feature_usage->variance)
            {
            var[0] = f4_var (P_matrix0, tones);

            }
  Tp = &Texture->variance[0];
  results (Tp, F4, var);

  if (feature_usage->IDM)
            {
            idm[0] = f5_idm (P_matrix0, tones);

            }
  Tp = &Texture->IDM[0];
  results (Tp, F5, idm);

  if (feature_usage->sum_avg)
            {
```

```
        savg[0] = f6_savg (P_matrix0, tones);

        }
Tp = &Texture->sum_avg[0];
results (Tp, F6, savg);

if (feature_usage->sum_var)
        {
        sentropy[0] = f8_sentropy (P_matrix0, tones);

        }
if (feature_usage->sum_entropy)
        {
        svar[0] = f7_svar (P_matrix0, tones, sentropy[0]);

        }
Tp = &Texture->sum_var[0];
results (Tp, F7, svar);
Tp = &Texture->sum_entropy[0];
results (Tp, F8, sentropy);

if (feature_usage->entropy)
        {
        entropy[0] = f9_entropy (P_matrix0, tones);

        }
Tp = &Texture->entropy[0];
results (Tp, F9, entropy);

if (feature_usage->diff_var)
        {
        dvar[0] = f10_dvar (P_matrix0, tones);

        }
Tp = &Texture->diff_var[0];
results (Tp, F10, dvar);

if (feature_usage->diff_entropy)
        {
        dentropy[0] = f11_dentropy (P_matrix0, tones);

        }
Tp = &Texture->diff_entropy[0];
results (Tp, F11, dentropy);
```

```
 if (feature_usage->meas_corr1)
        {
        icorr[0] = f12_icorr (P_matrix0, tones);

        }
 Tp = &Texture->meas_corr1[0];
 results (Tp, F12, icorr);

 if (feature_usage->meas_corr2)
        {
        icorr[0] = f13_icorr (P_matrix0, tones);

        }
 Tp = &Texture->meas_corr2[0];
 results (Tp, F13, icorr);

 if (feature_usage->max_corr_coef)
        {
        maxcorr[0] = f14_maxcorr (P_matrix0, tones);

        }
 /* M. Boland - 24 Nov 98 */
 else {
   maxcorr[0] = 0 ;
 }

 Tp = &Texture->max_corr_coef[0];
 results (Tp, F14, maxcorr);

 for (i=0; i<=tones; i++) free(P_matrix0[i]);

 free(P_matrix0);

/*  fprintf (stderr, " done.)\n"); */
 return (Texture);
 /* exit (0);*/
}

float f1_asm (P, Ng)
 float **P;
 int Ng;

/* Angular Second Moment */
```

```
{
  int i, j;
  float sum = 0;

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      sum += P[i][j] * P[i][j];

  return sum;

  /*
   * The angular second-moment feature (ASM) f1 is a measure of homogeneity
   * of the image. In a homogeneous image, there are very few dominant
   * gray-tone transitions. Hence the P matrix for such an image will have
   * fewer entries of large magnitude.
   */
}


float f2_contrast (P, Ng)
  float **P;
  int Ng;

/* Contrast */
{
  int i, j, n;
  float sum = 0, bigsum = 0;

  for (n = 0; n < Ng; ++n)
  {
    for (i = 0; i < Ng; ++i)
      for (j = 0; j < Ng; ++j)
            if ((i - j) == n || (j - i) == n)
              sum += P[i][j];
    bigsum += n * n * sum;

    sum = 0;
  }
  return bigsum;

  /*
   * The contrast feature is a difference moment of the P matrix and is a
   * measure of the contrast or the amount of local variations present in an
   * image.
```

```
   */
}

float f3_corr (P, Ng)
  float **P;
  int Ng;

/* Correlation */
{
  int i, j;
  float sum_sqrx = 0, sum_sqry = 0, tmp, *px;
  float meanx =0 , meany = 0 , stddevx, stddevy;

  px = pgm_vector (0, Ng);
  for (i = 0; i < Ng; ++i)
    px[i] = 0;

  /*
   * px[i] is the (i-1)th entry in the marginal probability matrix obtained
   * by summing the rows of p[i][j]
   */
  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      px[i] += P[i][j];


  /* Now calculate the means and standard deviations of px and py */
  /*- fix supplied by J. Michael Christensen, 21 Jun 1991 */
  /*- further modified by James Darrell McCauley, 16 Aug 1991
   *     after realizing that meanx=meany and stddevx=stddevy
   */
  for (i = 0; i < Ng; ++i)
  {
    meanx += px[i]*i;
    sum_sqrx += px[i]*i*i;
  }
  /* M. Boland meanx = meanx/(sqrt(Ng)); */
  meany = meanx;
  sum_sqry = sum_sqrx;
  stddevx = sqrt (sum_sqrx - (meanx * meanx));
  stddevy = stddevx;

  /* Finally, the correlation ... */
  for (tmp = 0, i = 0; i < Ng; ++i)
```

```
    for (j = 0; j < Ng; ++j)
      tmp += i*j*P[i][j];

 free(px);
 return (tmp - meanx * meany) / (stddevx * stddevy);
  /*
   * This correlation feature is a measure of gray-tone linear-dependencies
   * in the image.
   */
}


float f4_var (P, Ng)
  float **P;
  int Ng;

/* Sum of Squares: Variance */
{
  int i, j;
  float mean = 0, var = 0;

  /*- Corrected by James Darrell McCauley, 16 Aug 1991
   *  calculates the mean intensity level instead of the mean of
   *  cooccurrence matrix elements
   */
  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      mean += i * P[i][j];

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      /*  M. Boland - var += (i + 1 - mean) * (i + 1 - mean) * P[i][j]; */
      var += (i - mean) * (i - mean) * P[i][j];

  return var;
}

float f5_idm (P, Ng)
  float **P;
  int Ng;


/* Sum Average */
{
```

```
  int i, j;
  extern float Pxpy[2 * PGM_MAXMAXVAL];
  float savg = 0;

  for (i = 0; i <= 2 * Ng; ++i)
    Pxpy[i] = 0;

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      /* M. Boland Pxpy[i + j + 2] += P[i][j]; */
      /* Indexing from 2 instead of 0 is inconsistent with rest of code*/
      Pxpy[i + j] += P[i][j];
  /* M. Boland for (i = 2; i <= 2 * Ng; ++i) */
  /* Indexing from 2 instead of 0 is inconsistent with rest of code*/
  for (i = 0; i <= (2 * Ng - 2); ++i)
    savg += i * Pxpy[i];

  return savg;
}


float f7_svar (P, Ng, S)
  float **P, S;
  int Ng;

/* Sum Variance */
{
  int i, j;
  extern float Pxpy[2 * PGM_MAXMAXVAL];
  float var = 0;

  for (i = 0; i <= 2 * Ng; ++i)
    Pxpy[i] = 0;

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      /* M. Boland Pxpy[i + j + 2] += P[i][j]; */
      /* Indexing from 2 instead of 0 is inconsistent with rest of code*/
      Pxpy[i + j] += P[i][j];

  /*  M. Boland for (i = 2; i <= 2 * Ng; ++i) */
  /* Indexing from 2 instead of 0 is inconsistent with rest of code*/
  for (i = 0; i <= (2 * Ng - 2); ++i)
    var += (i - S) * (i - S) * Pxpy[i];
```

```
  return var;
}

float f8_sentropy (P, Ng)
  float **P;
  int Ng;

/* Sum Entropy */
{
  int i, j;
  extern float Pxpy[2 * PGM_MAXMAXVAL];
  float sentropy = 0;

  for (i = 0; i <= 2 * Ng; ++i)
    Pxpy[i] = 0;

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      Pxpy[i + j + 2] += P[i][j];

  for (i = 2; i <= 2 * Ng; ++i)
    /*  M. Boland  sentropy -= Pxpy[i] * log10 (Pxpy[i] + EPSILON); */
    sentropy -= Pxpy[i] * log10 (Pxpy[i] + EPSILON)/log10(2.0) ;

  return sentropy;
}


float f9_entropy (P, Ng)
  float **P;
  int Ng;

/* Entropy */
{
  int i, j;
  float entropy = 0;

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      /*      entropy += P[i][j] * log10 (P[i][j] + EPSILON); */
      entropy += P[i][j] * log10 (P[i][j] + EPSILON)/log10(2.0) ;

  return -entropy;
```

```
}


float f10_dvar (P, Ng)
  float **P;
  int Ng;

/* Difference Variance */
{
  int i, j, tmp;
  extern float Pxpy[2 * PGM_MAXMAXVAL];
  float sum = 0, sum_sqr = 0, var = 0;

  for (i = 0; i <= 2 * Ng; ++i)
    Pxpy[i] = 0;

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      Pxpy[abs (i - j)] += P[i][j];

  /* Now calculate the variance of Pxpy (Px-y) */
  for (i = 0; i < Ng; ++i)
  {
    sum += i * Pxpy[i] ;
    sum_sqr += i * i * Pxpy[i] ;
    /* M. Boland sum += Pxpy[i];
    sum_sqr += Pxpy[i] * Pxpy[i];*/
  }
  /*tmp = Ng * Ng ;  M. Boland - wrong anyway, should be Ng */
  /*var = ((tmp * sum_sqr) - (sum * sum)) / (tmp * tmp); */

  var = sum_sqr - sum*sum ;

  return var;
}

float f11_dentropy (P, Ng)
  float **P;
  int Ng;

/* Difference Entropy */
{
  int i, j, tmp;
  extern float Pxpy[2 * PGM_MAXMAXVAL];
```

```
  float sum = 0, sum_sqr = 0, var = 0;

  for (i = 0; i <= 2 * Ng; ++i)
    Pxpy[i] = 0;

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
      Pxpy[abs (i - j)] += P[i][j];

  for (i = 0; i < Ng; ++i)
    /*    sum += Pxpy[i] * log10 (Pxpy[i] + EPSILON); */
    sum += Pxpy[i] * log10 (Pxpy[i] + EPSILON)/log10(2.0) ;

  return -sum;
}

float f12_icorr (P, Ng)
  float **P;
  int Ng;

/* Information Measures of Correlation */
/* All /log10(2.0) added by M. Boland */
{
  int i, j, tmp;
  float *px, *py;
  float hx = 0, hy = 0, hxy = 0, hxy1 = 0, hxy2 = 0;

  px = pgm_vector (0, Ng);
  py = pgm_vector (0, Ng);

  /*
   * px[i] is the (i-1)th entry in the marginal probability matrix obtained
   * by summing the rows of p[i][j]
   */
  for (i = 0; i < Ng; ++i)
  {
    for (j = 0; j < Ng; ++j)
    {
      px[i] += P[i][j];
      py[j] += P[i][j];
    }
  }

  for (i = 0; i < Ng; ++i)
```

```
    for (j = 0; j < Ng; ++j)
    {
      hxy1 -= P[i][j] * log10 (px[i] * py[j] + EPSILON)/log10(2.0);
      hxy2 -= px[i] * py[j] * log10 (px[i] * py[j] + EPSILON)/log10(2.0);
      hxy -= P[i][j] * log10 (P[i][j] + EPSILON)/log10(2.0);
    }

  /* Calculate entropies of px and py - is this right? */
  for (i = 0; i < Ng; ++i)
  {
    hx -= px[i] * log10 (px[i] + EPSILON)/log10(2.0);
    hy -= py[i] * log10 (py[i] + EPSILON)/log10(2.0);
  }
/*  fprintf(stderr,"hxy1=%f\thxy=%f\thx=%f\thy=%f\n",hxy1,hxy,hx,hy); */
  free(px);
  free(py);
  return ((hxy - hxy1) / (hx > hy ? hx : hy));
}

float f13_icorr (P, Ng)
  float **P;
  int Ng;

/* Information Measures of Correlation */
/* All /log10(2.0) added by M. Boland */

{
  int i, j;
  float *px, *py;
  float hx = 0, hy = 0, hxy = 0, hxy1 = 0, hxy2 = 0;

  px = pgm_vector (0, Ng);
  py = pgm_vector (0, Ng);

  /*
   * px[i] is the (i-1)th entry in the marginal probability matrix obtained
   * by summing the rows of p[i][j]
   */
  for (i = 0; i < Ng; ++i)
  {
    for (j = 0; j < Ng; ++j)
    {
      px[i] += P[i][j];
      py[j] += P[i][j];
```

```
    }
  }

  for (i = 0; i < Ng; ++i)
    for (j = 0; j < Ng; ++j)
    {
      hxy1 -= P[i][j] * log10 (px[i] * py[j] + EPSILON)/log10(2.0);
      hxy2 -= px[i] * py[j] * log10 (px[i] * py[j] + EPSILON)/log10(2.0);
      hxy -= P[i][j] * log10 (P[i][j] + EPSILON)/log10(2.0);
    }

  /* Calculate entropies of px and py */
  for (i = 0; i < Ng; ++i)
  {
    hx -= px[i] * log10 (px[i] + EPSILON)/log10(2.0);
    hy -= py[i] * log10 (py[i] + EPSILON)/log10(2.0);
  }
/*  fprintf(stderr,"hx=%f\thxy2=%f\n",hx,hxy2); */
  free(px);
  free(py);
  return (sqrt (abs (1 - exp (-2.0 * (hxy2 - hxy)))));
}

float f14_maxcorr (P, Ng)
  float **P;
  int Ng;

/* Returns the Maximal Correlation Coefficient */
{
  int i, j, k;
  float *px, *py, **Q;
  float *x, *iy, tmp;
  float f;

  px = pgm_vector (0, Ng);
  py = pgm_vector (0, Ng);
  Q = pgm_matrix (1, Ng + 1, 1, Ng + 1);
  x = pgm_vector (1, Ng);
  iy = pgm_vector (1, Ng);

  /*
   * px[i] is the (i-1)th entry in the marginal probability matrix obtained
   * by summing the rows of p[i][j]
   */
```

```
for (i = 0; i < Ng; ++i)
{
  for (j = 0; j < Ng; ++j)
  {
    px[i] += P[i][j];
    py[j] += P[i][j];
  }
}

/* Find the Q matrix */
for (i = 0; i < Ng; ++i)
{
  for (j = 0; j < Ng; ++j)
  {
    Q[i + 1][j + 1] = 0;
    for (k = 0; k < Ng; ++k)
          Q[i + 1][j + 1] += P[i][k] * P[j][k] / px[i] / py[k];
  }
}

/* Balance the matrix */
mkbalanced (Q, Ng);
/* Reduction to Hessenberg Form */
reduction (Q, Ng);
/* Finding eigenvalue for nonsymetric matrix using QR algorithm */
if (!hessenberg (Q, Ng, x, iy))
          { for (i=1; i<=Ng+1; i++) free(Q[i]+1);
            free(Q+1);
            free((char *)px);
            free((char *)py);
            free((x+1));
            free((iy+1));
            return 0.0;

          }
/* simplesrt(Ng,x); */
/* Returns the sqrt of the second largest eigenvalue of Q */
for (i = 2, tmp = x[1]; i <= Ng; ++i)
  tmp = (tmp > x[i]) ? tmp : x[i];

f = sqrt(x[Ng - 1]);

for (i=1; i<=Ng+1; i++) free(Q[i]+1);
free(Q+1);
```

```
 free((char *)px);
 free((char *)py);
 free((x+1));
 free((iy+1));

 return f;
}

float *pgm_vector (nl, nh)
  int nl, nh;
{
  float *v;
  int    i;

  v = (float *) malloc ((unsigned) (nh - nl + 1) * sizeof (float));
  if (!v)
    fprintf (stderr, "memory allocation failure (pgm_vector) "), exit (1);

  for (i=0; i<=(nh-nl); i++) v[i]=0;
  return v - nl;
}


float **pgm_matrix (nrl, nrh, ncl, nch)
  int nrl, nrh, ncl, nch;

/* Allocates a float matrix with range [nrl..nrh][ncl..nch] */
{
  int i;
  float **m;

  /* allocate pointers to rows */
  m = (float **) malloc ((unsigned) (nrh - nrl + 1) * sizeof (float *));
  if (!m)
    fprintf (stderr, "memory allocation failure (pgm_matrix 1) "), exit (1);
  m -= ncl;

  /* allocate rows and set pointers to them */
  for (i = nrl; i <= nrh; i++)
  {
    m[i] = (float *) malloc ((unsigned) (nch - ncl + 1) * sizeof (float));
    if (!m[i])
      fprintf (stderr, "memory allocation failure (pgm_matrix 2) "),
            exit (2);
```

```
    m[i] -= ncl;
  }
  /* return pointer to array of pointers to rows */
  return m;
}

void results (Tp, c, a)
  float *Tp;
  char *c;
  float *a;
{
  int i;
  float max, min;
  max = a[0];
  min = a[0];
/*  DOT;
  fprintf (stdout, "%s", c);
*/  for (i = 0; i < 4; ++i, *Tp++)
    {
    if (a[i] <= min)
            min = a[i];
    if (a[i] > max)
            max = a[i];
  /*  fprintf (stdout, "% 1.3e ", a[i]); */
    *Tp = a[i];
    }
/*  fprintf (stdout, "% 1.3e  % 1.3e\n",
    (a[0] + a[1] + a[2] + a[3]) / 4,max-min); */
  *Tp = (a[0] + a[1] + a[2] + a[3]) / 4;
  *Tp++;
  *Tp = max - min;


}

void simplesrt (n, arr)
  int n;
  float arr[];
{
  int i, j;
  float a;

  for (j = 2; j <= n; j++)
  {
```

```
    a = arr[j];
    i = j - 1;
    while (i > 0 && arr[i] > a)
    {
      arr[i + 1] = arr[i];
      i--;
    }
    arr[i + 1] = a;
  }
}

void mkbalanced (a, n)
  float **a;
  int n;
{
  int last, j, i;
  float s, r, g, f, c, sqrdx;

  sqrdx = RADIX * RADIX;
  last = 0;
  while (last == 0)
  {
    last = 1;
    for (i = 1; i <= n; i++)
    {
      r = c = 0.0;
      for (j = 1; j <= n; j++)
            if (j != i)
            {
              c += fabs (a[j][i]);
              r += fabs (a[i][j]);
            }
      if (c && r)
      {
            g = r / RADIX;
            f = 1.0;
            s = c + r;
            while (c < g)
            {
              f *= RADIX;
              c *= sqrdx;
            }
            g = r * RADIX;
            while (c > g)
```

```
            {
              f /= RADIX;
              c /= sqrdx;
            }
            if ((c + r) / f < 0.95 * s)
            {
              last = 0;
              g = 1.0 / f;
              for (j = 1; j <= n; j++)
                a[i][j] *= g;
              for (j = 1; j <= n; j++)
                a[j][i] *= f;
            }
        }
      }
    }
}


void reduction (a, n)
  float **a;
  int n;
{
  int m, j, i;
  float y, x;

  for (m = 2; m < n; m++)
  {
    x = 0.0;
    i = m;
    for (j = m; j <= n; j++)
    {
      if (fabs (a[j][m - 1]) > fabs (x))
      {
            x = a[j][m - 1];
            i = j;
      }
    }
    if (i != m)
    {
      for (j = m - 1; j <= n; j++)
            SWAP (a[i][j], a[m][j])
            for (j = 1; j <= n; j++)
              SWAP (a[j][i], a[j][m])
```

```
            a[j][i] = a[j][i];
    }
    if (x)
    {
      for (i = m + 1; i <= n; i++)
      {
            if (y = a[i][m - 1])
            {
              y /= x;
              a[i][m - 1] = y;
              for (j = m; j <= n; j++)
                a[i][j] -= y * a[m][j];
              for (j = 1; j <= n; j++)
                a[j][m] += y * a[j][i];
            }
      }
    }
  }
}

int hessenberg (a, n, wr, wi)
  float **a, wr[], wi[];
  int n;

{
  int nn, m, l, k, j, its, i, mmin;
  float z, y, x, w, v, u, t, s, r, q, p, anorm;

  anorm = fabs (a[1][1]);
  for (i = 2; i <= n; i++)
    for (j = (i - 1); j <= n; j++)
      anorm += fabs (a[i][j]);
  nn = n;
  t = 0.0;
  while (nn >= 1)
  {
    its = 0;
    do
    {
      for (l = nn; l >= 2; l--)
      {
            s = fabs (a[l - 1][l - 1]) + fabs (a[l][l]);
            if (s == 0.0)
              s = anorm;
```

```
            if ((float) (fabs (a[l][l - 1]) + s) == s)
              break;
        }
        x = a[nn][nn];
        if (l == nn)
        {
            wr[nn] = x + t;
            wi[nn--] = 0.0;
        }
        else
        {
            y = a[nn - 1][nn - 1];
            w = a[nn][nn - 1] * a[nn - 1][nn];
            if (l == (nn - 1))
            {
              p = 0.5 * (y - x);
              q = p * p + w;
              z = sqrt (fabs (q));
              x += t;
              if (q >= 0.0)
              {
                z = p + SIGN (z, p);
                wr[nn - 1] = wr[nn] = x + z;
                if (z)
                  wr[nn] = x - w / z;
                wi[nn - 1] = wi[nn] = 0.0;
              }
              else
              {
                wr[nn - 1] = wr[nn] = x + p;
                wi[nn - 1] = -(wi[nn] = z);
              }
              nn -= 2;
            }
            else
            {
              if (its == 30)
                {
/*              fprintf (stderr,
  "Too many iterations to required to find %s\ngiving up\n", F14);  */
                 return 0; /*exit (1);*/
                 }
              if (its == 10 || its == 20)
              {
```

```
  t += x;
  for (i = 1; i <= nn; i++)
    a[i][i] -= x;
  s = fabs (a[nn][nn - 1]) + fabs (a[nn - 1][nn - 2]);
  y = x = 0.75 * s;
  w = -0.4375 * s * s;
 }
 ++its;
 for (m = (nn - 2); m >= l; m--)
 {
  z = a[m][m];
  r = x - z;
  s = y - z;
  p = (r * s - w) / a[m + 1][m] + a[m][m + 1];
  q = a[m + 1][m + 1] - z - r - s;
  r = a[m + 2][m + 1];
  s = fabs (p) + fabs (q) + fabs (r);
  p /= s;
  q /= s;
  r /= s;
  if (m == l)
    break;
  u = fabs (a[m][m - 1]) * (fabs (q) + fabs (r));
  v = fabs (p) * (fabs (a[m - 1][m - 1]) +
                  fabs (z) + fabs (a[m + 1][m + 1]));
  if ((float) (u + v) == v)
    break;
 }
 for (i = m + 2; i <= nn; i++)
 {
  a[i][i - 2] = 0.0;
  if (i != (m + 2))
    a[i][i - 3] = 0.0;
 }
 for (k = m; k <= nn - 1; k++)
 {
  if (k != m)
  {
    p = a[k][k - 1];
    q = a[k + 1][k - 1];
    r = 0.0;
    if (k != (nn - 1))
        r = a[k + 2][k - 1];
    if (x = fabs (p) + fabs (q) + fabs (r))
```

```
 {
        p /= x;
        q /= x;
        r /= x;
 }
}
if (s = SIGN (sqrt (p * p + q * q + r * r), p))
{
 if (k == m)
 {
        if (l != m)
          a[k][k - 1] = -a[k][k - 1];
 }
 else
        a[k][k - 1] = -s * x;
 p += s;
 x = p / s;
 y = q / s;
 z = r / s;
 q /= p;
 r /= p;
 for (j = k; j <= nn; j++)
 {
        p = a[k][j] + q * a[k + 1][j];
        if (k != (nn - 1))
        {
          p += r * a[k + 2][j];
          a[k + 2][j] -= p * z;
        }
        a[k + 1][j] -= p * y;
        a[k][j] -= p * x;
 }
 mmin = nn < k + 3 ? nn : k + 3;
 for (i = l; i <= mmin; i++)
 {
        p = x * a[i][k] + y * a[i][k + 1];
        if (k != (nn - 1))
        {
          p += z * a[i][k + 2];
          a[i][k + 2] -= p * r;
        }
        a[i][k + 1] -= p * q;
        a[i][k] -= p;
 }
```

```
                }
              }
            }
      }
    } while (l < nn - 1);
  }
return 1;
}
```

APPENDIX 6

```
function PCA
[U, D, V] = svd(data,0)   %(Notice the data must comprise images in its columns !)
%The columns of U contains the eigenvectors of the covariance matrix of the original data, D*D' are the corresponding eigenvalues and the eigenvectors are usually sorted according to them (from the highest to the smallest.
K=3 ; % for selecting the first three PCAs.
newU = U(:,1:k); % to keep only the first three of them
% therefore the new feature vector can be formed as
Y = newU'*data  %(if the original data has the dimension of p x m, then newU is p x k, and Y is k x m), where k < p (hence data dimensionality reduction from p to k).
```