

An
Object Detection Approach
for
Cluttered Images

R. Kok

Thesis presented in partial fulfillment
of the requirements for the degree of

M. Sc. ING (E&E)

at the

UNIVERSITY OF STELLENBOSCH

Supervisor: **B.M. Herbst**

Co-supervisor: **J.G. Lourens**

December 2003

Declaration

I, the undersigned, hereby declare that this thesis and the work contained therein is my own original work, except where indicated.

Signature :

Date : November 26, 2003

Abstract

We investigate object detection against cluttered backgrounds, based on the MINACE (Minimum Noise and Correlation Energy) filter. Application of the filter is followed by a suitable segmentation algorithm, and the standard techniques of global and local thresholding are compared to watershed-based segmentation. The aim of this approach is to provide a custom region-based object detection algorithm with a concise set of regions of interest.

Two industrial case studies are examined: diamond detection in X-ray images, and the reading of a dynamic, and ink stamped, 2D barcode on packaging clutter. We demonstrate the robustness of our approach on these two diverse applications, and develop a complete algorithmic prototype for an automatic stamped code reader.

Opsomming

Hierdie tesis ondersoek die herkenning van voorwerpe teen onduidelike agtergronde. Ons benadering maak staat op die MINACE ("Minimum Noise and Correlation Energy") korrelasiefilter. Die filter word aangewend saam met 'n gepaste segmenteringsalgoritme, en die standaard tegnieke van globale en lokale drumpelingsalgoritmes word vergelyk met 'n waterskeidingsgebaseerde segmenteringsalgoritme. Die doel van hierdie deteksiebenadering is om 'n klein stel moontlike voorwerpe te kan verskaf aan enige klassifikasie-algoritme wat fokus op die voorwerpe self.

Twee industriële toepassings word ondersoek: die opsporing van diamante in X-straal beelde, en die lees van 'n dinamiese, inkgedrukte, 2D balkieskode op verpakkingsmateriaal. Ons demonstreer die robuustheid van ons benadering met hierdie twee uiteenlopende voorbeelde, en ontwikkel 'n volledige algoritmiese prototipe vir 'n outomatiese stempelkode leser.

Acknowledgements

I owe acknowledgements, academically, for my mentors:

- Prof. Ben Herbst, for his endless support and patience the last four years...
- Dr. David Weber, for introducing me to the wonderful world of DSP, caffeine and penguin power...

and personally, for my support group:

- close family, especially parents and sister, for their care...
- various co-residents of the DSP lab I have known, for making it a unique and memorable, working, environment...
- friends, for providing Quality...

Contents

1	Introduction	1
1.1	Automatic Diamond Detector	1
1.2	Reading A Stamped Code	2
1.3	Detection Approach	2
1.4	Thesis Layout	2
2	Object Detection Approaches	3
2.1	Object detection via SDF filters	3
2.2	Object detection using artificial neural networks	4
2.3	Object detection via Gabor wavelet filters	5
2.4	Object detection via quadratic Gabor filters	6
2.5	Conclusion	7
3	Correlation Filters: The Synthetic Discriminant Function	8
3.1	Background	8
3.2	Designing an SDF filter	9
3.3	Problems with the ECP SDF filter	12
3.4	Evolution of the MINACE filter	13
3.4.1	Minimum Variance SDF (MVSDf) filter	13
3.4.2	Frequency domain SDF filters	13
3.4.3	Minimum Average Correlation Energy (MACE) filter	14
3.4.4	Frequency domain MVSDf filter	15

3.4.5	Minimum Correlation Energy (MICE) filter	15
3.4.6	Minimum Noise and Correlation Energy (MINACE) filter	16
3.5	Training procedure	16
3.6	Additional SDF filter considerations	18
3.6.1	Simplifying the noise energy	18
3.6.2	False-class training	18
3.6.3	Zero-mean filters	19
3.6.4	Complex constraints	19
3.6.5	Natural vs. artificial training sets	20
3.7	Design parameter summary	20
3.7.1	Filter size	20
3.7.2	Energy minimization weight (c_{em})	21
3.7.3	Thresholds	21
3.7.4	Other considerations	21
3.8	Conclusion	22
4	Image Segmentation	23
4.1	Sample filter output	23
4.2	Global thresholding	24
4.3	Adaptive thresholding	25
4.4	Watershed-based thresholding	29
4.4.1	Approaching peak detection	30
4.4.2	Algorithm	30
4.4.3	Implementation	31
4.5	Conclusion	34
5	Case Study: X-Ray Diamond Detection	35
5.1	Source images	36
5.1.1	Overview	36

5.1.2	Diamond database	40
5.2	MINACE detection filter	43
5.2.1	Investigating filter design choices	44
5.2.2	Case studies	52
5.3	Segmentation	59
5.4	Reducing false alarms	64
5.4.1	Principle components	64
5.4.2	Contour variance	65
5.4.3	Cluster elimination	67
5.5	Final evaluation	68
5.6	Conclusion	71
6	Case Study: Reading A Stamped Code	75
6.1	Initial project stages	76
6.1.1	Introduction	76
6.1.2	Summarizing the project differences	78
6.1.3	Data collection	78
6.1.4	Training set: Modelling the squares	79
6.2	Preprocessing	81
6.3	MINACE detection filter	83
6.3.1	Parameter decisions	84
6.3.2	Determining c_{em}	87
6.3.3	Performance in clutter	88
6.3.4	Training for size variations	90
6.4	Segmentation	91
6.5	Parameter extraction and ROI verification stage	92
6.6	Detector evaluation	100
6.7	Conclusion	103

7 Conclusion	104
Bibliography	106
A Software Development	110
A.1 Vital statistics	110
A.2 Overview of new software	111

List of Figures

3.1	Simple demonstration of 2D correlation.	9
4.1	Sample MINACE filter output for comparing peak detection.	24
4.2	Example of global thresholding, for various threshold values.	25
4.3	Superimposition of region borders (after global thresholding with $t = 0.7$), on filtered output, with region peaks indicated.	25
4.4	Example of local thresholding, for various threshold values, using the mean characteristic on a 35×35 thresholding element.	26
4.5	Investigating the effect of the element size on mean-based local threshold- ing, for a threshold value of $t = 0.5$	27
4.6	Example of local thresholding, for various threshold values, using the me- dian characteristic on a 35×35 thresholding element.	27
4.7	Investigating the effect of the element size on median-based local thresh- olding, for a threshold value of $t = 0.5$	28
4.8	Example of local thresholding, for various threshold values, using the Niblack technique on a 25×25 thresholding element, with the standard deviation weight $c = 0.5$	29
4.9	Investigating the effect of the standard deviation weight c on Niblack bina- rization, for a threshold value of $t = 0.4$ and a 25×25 thresholding element.	29
4.10	Demonstration of watershed algorithm on sample MINACE filter output.	32
4.11	Thresholding the regions after the watershed algorithm by their peak val- ues, using a threshold value t of 0.7.	32
4.12	Determine ROI shapes after watershed-based thresholding: clip each ROI's grayscale histogram to top 0.5 of intensity range ($t = 0.7$).	33
4.13	Investigating other threshold values when using watershed-based thresh- olding.	34

5.1	Typical raw output of X-ray personnel scanner.	36
5.2	Close-up with several clear vertical scanning side-effects.	37
5.3	Column-wise outputs of the two channel boundary detection algorithms tested.	38
5.4	Typical side-effects of the X-ray scanning mechanism.	38
5.5	Intensity histogram of a typical X-ray photograph of a worker.	39
5.6	Illustrating the preprocessing adjustments on a raw source image.	40
5.7	Layout of the mask of diamonds in February 1999 data generation session.	41
5.8	Layout of the mask of diamonds in July 1999 data generation session.	42
5.9	Data set distribution of “large” diamonds.	43
5.10	Data set distribution of “medium” diamonds.	44
5.11	Data set distribution of “small” diamonds.	45
5.12	Components of <i>Performance</i> for “all” data set, as influenced by filter parameters.	46
5.13	<i>Performance</i> for “all” data set, as influenced by filter parameters.	48
5.14	<i>Training Set Size</i> for “all” data set, as influenced by filter parameters.	49
5.15	<i>Condition Number</i> for “all” data set, as influenced by filter parameters.	50
5.16	<i>Performance</i> for “large” data set, as influenced by filter parameters.	50
5.17	<i>Performance</i> for “medium” data set, as influenced by filter parameters.	51
5.18	<i>Performance</i> for “small” data set, as influenced by filter parameters.	51
5.19	The image used for testing the filters.	52
5.20	Positive test image output for best filter from “all” data set.	53
5.21	Positive test image outputs for energy minimization weight investigation.	54
5.22	Positive test image output for best filter from “large” data set.	55
5.23	Positive test image output for best filter from “medium” data set.	56
5.24	Positive test image outputs for best filters from “small” data set.	57
5.25	Positive test image output for test filter from “large” & “medium” combination data set.	58

5.26	Normalized space-domain representation of the chosen MINACE filter, trained with the “large” & “medium” combination data set and a c_{em} -value of 10^{-4}	58
5.27	Demonstrating global thresholding on filtered output.	60
5.28	Demonstrating mean-based local thresholding on filtered output.	60
5.29	Demonstrating Niblack local thresholding on filtered output.	61
5.30	Demonstrating watershed-based thresholding on filtered output.	62
5.31	Demonstrating controlled cluster elimination after watershed-based thresholding.	63
5.32	Comparing the performance of the four segmentation techniques for various threshold values.	63
5.33	Comparing the performance of the four segmentation techniques for various threshold values, after principle component postprocessing.	65
5.34	Comparing the performance of the four segmentation techniques for various threshold values, after principle component and contour variance postprocessing.	66
5.35	Comparing the performance of the four mentioned cluster elimination approaches after watershed-based thresholding, after principle component and contour variance postprocessing.	67
5.36	Comparing the performance of the four final segmentation techniques for various threshold values on source image “AddJuly1999/ADDtestImage4”, after principle component and contour variance postprocessing.	70
5.37	Comparing the performance of the four final segmentation techniques for various threshold values on source image “AddJuly1999/ADDtestImage3”, after principle component and contour variance postprocessing.	70
5.38	Physical results when applying the chosen MINACE filter and watershed-based segmentation with full cluster elimination and regional postprocessing.	73
6.1	A scanned stamp.	75
6.2	The stamping device.	77
6.3	Some poor quality stamps.	77
6.4	The data set part 1: Two sheets with clean background.	79
6.5	The data set part 2: Two sheets with light clutter.	80
6.6	The data set part 3: Two sheets with heavy clutter.	81

6.7	The data set part 4: Two sheets with different stamping pressure.	82
6.8	The data set part 5: Three sheets with ink degradation.	83
6.9	Preprocessing stages on an uncluttered image.	84
6.10	Preprocessing stages on a lightly cluttered image.	85
6.11	Preprocessing stages on a heavily cluttered image.	86
6.12	Filter outputs for energy minimization weight investigation.	87
6.13	Normalized space-domain representation of the chosen MINACE filter, trained with some size variation and a c_{em} -value of 0.02.	88
6.14	Applying the chosen MINACE filter in light clutter, with varying amounts of preprocessing.	89
6.15	Applying the chosen MINACE filter in heavy clutter, with varying amounts of preprocessing.	89
6.16	Performance of MINACE filter trained with different levels of square size variation.	91
6.17	Initial stages of watershed-based segmentation. The MINACE filter output is of a stamp in heavy clutter.	92
6.18	Comparison of the watershed-based segmentation technique and global thresholding, for a threshold value of 0.3.	92
6.19	The perpendicular line averaging concept, central to the Hough transform.	93
6.20	Example of a differential Hough matrix, with polar co-ordinates from the center of the ROI.	95
6.21	The search vector calculated from the differential Hough matrix, used to search for the base ROI rotation ($0^\circ - 90^\circ$).	96
6.22	Differential Hough matrix for rotation refinement of the example square, relative to the polar co-ordinate of the best detected side.	96
6.23	Radial differential Hough curves for side distance refinement of the example square, relative to the initial estimate.	97
6.24	Reading the datamatrices from a sample stamp.	99
6.25	Squares with poor edges are rejected by the verification stage.	100
6.26	Some squares in heavy clutter are not detected by global thresholding.	101
6.27	Some squares which have between 10 and 15 bit errors.	101

List of Tables

5.1	Distribution of available unique diamonds between various true-class data sets.	43
5.2	Raw matrix showing <i>Detection Rate</i> % for “all” data set, as influenced by filter parameters.	47
5.3	Parameter summary for best filter from “all” data set.	53
5.4	Parameter summary for energy minimization weight investigation.	54
5.5	Parameter summary for best filter from “large” data set.	55
5.6	Parameter summary for best filter from “medium” data set.	56
5.7	Parameter summary for best filters from “small” data set.	56
5.8	Parameter summary for test filter from “large” & “medium” combination data set.	57
5.9	Approximate execution times of the various stages of the detector, using the “AddJuly1999/ADDtestImage5” source image and a threshold value of 0.7.	69
5.10	Performance statistics for sample filters of the four segmentation techniques.	72
6.1	Filter observations for energy minimization weight investigation.	87
6.2	Bit error rates for the testing set images.	102
6.3	Comparing the execution times of the various stages of the detector when using either global thresholding or watershed-based segmentation on a 800 × 800 heavily cluttered image, using a threshold value of 0.5.	102

Notation

N, i, E_S	variables
\mathbf{x}_i, \mathbf{u}	column vectors
\mathbf{X}, \mathbf{S}	matrices
$\mathbf{N}(m, n), y_i(m, n)$	indexed matrix (discrete two-dimensional variable)
a_i	i^{th} element of column vector \mathbf{a}
\mathbf{x}_i	i^{th} column vector of matrix \mathbf{X}
$[\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_N] = \mathbf{X}$	matrix of column vectors
$\hat{\mathbf{h}}, \hat{\mathbf{N}}$	something in frequency domain
$y_i^*(m, n), \mathbf{x}^*$	complex conjugate
$\mathbf{X}^H, \hat{\mathbf{h}}^H$	Hermitian (complex conjugate transpose)
\odot	2-dimensional digital correlation

Abbreviations

2D	—	Two-Dimensional
SDF	—	Synthetic Discriminant Function (filter)
FFT	—	Fast Fourier Transform
MINACE	—	Minimum Noise and Correlation Energy (filter)
SAR	—	Synthetic Aperture Radar
ANN	—	Artificial Neural Network
MACE	—	Minimum Average Correlation Energy (filter)
MVSDF	—	Minimum Variance SDF (filter)
ECP	—	Equal Correlation Peak (filter)
c_{em}	—	energy minimization weight
MB	—	Megabyte
ROI(s)	—	Region(s) of Interest
FIR	—	Finite Impulse Response (filter)
dc (value)	—	zero index of frequency spectrum (literally: direct current)
fifo	—	first-in, first-out

Chapter 1

Introduction

This thesis is a study of the detection of objects within images. The objects considered here do not always present themselves in the same way — it is necessary to detect the objects subject to a range of deformations. Matters are further complicated when the objects appear in the presence of background clutter.

Two applications are studied; one with ill defined deformations against severe background clutter, the second with well defined deformations against some background clutter.

1.1 Automatic Diamond Detector

The first application is from the diamond industry (more specifically, De Beers Consolidated Mines Limited). They are naturally concerned about the theft of diamonds by personnel. These thefts are typically executed by concealing the diamonds on the body. Accordingly, an Automatic Personnel Diamond Detector system was developed for the detection of unauthorized possession of diamonds. The detector consists of a Scannex machine providing high resolution digital X-ray photographs of personnel. At the time when the project started, these photographs were digitally enhanced by filter banks and manually processed by an operator.

The objective of this case study is to design and build enhancement filters and location algorithms to improve, and ideally to automate, the detection of uncut diamonds hidden on a person. In general this problem can be stated as the detection of low feature, low contrast, amorphous objects against cluttered backgrounds.

1.2 Reading A Stamped Code

The second application is for the packaging industry. The stamped 2D code consists of two square structures, each containing binary data and possessing a certain rotation. These properties represent information about the product and packaging process. 2D codes are typically stamped onto the packaging material, and this material contributes to the background clutter. The codes are read with digital cameras.

The objective is to design image processing software for the location of the stamps and the accurate retrieval of the information they represent (subject to specific operational conditions).

1.3 Detection Approach

These two real-time applications need a fast, shift-invariant, rotation-invariant, and, if possible, size-invariant detection technique. Shift-invariance is always required where no a priori information is available on the location of the objects within the images. Rotation-invariance is necessary when a correct alignment of the camera with the objects cannot be guaranteed. Size-invariance can be difficult to accommodate, and, depending on the application, may be alternatively solved by filter banks designed for a range of different sizes, or a specification on the working environment.

Filter-like detection algorithms are well known and used extensively for such applications. Regions of interest (ROIs) are extracted from the output of such a detection algorithm, and, depending on the application, further verification may be done.

1.4 Thesis Layout

The next chapter overviews a number of object detection approaches, using examples from the literature. Its purpose is to motivate the choice of primary detection technique on which the detection approach is based. Chapter 3 discusses the background, theory and implementation considerations of this technique. Chapter 4 investigates the segmentation of filter outputs using a number of thresholding algorithms. Chapters 5 and 6 discuss the two case studies of diamond detection and stamped code reader, respectively.

Chapter 2

Object Detection Approaches

Pattern recognition involves the description, manipulation and recognition of pattern features toward artificial cognizance, or machine awareness. In this field, the location of objects is a basic problem. Various tools are available to address the related issues of object detection, segmentation and verification. This chapter gives an overview of some documented techniques for object detection, but is by no means exhaustive.

2.1 Object detection via SDF filters

The origin of Synthetic Discriminant Function (SDF) filters can be traced to 2D matched (or correlation) filters [1]. The SDF filter was developed to detect objects irrespective of pose, e.g. rotation or scale. In its basic form it has the shortcomings of being prone to noise and of having difficulty to shape a well-defined output peak. A more recent variant of the SDF filter, the Minimum Noise and Correlation Energy (MINACE) filter [2], addresses these problems. SDF filters are designed using standard linear algebra techniques, and the single filtering operation can be performed using the Fast Fourier Transform (FFT).

One application where the use of SDF filters was studied extensively, is the detection of vehicles in Synthetic Aperture Radar (SAR) images [3]. SAR sensors are popular for their good performance in diverse weather conditions. The output images of SAR are difficult to analyze because of the presence of speckle noise and the spurious return signal pattern, typical of radar. As with many other scene analysis applications, a rotation-invariant multi-class detection and classification system is desired.

In general the MINACE filter is valued for its detection (emphasizing regions of interest), discrimination (the design can minimize false alarms) and classification (handling multiple object classes) abilities. A highly desirable feature of SDF filters, though, is their execution speed.

2.2 Object detection using artificial neural networks

According to Haykin [4], artificial neural networks (ANNs) may be defined as follows:

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network from its environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

During the last two decades the field of neural network technology has grown large and complex. There are several network topologies and techniques available with which a network's synaptic weights are trained for an application. The neural network has become a major tool for pattern recognition.

Briefly consider a problem from the medical field, namely the detection of lung cancer nodules in radiographic images (almost identical to the problem of mammogram analysis). Since the consequence of a false rejection is serious, this application has great potential for assisting radiologists with their diagnosis. The low contrast images and the nature of the objects make this a difficult problem (similar to the diamond detection application discussed in this thesis).

One proposed solution [5] to this problem is implemented in two neural network based phases, and takes as input a preprocessed image of the lung area. One of the tasks of the preprocessing is to improve the contrast of the image.

Both neural networks use a multilayer feedforward architecture and sigmoid activation functions, and are trained using the backpropagation algorithm. The first phase's ANN

locates suspected nodule areas from a low resolution version of the image. The training of this network requires the construction of a “desired output image”. The second phase’s ANN verifies the suspected nodule areas, utilizing the curvature peak of the nodule surface as an effective discerning feature space. This network is trained in two phases (using documented additions to the backpropagation algorithm), to facilitate a fine learning process.

The use of ANN technology is motivated by its generalization property and by its capability to learn from training data. The article [5] reports satisfactory results, i.e. good detection and few enough false positives that they will not upset the radiologist.

Although ANNs are very useful, it is difficult to predict how well they will work in any given application, and the implementation of ANN detection systems can differ greatly between applications. Also, using ANN technology on its own has the disadvantage of a relatively high processing cost. Where speed is important, it might be more viable to use neural networks only for a verification stage.

2.3 Object detection via Gabor wavelet filters

Gabor filters were originally proposed to model the spatial-frequency-selective and the orientation-selective properties of the visual cortex [6]. This family of filters lends itself naturally to the formation of filter banks. Such filter banks are often applied to detect objects at different directional poses.

A common military problem is the detection of vehicles in cluttered landscapes, using infrared imaging. An article [7] on this problem compares the performance of three distortion- and shift-invariant algorithms, all of them based on Gabor wavelet filter technology. The comparison is done on a TRIM database of infrared images. This database contains 158 different objects, each in a large range of distortions, rotations and representations. The following algorithms are discussed:

1. The Gabor Wavelet Transform algorithm is a combination of a macro real object detection filter (a nonlinear sum of six real Gabor functions where the Gabor function parameters and combination coefficients are determined by neural network optimization), a horizontal imaginary Gabor function and a clutter Gabor filter.
2. The Gabor Basis Function algorithm is synthesized much like an SDF filter, using

a linear combination of Gabor functions representing a number of objects. A shape is specified for the desired correlation output plane, and the mean squared error in this shape is minimized. The correlation plane energy due to a selection of typical clutter particles is also minimized.

3. The Morphological Wavelet Transform (MWT) algorithm is basically a weighted subtraction between the output of a close minus open morphological filter, and that of a thresholded Gaussian smoothed Gabor wavelet clutter map pattern. The result of this is thresholded and dilated to yield the detected objects. Robust parameter selection criteria have been devised to improve the results.

Of these three techniques, the MWT algorithm (employing a clutter map) is reported [7] to perform the best. The various combinations between the MWT algorithm and the other two algorithms yield good results, invariably better than the MWT algorithm alone.

A large amount of work concerning evolved variations on Gabor filter technology is utilized in this example. It is of special interest to see the lengths to which it is possible to combine the outputs of different detection algorithms. The focus of Gabor filters does not seem suitable for the specific objects addressed in this thesis. Nevertheless, there are numerous techniques with which a detector can be shaped around Gabor filter technology.

2.4 Object detection via quadratic Gabor filters

The linear decision surface of correlation filters can be improved using a quadratic surface, allowing better detection and classification. An article [8] that uses such an improvement on the Gabor filter class, also deals with the problem of vehicle detection. This article describes the design of a quadratic Gabor filter using several macro filters, where each macro filter is a linear combination of several separately designed linear Gabor basis filters. A large standardized TRIM-2 infrared vehicle database is available for the selection of object and clutter databases.

The proposed filter is build around an extended piecewise quadratic neural network. The inputs of the network are the resulting filtered output image plane pixels from each of six Gabor basis filters. These Gabor filters are designed for specific object features (using both directional blob detecting real Gabor functions and directional edge detecting imaginary Gabor functions), using a parameter optimization algorithm. The network's

function, therefore, is to determine the optimum parameters for the quadratic combination of the Gabor filters' outputs. Built into this network is a possible complexity reduction, controlled by the number of nodes in the second (and middle) layer of the network. This option allows the Gabor filters to be linearly combined into a smaller number of macro filters.

The quadratic Gabor filter outperforms a Gabor basis function filter [9] on the same problem. This work's significance is in its successful improvement of the linear decision surfaces associated with standard correlation filters.

In conclusion, this technique depends on the availability of several object features from which independent detection filters can be designed. Although this does not seem suitable for initiating a general approach for multiple applications, improving the decision surface of detectors could be very valuable for some applications.

2.5 Conclusion

The SDF filter class is chosen to address the two problems studied in this thesis. The following are the reasons for this choice:

- It is a robust and general purpose algorithm, originating from 2D correlation technology.
- It is a mature filter class — the main shortcomings of the original SDF filter has been solved with the later MINACE filter.
- It is very fast — the filter is implemented using a single correlation, and this can be done efficiently using FFT's.
- Relative ease of design — once the detection technique is familiar, a detector for a new application can be designed with minimal changes.

Chapter 3

Correlation Filters: The Synthetic Discriminant Function

This chapter provides a brief theoretical overview of SDF filters, leading up to the MI-NACE filter and the practical considerations of its implementation. Most of the theory discussed here can be found in the classic tutorial by Kumar [10].

3.1 Background

The basic equation for a 2D correlation filter is given by

$$y(m_{cc}, n_{cc}) = \{a \odot b\}(m_{cc}, n_{cc}) := \sum_m \sum_n a^*(m + m_{cc}, n + n_{cc}) b(m, n), \quad (3.1)$$

where the indexed variables $a(m, n)$ and $b(m, n)$ are the two images to be correlated and y is the output.

Figure 3.1 shows an example of 2D correlation, using the two 15×15 images a and b . The dimensions of the output images are twice the input dimensions minus one. Since an image is perfectly correlated with itself, it should have a prominent correlation peak. Figure 3.1 shows this in the autocorrelation of a , and also shows that the best match between a and b is much weaker, as expected.

When an object class is to be detected (i.e. when the object can assume different poses), the performance of correlation deteriorates as rapidly as the variance of the objects in the class increases. SDF filters were created to overcome this limitation.

The motivation behind SDF filters is to design a generalized correlation filter to give a specified response at the origin for a set of related objects, or different poses of the same object. Therefore each object (or deformation of an object) belonging to a class is identified by its filter response at the origin.

3.2 Designing an SDF filter

Consider N training images, $x_i(m, n)$, $i = 1, 2, \dots, N$, and a yet unknown filter $h(m, n)$. The equation for the correlation between the filter and the training images is

$$y_i(m_{cc}, n_{cc}) = (x_i \odot h)(m_{cc}, n_{cc}) := \sum_m \sum_n x_i^*(m + m_{cc}, n + n_{cc}) h(m, n). \quad (3.2)$$

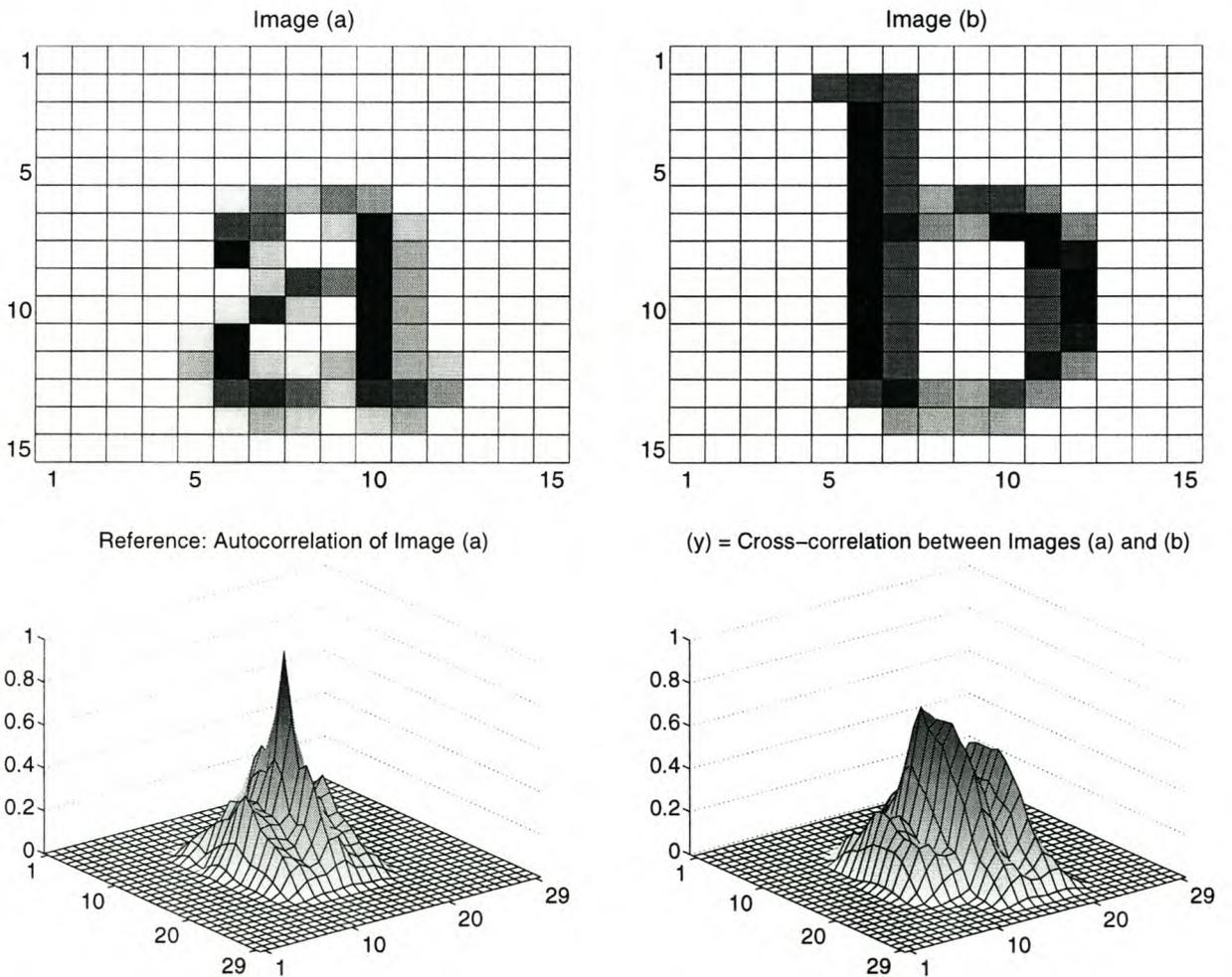


Figure 3.1 Simple demonstration of 2D correlation.



Lacking enough degrees of freedom, it is not possible to design a filter that gives a specified response for the whole correlation plane for all N images in the training set. The filter response is therefore constrained only at the origin [1], and this is written as

$$u_i := (x_i \odot h)(0, 0) = \sum_m \sum_n x_i^*(m, n) h(m, n) \quad (3.3)$$

for all images $i = 1, 2, \dots, N$, where u_i specifies the correlation response of image x_i at the origin (typically, but not necessarily, chosen as 1 for all true-class images).

For brevity, the filter and the training images are lexicographically ordered into vector form, and the constraint is rewritten in matrix notation as $\mathbf{x}_i^H \mathbf{h} = u_i$. Defining $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_N]$, the constraint equation becomes

$$\mathbf{X}^H \mathbf{h} = \mathbf{u}, \quad (3.4)$$

where $\mathbf{u}^T = [u_1 \ u_2 \ \dots \ u_N]$. This is an underdetermined system, allowing an infinite number of solutions. Assuming that \mathbf{h} is a linear combination (with weights a_i) of the N training images, i.e.

$$\mathbf{h} = \mathbf{X} \mathbf{a}, \quad (3.5)$$

one obtains a unique solution. This is standard linear algebra [11], but here it also has an intuitive meaning: in the original optical application, the filter had to be synthesizable by multiple-exposure techniques [1]. Finding \mathbf{a} by cancelling \mathbf{h} from equations (3.4) and (3.5), the filter equation becomes

$$\mathbf{h}_{ECP} = \mathbf{X} (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{u}. \quad (3.6)$$

The inverse $(\mathbf{X}^H \mathbf{X})^{-1}$ exists if and only if the training images are independent (\mathbf{X} is of full rank). This original design is known as the equal correlation peak (ECP) SDF filter.

If condition (3.5) is dropped, more possibilities become available. Returning to the constraint equation (3.4), the length d of the vector \mathbf{h} (and of the $d \times N$ matrix \mathbf{X}) is much bigger than the length N of the vector \mathbf{u} . There are N equations and d variables, which means there are $d - N$ free variables (for a real constraint vector \mathbf{u}) which can be used to achieve various other design objectives. A general expression [12] for any \mathbf{h} that can satisfy (3.4) consists of a particular (\mathbf{h}_p) and a homogeneous (\mathbf{h}_0) solution, and is written as

$$\mathbf{h}_{SDF} = \mathbf{h}_p + \mathbf{h}_0. \quad (3.7)$$

A particular solution is already available in (3.6), and the homogeneous solutions are calculated from

$$\mathbf{X}^H \mathbf{h}_0 = \mathbf{0}. \quad (3.8)$$

This equation says that the solutions \mathbf{h}_0 are perpendicular to the rows of \mathbf{X}^H , or the columns of \mathbf{X} . Therefore, any vector \mathbf{z} can be written as

$$\mathbf{z} = \mathbf{X}\mathbf{r} + \mathbf{h}_0. \quad (3.9)$$

The vector \mathbf{r} is determined by premultiplying (3.9) with \mathbf{X}^H and using (3.8) to yield

$$\begin{aligned} \mathbf{X}^H\mathbf{z} &= \mathbf{X}^H\mathbf{X}\mathbf{r} + \mathbf{X}^H\mathbf{h}_0 \\ &= \mathbf{X}^H\mathbf{X}\mathbf{r}, \end{aligned} \quad (3.10)$$

or $\mathbf{r} = (\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H\mathbf{z}$. It therefore follows that

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{z} - \mathbf{X}\mathbf{r} \\ &= \mathbf{z} - \mathbf{X}(\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H\mathbf{z} \\ &= \left[\mathbf{I}_d - \mathbf{X}(\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H \right] \mathbf{z}, \end{aligned} \quad (3.11)$$

where \mathbf{I}_d is an identity matrix sized $d \times d$. The general solution is now

$$\mathbf{h}_{SDF} = \mathbf{X}(\mathbf{X}^H\mathbf{X})^{-1}\mathbf{u} + \left[\mathbf{I}_d - \mathbf{X}(\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H \right] \mathbf{z}, \quad (3.12)$$

where \mathbf{z} is arbitrary.

A goal-oriented solution to the SDF filter is possible by exploiting the freedom provided by (3.4) and its general solution (3.12). To this end, the filter is redesigned to minimize the function

$$f(\mathbf{h}) = \mathbf{h}^H\mathbf{M}\mathbf{h} \quad (3.13)$$

subject to the constraint in (3.4). For the time being, \mathbf{M} can be any $d \times d$, symmetric, positive definite matrix designed specifically to improve certain properties of the filter. A basic choice for \mathbf{M} is \mathbf{I}_d , so that $f(\mathbf{h}) = \mathbf{h}^H\mathbf{h}$ (the filter energy) is minimized. Other choices for \mathbf{M} are discussed in Section 3.4.

The solution to this problem of constrained optimization is obtained by using the method of Lagrange multipliers [13, 14]. The Lagrange variables are introduced into (3.13), yielding the new function to be minimized as

$$F(\mathbf{h}, \boldsymbol{\lambda}) = \mathbf{h}^H\mathbf{M}\mathbf{h} - \boldsymbol{\lambda}^H(\mathbf{u} - \mathbf{X}^H\mathbf{h}), \quad (3.14)$$

where $\boldsymbol{\lambda}^T = [\lambda_1 \lambda_2 \dots \lambda_N]$. To solve $\nabla F = 0$ with respect to \mathbf{h} , the partial derivative $\nabla_{\mathbf{h}}F = 0$ yields $2\mathbf{h} - \mathbf{X}\boldsymbol{\lambda} = 0$ (because \mathbf{M} is symmetric), and therefore

$$\mathbf{h} = \mathbf{M}^{-1}\mathbf{X} \left(\frac{1}{2}\boldsymbol{\lambda} \right). \quad (3.15)$$

To solve for λ , (3.15) is premultiplied by \mathbf{X}^H and inserted into the constraint equation in (3.4):

$$\mathbf{X}^H \mathbf{h} = \mathbf{X}^H \mathbf{M}^{-1} \mathbf{X} \left(\frac{1}{2} \lambda \right) = \mathbf{u}. \quad (3.16)$$

This yields $\frac{1}{2} \lambda = (\mathbf{X}^H \mathbf{M}^{-1} \mathbf{X})^{-1} \mathbf{u}$, and substituting into (3.15) results in

$$\mathbf{h} = \mathbf{M}^{-1} \mathbf{X} (\mathbf{X}^H \mathbf{M}^{-1} \mathbf{X})^{-1} \mathbf{u}. \quad (3.17)$$

It is clear that the ECP SDF filter is the result of choosing $\mathbf{M} = \mathbf{I}_d$, i.e. minimizing the energy in the filter subject to the origin correlation peak constraints in (3.4).

3.3 Problems with the ECP SDF filter

There are two significant problems with the ECP SDF filter:

Origin uncertainty: Since the filter is only constrained for a certain origin response, the rest of the output correlation plane (i.e. where the filter and true class object are not exactly aligned) is undefined. Although the energy in the filter is minimized, the origin response of an object will not necessarily be a maximum, and object detection using peak searches may be unreliable. This happens when the correlation sidelobes¹ are uncontrolled.

Noise and intra-class recognition: The expected filter response is easily changed by both noise and the implied distortion when detecting an object not in the training set. This means that origin responses can be very different from the constrained value, and peaks can easily not be where they would have been.

Fortunately, there is considerable freedom in the choice of filter parameters, while still maintaining (3.4). This freedom can be exploited further to overcome the problems mentioned above.

¹“Sidelobe” in this context should not be confused with the similarly named frequency domain effect of time domain windowing. Here, the sidelobe problem refers to the effect of a centered correlation resulting in a slope of a peak instead of a peak, and the formation of other correlation peaks (like mountain foothills) found when the filter is only partly over the desired object.

3.4 Evolution of the MINACE filter

While there are numerous variants of the SDF filter, this discussion traces only the immediate predecessors of the MINACE filter to focus on the reasons for its development.

3.4.1 Minimum Variance SDF (MVSDF) filter

The MVSDF filter [10, 15] is a straightforward enhancement of the ECP SDF filter. It minimizes the variance in the filter output caused by input noise. This effectively decreases the filter's sensitivity to object distortions, because intraclass distortions can be modeled as additive noise.

Consider the training image \mathbf{x}_i distorted by additive noise \mathbf{n} (a discrete noise process). The resulting output cross-correlation value with some filter \mathbf{h} designed to meet the constraint in (3.4) is then

$$y_i = \mathbf{h}^H (\mathbf{x}_i + \mathbf{n}) = u_i + \mathbf{h}^H \mathbf{n}. \quad (3.18)$$

Assume that the noise has zero mean and covariance \mathbf{C} . Its energy in the output correlation plane is the variance of y_i , expressed as

$$\sigma_{y_i}^2 = E\{|\mathbf{h}^H \mathbf{n}|^2\} = E\{\mathbf{h}^H \mathbf{n} \mathbf{n}^H \mathbf{h}\} = \mathbf{h}^H \mathbf{C} \mathbf{h}. \quad (3.19)$$

The equation in (3.19) can be minimized giving the same solution as (3.13) (if \mathbf{M} is substituted with \mathbf{C}), because the covariance matrix \mathbf{C} is positive definite and symmetric. The filter equation is:

$$\mathbf{h}_{MVSDF} = \mathbf{C}^{-1} \mathbf{X} (\mathbf{X}^H \mathbf{C}^{-1} \mathbf{X})^{-1} \mathbf{u}. \quad (3.20)$$

For the special case of white noise [10] ($\mathbf{C} = \sigma^2 \mathbf{I}_d$), $\mathbf{h}_{MVSDF} = \mathbf{h}_{ECP}$.

3.4.2 Frequency domain SDF filters

The advantage of frequency domain SDF filters is that certain parameters can be minimized which are not as accessible in the time domain. A number of filters based on this approach has been introduced. These filters tend to focus on gaining more control over the rest of the correlation plane, thus avoiding the sidelobe problem.

The transition to the design of SDF filters in the frequency domain is made using Parseval's theorem, which allows (3.4) to be written in the frequency domain as

$$\hat{\mathbf{X}}^H \hat{\mathbf{h}} = \mathbf{v}, \quad (3.21)$$

where $\hat{\mathbf{s}}$ is the Fourier Transform of \mathbf{s} , $\mathbf{v} = d\mathbf{u}$ where d is the appropriate scaling constant associated with the Fourier Transform and $\hat{\mathbf{X}}$ contains the vectors $\hat{\mathbf{x}}_i$ which are the vectorized Fourier transforms of the training images $x(i, j)$. The filter equation takes the same form as (3.17) and is

$$\hat{\mathbf{h}} = \hat{\mathbf{M}}^{-1} \hat{\mathbf{X}} \left(\hat{\mathbf{X}}^H \hat{\mathbf{M}}^{-1} \hat{\mathbf{X}} \right)^{-1} \mathbf{v}, \quad (3.22)$$

where $\hat{\mathbf{M}}$ is again an arbitrary, positive definite, symmetric matrix used to meet specific design criteria.

3.4.3 Minimum Average Correlation Energy (MACE) filter

The MACE filter [10, 16] tries to suppress the sidelobes by minimizing the average correlation plane energy in the filter output caused by the class object. This results in sharper correlation peaks and lower responses near the peak.

Using the convolution theorem, the Fourier transform of the correlation between \mathbf{x}_i and \mathbf{h} can be written as $\hat{\mathbf{h}}^H \hat{\mathbf{D}}_i$, where the matrix $\hat{\mathbf{D}}_i$ is defined with $\hat{\mathbf{x}}_i$ on its diagonal and zero elsewhere. The correlation plane energy caused by input image i is defined as

$$E_i = \left| \hat{\mathbf{h}}^H \hat{\mathbf{D}}_i \right|^2 = \hat{\mathbf{h}}^H \hat{\mathbf{D}}_i \hat{\mathbf{D}}_i^H \hat{\mathbf{h}} = \hat{\mathbf{h}}^H \hat{\mathbf{S}}_i \hat{\mathbf{h}}, \quad (3.23)$$

where $\hat{\mathbf{S}}_i = \hat{\mathbf{D}}_i \hat{\mathbf{D}}_i^H$ is the matrix with the absolute square of the units of $\hat{\mathbf{x}}_i$ (power spectral density) on its diagonal. It follows that

$$\begin{aligned} E_{avg} &= \frac{1}{N} \sum_{i=0}^N E_i \\ &= \hat{\mathbf{h}}^H \left\{ \frac{1}{N} \sum_{i=0}^N \hat{\mathbf{S}}_i \right\} \hat{\mathbf{h}} \\ &= \hat{\mathbf{h}}^H \hat{\mathbf{S}} \hat{\mathbf{h}}, \end{aligned} \quad (3.24)$$

where $\hat{\mathbf{S}} = \frac{1}{N} \sum_{i=0}^N \hat{\mathbf{S}}_i$. Since $\hat{\mathbf{S}}$ is positive definite and symmetric, the Lagrange multiplier solution to the minimization of E_{avg} , constrained by (3.21), is

$$\hat{\mathbf{h}}_{MACE} = \hat{\mathbf{S}}^{-1} \hat{\mathbf{X}} \left(\hat{\mathbf{X}}^H \hat{\mathbf{S}}^{-1} \hat{\mathbf{X}} \right)^{-1} \mathbf{v}. \quad (3.25)$$

3.4.4 Frequency domain MVSDF filter

Using the same technique as with the MACE filter, the Fourier transform of the correlation between the noise process \mathbf{n} and the filter \mathbf{h} can be written as $\hat{\mathbf{h}}^H \hat{\mathbf{N}}'$, where $\hat{\mathbf{N}}'$ is defined with the expected noise frequency spectrum $\hat{\mathbf{n}}$ on its diagonal and zero elsewhere. Therefore the correlation plane energy caused by the noise is

$$E = \left| \hat{\mathbf{h}}^H \hat{\mathbf{N}}' \right|^2 = \hat{\mathbf{h}}^H \hat{\mathbf{N}}' \hat{\mathbf{N}}'^H \hat{\mathbf{h}} = \hat{\mathbf{h}}^H \hat{\mathbf{N}} \hat{\mathbf{h}}, \quad (3.26)$$

where $\hat{\mathbf{N}} = \hat{\mathbf{N}}' \hat{\mathbf{N}}'^H$ is the matrix with the absolute square of the units of $\hat{\mathbf{n}}$ on its diagonal. $\hat{\mathbf{N}}$ is positive definite and symmetric, and it follows that the filter equation is

$$\hat{\mathbf{h}}_{MVSDF} = \hat{\mathbf{N}}^{-1} \hat{\mathbf{X}} \left(\hat{\mathbf{X}}^H \hat{\mathbf{N}}^{-1} \hat{\mathbf{X}} \right)^{-1} \mathbf{v}. \quad (3.27)$$

3.4.5 Minimum Correlation Energy (MICE) filter

The MICE filter [2] is a variation of the MACE filter. In the equation for the energy to be minimized (previously (3.24)), instead of using the average correlation plane energy, it imposes an upper bound on the correlation plane energy caused by the class object. Ravichandran and Casasent [2] believe that minimizing the average energy provides poor control over the possible variance of the correlation plane energies.

The new expression for the energy minimization is designed to be of the same form as the expression for E_i in (3.23). Since the upper bound satisfies

$$E_i \leq E_{max}, \quad i = 1, 2, \dots, N, \quad (3.28)$$

$\hat{\mathbf{S}}$ is redefined so that $E_{max} = \hat{\mathbf{h}}^H \hat{\mathbf{S}} \hat{\mathbf{h}}$. Using (3.28) and (3.23) it follows that

$$\hat{\mathbf{S}}_i \leq \hat{\mathbf{S}}, \quad i = 1, 2, \dots, N. \quad (3.29)$$

Recall that matrix $\hat{\mathbf{S}}_i$ is defined to have the spectral density of image \mathbf{x}_i on its diagonal. Since a spectral density function is real and positive, a good choice of $\hat{\mathbf{S}}$ is simply

$$\hat{\mathbf{S}}(k, k) = \max \left[\hat{\mathbf{S}}_1(k, k), \hat{\mathbf{S}}_2(k, k), \dots, \hat{\mathbf{S}}_N(k, k) \right]. \quad (3.30)$$

The expression for the filter is as before, i.e.

$$\hat{\mathbf{h}}_{MICE} = \hat{\mathbf{S}}^{-1} \hat{\mathbf{X}} \left(\hat{\mathbf{X}}^H \hat{\mathbf{S}}^{-1} \hat{\mathbf{X}} \right)^{-1} \mathbf{v}. \quad (3.31)$$

The MICE filter is not well known, because it was introduced in the same article as the MINACE filter for the express purpose of improving the MACE filter's energy minimization technique. This improvement is important for the design of a better combination between the MVSDF and MACE filters than if the energies for minimization are simply added together.

3.4.6 Minimum Noise and Correlation Energy (MINACE) filter

The MINACE filter [2] is designed using a weighted minimization of the correlation plane energies caused by the input noise, and by the class object. It is therefore a combination of the ideas of the MVSDF and MACE filters. Instead of using average energy however, it uses the approach of the MICE filter to minimize the maximum energy.

In accordance with the design goals, the upper bound for the energy to be minimized is written as

$$\{E_i, \sigma^2\} \leq E_{max}, \quad i = 1, 2, \dots, N, \quad (3.32)$$

with the variance, or noise energy, coming from (3.19). If $\hat{\mathbf{T}}$ is defined so that $E_{max} = \hat{\mathbf{h}}^H \hat{\mathbf{T}} \hat{\mathbf{h}}$, and (3.27) is used together with the design procedure of the MICE filter in Section 3.4.5, it follows that a good choice for $\hat{\mathbf{T}}$ is

$$\hat{\mathbf{T}}(k, k) = \max \left[\hat{\mathbf{S}}_1(k, k), \hat{\mathbf{S}}_2(k, k), \dots, \hat{\mathbf{S}}_N(k, k), c_{em} \hat{\mathbf{N}}(k, k) \right], \quad (3.33)$$

where c_{em} is the weight parameter with which the emphasis of the minimization can be adjusted. This yields the filter synthesis equation

$$\hat{\mathbf{h}}_{MINACE} = \hat{\mathbf{T}}^{-1} \hat{\mathbf{X}} \left(\hat{\mathbf{X}}^H \hat{\mathbf{T}}^{-1} \hat{\mathbf{X}} \right)^{-1} \mathbf{v}. \quad (3.34)$$

When combining the two energies for minimization with this spectral envelope technique (as opposed to a simple linear combination as discussed in [2,10]), a possible minimization bias toward either of the two energies is prevented [2]. Furthermore, $\hat{\mathbf{T}}$ can be seen as using the spectra of both the signal and the noise, but each emphasized at the spectral frequencies best suited to its strengths.

3.5 Training procedure

The standard training procedure of SDF filters is about finding a representative subset of the available true-class training set [10]. The desired subset, when used to calculate a

filter, enables the whole training set to satisfy the specified origin correlation peak value within an error margin. This error margin, specified as a threshold origin correlation peak value, is an important training parameter.

The training begins by selecting one arbitrary true-class training image for the training subset in matrix $\hat{\mathbf{X}}$, and calculating a filter $\hat{\mathbf{h}}$ from this initial subset. If the inner product (origin correlation) of this filter with each remaining image in the training set satisfies the training threshold, training is complete. Otherwise, the training image with the lowest origin correlation value is added to the training subset, and the procedure is repeated.

This training procedure minimizes the number of images used to calculate the filter. This maximizes the filter's degrees of freedom, which allows it to generalize better to new objects.

Care should be taken with the filter solution obtained from the $\hat{\mathbf{T}}$ -matrix (3.34). Poorly designed training sets can result in the filter being a badly conditioned matrix, leading to unexpectedly poor results.

The following details are also relevant:

1. It is important to select a training set carefully. All relevant features of the object should be represented, and an understanding of the background's significance may be included in the design of the filter (more detail in Section 3.6.5). A good guideline is to observe the experimental results and learn from them.
2. The true-class training images should be well centered, because the filter's training procedure relies only on *origin* correlation values.
3. The first training image affects the iterative creation of the training subset, and therefore also the filter's character and, eventually, performance. Starting with one of the more difficult training images is the guideline for creating a better filter.

All of these points emphasize the important role intuition plays in creating an SDF filter.

3.6 Additional SDF filter considerations

3.6.1 Simplifying the noise energy

One of the main functions of the noise minimization is to improve intraclass recognition. A white Gaussian noise spectrum is known to be a reasonable model for the purpose of intraclass distortions [2, 3], and simplifies the construction of (3.33).

A white Gaussian noise spectrum, with variance σ^2 , is written as $\sigma^2 \mathbf{I}_d$ (where \mathbf{I}_d is an identity matrix). Since the dc value of the noise spectrum is normalized to that of $\hat{\mathbf{S}}$, i.e. $\sigma^2 = \hat{\mathbf{S}}(1, 1)$, the noise spectrum is simplified to

$$\hat{\mathbf{N}} = \hat{\mathbf{S}}(1, 1) \mathbf{I}_d. \quad (3.35)$$

For normal grayscale training images, $\hat{\mathbf{S}}(1, 1)$ is the element of the spectrum with the highest energy content. Therefore, c_{em} need only be varied between 0 and 1 to cover its full range of effect. In practise it is usually more useful at lower values such as 0.001.

The alternative to this simplification is to generate biased and correlated noise from the application's typical clutter [17].

3.6.2 False-class training

False-class training is facilitated by constraining the origin correlation value of the images in the false-class to the least significant correlation value, i.e. 0. The matrix $\hat{\mathbf{X}}$ contains all the images from the various classes to be constrained, and the constraint vector \mathbf{u} is set to the appropriate values. Neither the constraint equation (3.4) nor any other design equation changes in the process.

The only difference in the training procedure when false-class images are used, is in the initialization of the training loop. When the first, arbitrary, true-class training image is selected, all of the false-class training images are also added to the training subset in $\hat{\mathbf{X}}$.

In practise, false-class training is useful when the characteristic clutter structures can be summarized in a few false-class training images [18]. The use of too many false-class training images diverges from the objective of the training procedure to minimize the number of images in the training set (Section 3.5). The decrease in degrees of freedom

(and poorer generalization) from the addition of these false-class training images can become more significant than the improvement in clutter rejection.

3.6.3 Zero-mean filters

When the additive noise corrupting the filter input does not have zero mean, the noise can be expressed as the sum of a deterministic mean vector \mathbf{m} and the zero mean random vector \mathbf{n} from Section 3.4.1 [10]. The effect of \mathbf{m} can be eliminated by constraining $\mathbf{h}^H \mathbf{m} = 0$. This is a false-class constraint, and is simply included in \mathbf{X} and \mathbf{u} (see Section 3.6.2).

This technique is useful for eliminating the effect of a constant background level present in the filter input, and is similar to using a high-pass filter before the correlation filter. The value used for the mean vector can be calculated as the mean intensity of the pixels in the training set.

3.6.4 Complex constraints

There is nothing that prevents the constraint vector \mathbf{u} from using complex values, since this can increase the free variables in the system of linear equations defined by the constraint equation (3.4). It has been shown that properly designed phases can benefit the minimized energy [10]. Only the magnitude of the filter output is then considered for classification. Such a filter may be unusually susceptible to clutter though, because a full circle (instead of a point) in the complex plane is now the desired output.

One specific use for complex constraints is to increase discrimination when there is a problem with images in the convex hull of the training set [19]. Per definition, the convex hull of the set of points described by $\{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_M\}$, where M is the number of images in the eventual training set, is the locus of all points

$$\mathbf{x} = \sum_{i=1}^M \lambda_i \mathbf{x}_i, \quad (3.36)$$

where $0 \leq \lambda_i$ and $\sum_{i=1}^M \lambda_i = 1$. The discrimination problem is demonstrated by calculating the filter output for any combinational image like \mathbf{x} in (3.36):

$$\mathbf{h}^H \mathbf{x} = \sum_{i=1}^M \lambda_i \mathbf{h}^H \mathbf{x}_i = \sum_{i=1}^M \lambda_i = 1. \quad (3.37)$$

This problem can be significant when the class object is rich in rotation-dependent features and rotation-invariance is specifically trained.

3.6.5 Natural vs. artificial training sets

Completely artificial training sets, i.e. where both the object and background are modeled, are useful for relatively simple, geometric object shapes where the expected distortions (other than rotation and scaling) can either be modeled or safely ignored. The overwhelming advantage of such a model is that an arbitrary number of training images can be generated for aspect views of the object class. This approach is used to detect stamps in Chapter 6.

Usually in the literature, the objects are more complex, and the object database consists of true-class examples in various poses. It is usually considered expedient to remove all background features from these training images artificially. The training set can be improved by changing the zero background value of the training images to a constant value, corresponding to the true operational average background value [17, 20]. False-class training may also be used to provide a description of typical application clutter [18].

In an application using projection imaging (e.g. X-ray images), it is more difficult to separate an object from its background. The objects will rarely appear in the scene without superimposed background features. In such cases one can safely provide a general idea of the expected background as part of the normal true-class training, because the features of the object class will be repeated much more than any feature of the background. Care must be taken with this implicit clutter rejection training to not emphasize any characteristic of the background in the training set. This technique of a completely natural training set is employed to detect diamonds in Chapter 5.

3.7 Design parameter summary

3.7.1 Filter size

Filter size is a straightforward discrete parameter. The intuitive guideline suggests filter dimensions capable of enclosing the biggest object sample in the database. The optimum size is expected to be slightly bigger than the largest object.

3.7.2 Energy minimization weight (c_{em})

This parameter is a continuous variable defined as $0 \leq c_{em} \leq 1$, and controls the relative importance of the two different energies minimized by the MINACE filter's synthesis algorithm (discussed in Section 3.4.6). Its value should be adjusted in a logarithmic scale to have the desired effect. It is an important parameter, capable of forming a MACE filter (by minimizing only the system signal energy) or a MVSDF filter (by minimizing only the system noise energy). Even so, the region around $c_{em} = 10^{-4}$ is widely used in literature and may be expected to yield the best results.

3.7.3 Thresholds

The most significant threshold is the one used to determine the termination of the training process (see Section 3.5). Another discriminating threshold is used in the testing phase, similar to the training threshold. It is slightly lower due to the unfamiliarity of the filter with the testing set.

Literature on SDF filters have established standard values for these thresholds, e.g. 0.8 for the training threshold and 0.7 for the testing threshold.

3.7.4 Other considerations

There are two last points worth discussing:

Training set size and composition When a limited number of images are available, the true-class training and testing set sizes should be carefully chosen. There must be enough training images to create a decent filter, yet enough testing images to achieve representative results. Furthermore, a true-class training set should contain roughly equal numbers of object sizes.

Multiple filters The use of multiple filters can, in some cases, improve a system's performance. For example, different filters can be designed for different parts of the object size distribution. These filters will each have lower size variance tolerance requirements than a single filter in such a system. This example should be especially useful if the filter's learning capacity is saturated with object features. Object size

invariance and rotation invariance each takes a large toll on a filter's capacity to discriminate.

3.8 Conclusion

The synthetic discriminant function has evolved into a mature and very usable filter class. Although the idea behind these filters is straightforward, there are many design details to bear in mind. These details help provide the correlation filter with its flexibility.

Although the theory in this chapter is not recent work, the MINACE filter is mature enough so that problems of object detection can be investigated with confidence. This work is not meant to be an exhaustive study, but it needs to be mentioned that the MINACE filter as described here still has a number of issues that need to be addressed:

1. The composition of the minimization function can be improved. In the literature about the MINACE filter studied for this work, there is a variety of deviations from the general model as described here, but it is not clear which of those can be accepted as a significant and general improvement.
2. These SDF filters has the bothersome training characteristic that more training images included in the filter (better object modeling) leads to poorer minimization (energy increases), and therefore more false alarms. This is addressed in [21], where the constraint equation is changed so that the variance of origin correlation peaks is minimized instead. Although this looks viable, it does not improve significantly on the performance achievable with the MINACE filter.

Now that a general tool is available with which to detect objects, another tool is needed to interpret the output.

Chapter 4

Image Segmentation

The MINACE filter produces an intensity output where the height of the landscape (the grayscale intensity) indicates the level of significance. In this raw format the output is already useful for visual analysis, but it needs to be segmented into regions of interest (ROIs) so that postprocessing can be used to reduce the number of false alarms. The process of finding ROIs in the output of a correlation filter amounts to detecting peaks in the grayscale landscape. Once this segmentation process is complete and the ROIs are available, the postprocessing can use any number of knowledge-based classification techniques.

This chapter considers a number of amplitude thresholding techniques, as well as the watershed algorithm, for image segmentation.

4.1 Sample filter output

A sample output from a MINACE filter is used to demonstrate the segmentation techniques in this chapter. The image in Figure 4.1 is taken from the highly cluttered diamond detection application discussed in Chapter 5, and this correlation output has a grayscale histogram ranging from -2.73 to 2.01 (in the figure the histogram is normalized). The image has been smoothed by a 7×7 smoothing filter. This image is chosen specifically for the high degree of variation contained in its grayscale landscape.

4.2 Global thresholding

Amplitude thresholding (also called binarization) [22] is the standard approach for peak detection. The simplest algorithm is global thresholding, which uses the same threshold value for all pixels in the image. If u is the grayscale level and t is the threshold, the transformation is written as

$$f(u) = \begin{cases} 0, & u < t \\ 1, & t \leq u. \end{cases} \quad (4.1)$$

Various global threshold values are applied to the sample image of Figure 4.1, and the results are shown in Figure 4.2. For the standard threshold value of 0.7, the ROIs are indicated on the filter output as shown in Figure 4.3.

The main problem with global thresholding is that it does not take gradual background variation into account. The result is that nearby peaks are often lumped into one region. Of course, this can be avoided by using a higher threshold value, but this is rarely desired because the threshold value is usually chosen to meet specific probability of detection or false alarm rate criteria. The alternative is to use more advanced thresholding techniques.

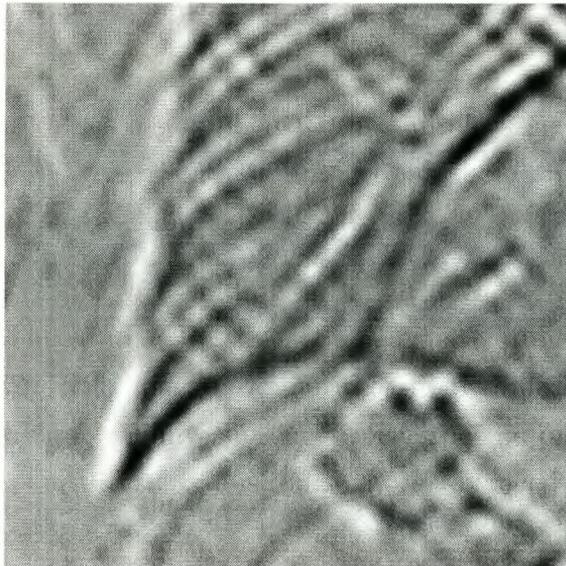


Figure 4.1 Sample MINACE filter output for comparing peak detection techniques (histogram not clipped, but normalized).

4.3 Adaptive thresholding

Adaptive (or local) thresholding [23–25] dynamically adjusts the threshold level depending on the characteristics of the local neighborhood. Often local thresholding is computationally simplified by dividing the image into a number of blocks and calculating the threshold adjustment for each block. The problem with this is the discontinuities at the block boundaries. For the moment, this is avoided by calculating the threshold for the

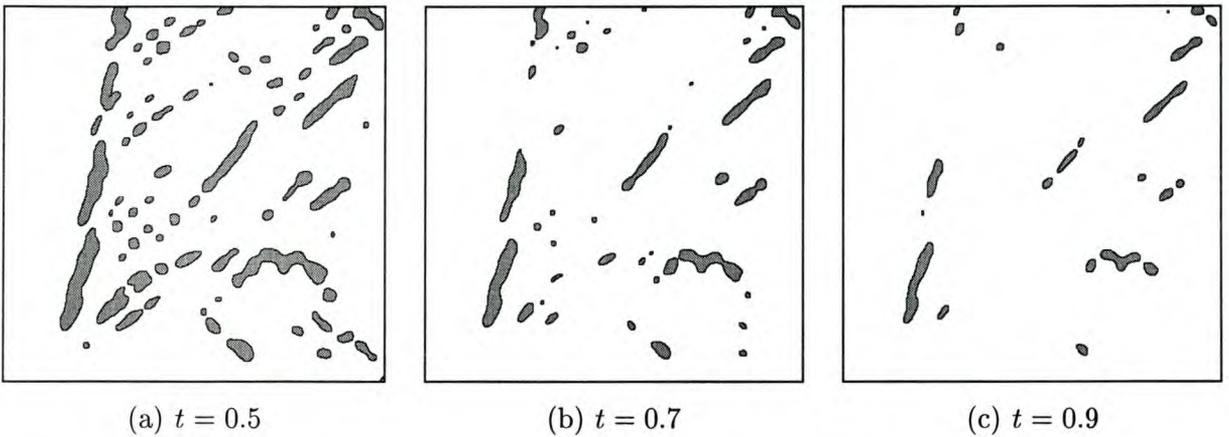


Figure 4.2 Example of global thresholding, for various threshold values.

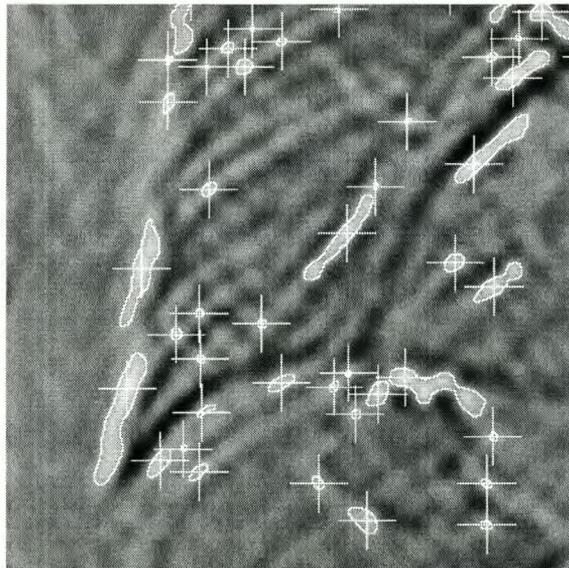


Figure 4.3 Superimposition of region borders (after global thresholding with $t = 0.7$), on filtered output, with region peaks indicated.

local region around every pixel.

The basic adaptive thresholding technique uses the mean grayscale value of the local region (the thresholding element, or window) to adjust the threshold. The transformation for any pixel intensity $x(i, j)$ is now written as

$$f(i, j) = \begin{cases} 0, & x(i, j) < t + m(i, j) \\ 1, & t + m(i, j) \leq x(i, j), \end{cases} \quad (4.2)$$

where the mean is the function $m(i, j) = \frac{1}{(2r+1)^2} \sum_{m=i-r}^{i+r} \sum_{n=j-r}^{j+r} x(m, n)$, and r is the radius of the thresholding element. Of course, the element need not be square, but for this situation it is a good choice. Note that $m(i, j)$ describes the convolution result between $x(i, j)$ and a square smoothing filter with sidelength $2r + 1$, and this can be calculated efficiently with FFT's.

For a thresholding element size of 35×35 (the same size as the MINACE filter used), a number of threshold values are investigated and the results shown in Figure 4.4. This technique appears to be less susceptible to region lumping than global thresholding, because some of the longer regions after global thresholding in Figure 4.3 are now split into separate ones. There are a couple of new regions as well, because the threshold modifier (the mean $m(i, j)$) can easily be negative for narrow peaks in negative backgrounds. The threshold value t is now a relative parameter, and the size of the thresholding element has a significant influence.

The effect of the element size is investigated in Figure 4.5, where the element radius is increased and decreased in steps of 5 pixels for the threshold value of 0.5. When the

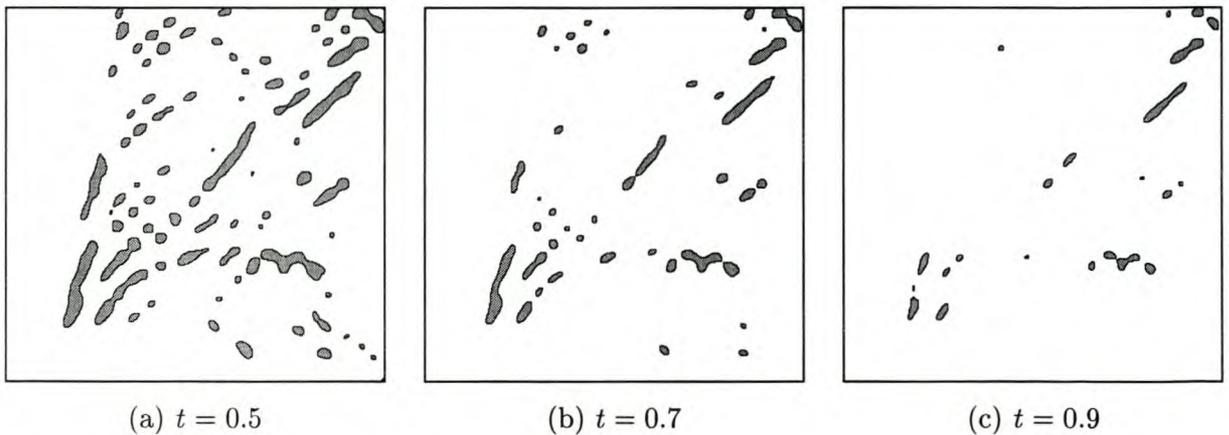


Figure 4.4 Example of local thresholding, for various threshold values, using the mean characteristic on a 35×35 thresholding element.

thresholding element is too small, it becomes too difficult for a peak to stand out from the average, and when it is too big, it is not really local anymore and does not benefit peak separation. If performance is judged by both the number of peaks and the rarity of region lumping, the case where the element size is 25×25 seems to be a good choice.

Adaptive thresholding can use any characteristic of the local neighborhood, not necessarily the mean. If, for instance, the median is used for $m(i, j)$ in (4.2), and the sample image is thresholded with an element size of 35×35 , the results would appear as in Figure 4.6. Again, several values for t are chosen. It looks as if the median characteristic is a slightly better choice than the mean, although it tends to be computationally expensive.

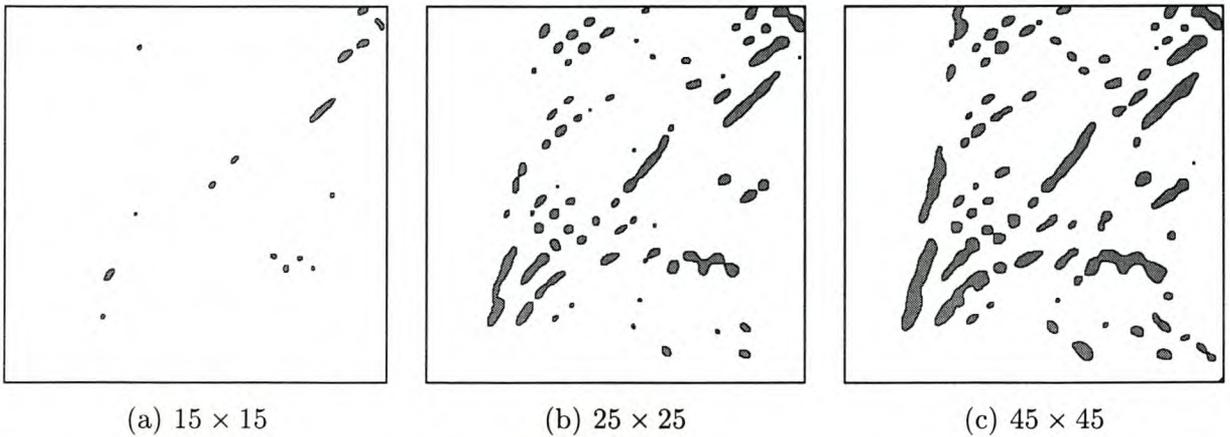


Figure 4.5 Investigating the effect of the element size on mean-based local thresholding, for a threshold value of $t = 0.5$.

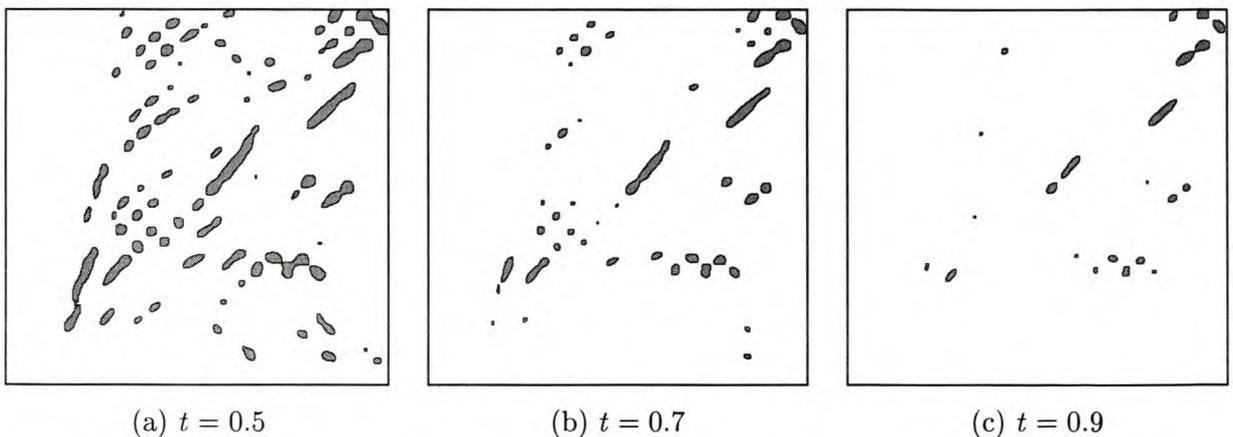


Figure 4.6 Example of local thresholding, for various threshold values, using the median characteristic on a 35×35 thresholding element.

The effect of the element size is investigated again, in Figure 4.7, where the element radius is increased and decreased once by 5 pixels. It appears as if the original 35×35 size is the better choice.

The last adaptive thresholding technique investigated is that of Niblack [26]. This technique uses, instead of the mean $m(i, j)$ in (4.2), a weighted linear combination of the mean and the standard deviation of the local neighborhood. If $s(i, j)$ is the standard deviation of the neighborhood around pixel (i, j) , the transformation is written as

$$f(i, j) = \begin{cases} 0, & x(i, j) < t + m(i, j) + c s(i, j) \\ 1, & t + m(i, j) + c s(i, j) \leq x(i, j), \end{cases} \quad (4.3)$$

where c is the weight of the standard deviation.

The effect of varying t when the Niblack technique is used with a thresholding element size of 25×25 and an initial choice for c of 0.5, is shown in Figure 4.8. Lower choices for the base threshold value t is needed to compensate for the addition of the standard deviation value to the dynamic threshold.

The effect of the weight c is investigated in Figure 4.9. Based on the results in Figure 4.8, a base threshold value of $t = 0.4$ is chosen. Figure 4.9 suggests that $c = 0.3$ is a good trade-off choice, and the results seem good. Although a direct implementation of Niblack thresholding is slow, it can be implemented efficiently.

The output of zero-mean MINACE filters does not have gradual background variation (only regions of correlation emphasis), therefore adaptive filtering only helps prevent the

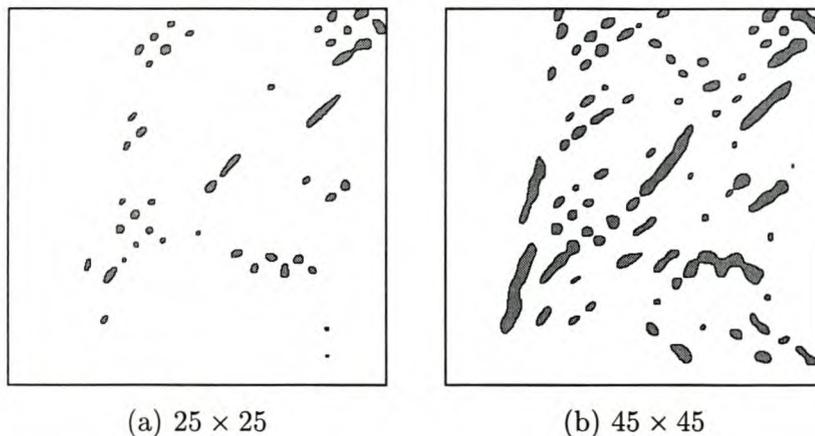


Figure 4.7 Investigating the effect of the element size on median-based local thresholding, for a threshold value of $t = 0.5$.

filter peaks to cluster. The downside introduced by adaptive filtering is an uncertainty of the absolute peak values that can pass the dynamic threshold. Another approach, based on the watershed algorithm, is investigated.

4.4 Watershed-based thresholding

The calculation of watersheds is a morphological operation for which the algorithms has evolved considerably since originally introduced by Digabel and Lantuéjoul [27]. When used for segmentation, the watershed transformation is optimal in the sense that every

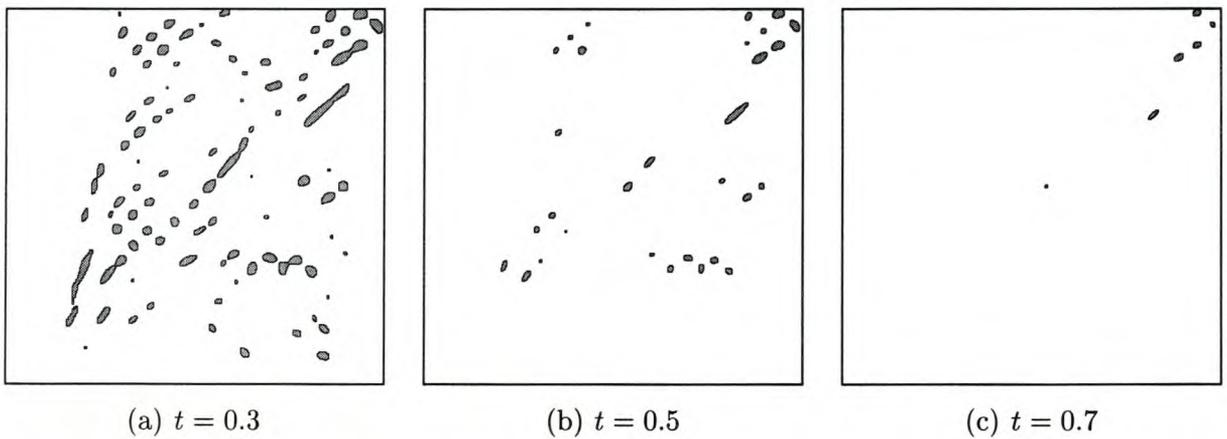


Figure 4.8 Example of local thresholding, for various threshold values, using the Niblack technique on a 25×25 thresholding element, with the standard deviation weight $c = 0.5$.

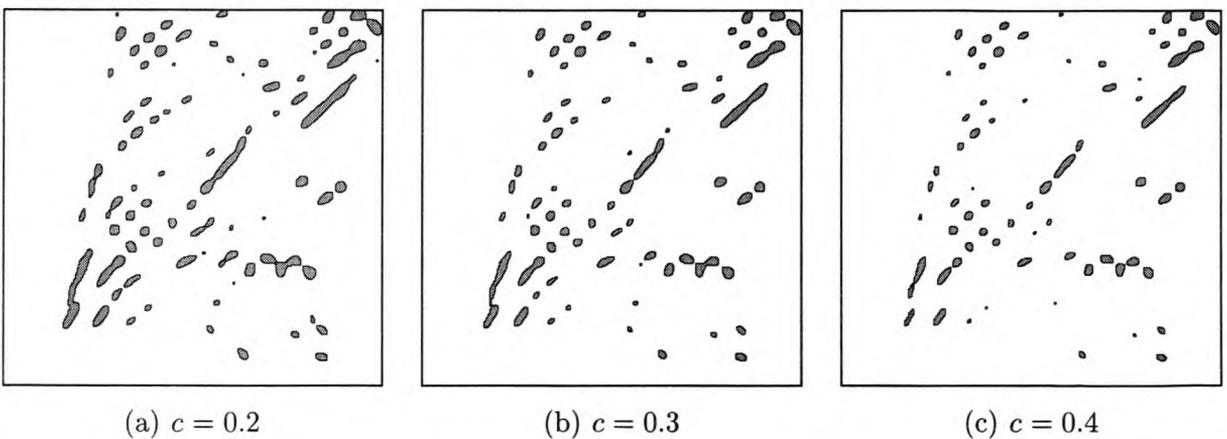


Figure 4.9 Investigating the effect of the standard deviation weight c on Niblack binarization, for a threshold value of $t = 0.4$ and a 25×25 thresholding element.

single peak in a grayscale landscape will be identified uniquely. The algorithm employed here is the one by Vincent and Soille [28]. In this watershed algorithm, there are still some issues with obtaining a true tessellation of watershed lines in an image. Since this application of peak detection is interested mainly in the peaks of the regions (not in the accuracy of the boundaries between the regions), these issues are irrelevant.

4.4.1 Approaching peak detection

The concept of a watershed line comes from the field of topography. If a line exists in a topographic relief map such that a drop of water placed on either side of the line will always flow away from it, this line is defined as a watershed. The drop of water will continue flowing until it reaches the minimum of the catchment basin in which it was dropped.

For the watershed algorithm to be applied to the MINACE filter output, the image intensities must be inverted so that the emphasized objects become valleys or hollows (instead of hills or peaks). They will then form the center of catchment basins in the watershed landscape. The algorithm identifies every catchment basin uniquely as a region, without any inherent selection process. In other words, any thresholding action takes place after segmentation and can be done directly on the peak (pixel of highest intensity before the inversion) of each region. This eliminates the possibility of region clustering and also the threshold uncertainty found with adaptive thresholding. Another important implication is that region shapes no longer depend on the threshold value.

4.4.2 Algorithm

The algorithm presented in [28] is based on the simulation of a grayscale landscape's gradual immersion. This is by far not the only algorithm for calculating watersheds, but it is fast and accurate. Here follows a brief summary of the algorithm:

1. Initialize Variables

- output matrix = "init" value
- distance matrix (for each element of the input matrix) = zero

2. **Intensity Sort:** all pixels are sorted in increasing order of grayscale intensity (with the aid of a reference matrix)
3. **Main Loop:** for each intensity value from low to high
 - (a) for all pixels of current intensity
 - mark pixel
 - if pixel adjacent¹ to watershed- or region pixel, put into fifo queue
 - (b) repeat until fifo queue of pixels empty
 - assign “watershed” or “region” status according to its neighbours¹
 - put all unassigned (yet marked in (a)) neighbours into the queue
 - (while keeping track of the pixel “distance” travelled since the original group of pixels in the queue, with the aid of a “distance” matrix)
 - (c) for all pixels of current intensity
 - reset distance of pixel to zero
 - if unassigned:
 - assign new region value to pixel (discovered new minimum)
 - iteratively assign all unassigned (yet marked in (a)) neighbour pixels to this region value, in the output matrix

4.4.3 Implementation

The result when the watershed algorithm is applied to the sample MINACE filter output of Figure 4.1, is shown in Figure 4.10. The only adjustment of the raw output in this figure is to start the region label number far enough from zero (the label of the watershed pixels), thus improving contrast. The superimposed image is shown here on the inverted version of the sample image Figure 4.1. The original will be used again from here on, so as not to confuse the idea of region “peaks”.

At this stage, every peak in the MINACE filter output is a unique ROI. A number of postprocessing steps is used to complete the thresholding. Firstly, any watershed pixel that touches two or more distinct ROIs is assigned to the adjacent region with the highest

¹Adjacency in this implementation of the watershed algorithm is chosen to be modeled with 8-way connectivity. This means that diagonal traversing is allowed as well as horizontal/vertical.

peak. The visible effect of this step is minor, but it prevents the possibility that two adjacent regions will not be recognised as such, when this becomes significant later. The next step is to locate the peak within every ROI, and use it to threshold its ROI. For a threshold value of 0.7, the result is shown in Figure 4.11.

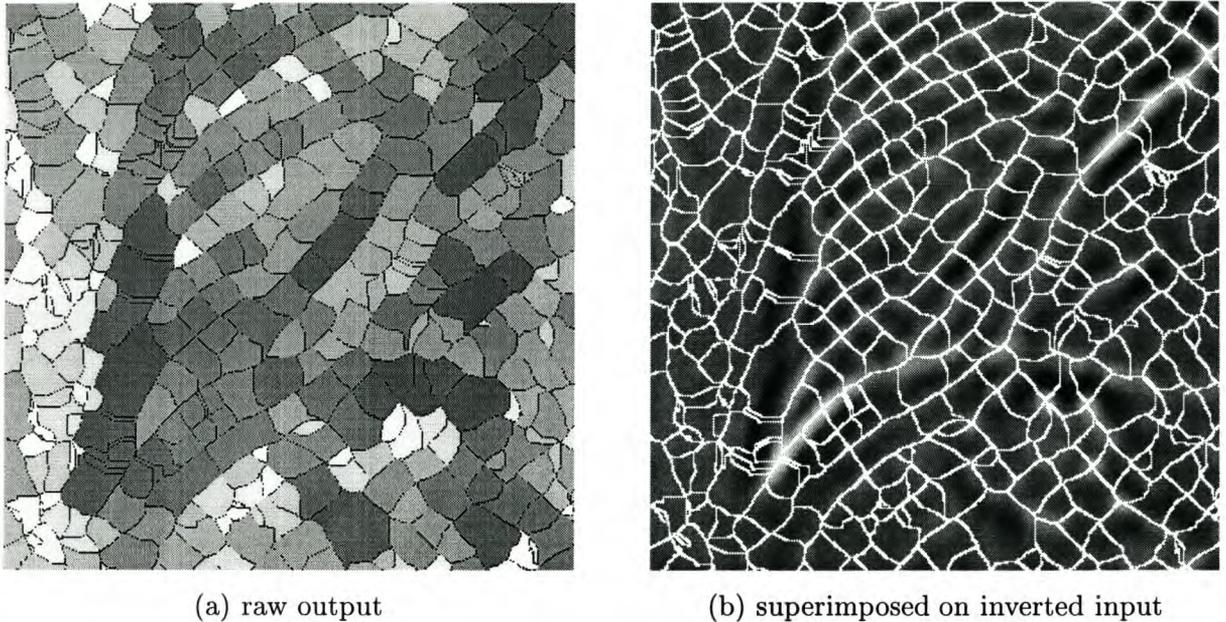


Figure 4.10 Demonstration of watershed algorithm on sample MINACE filter output.

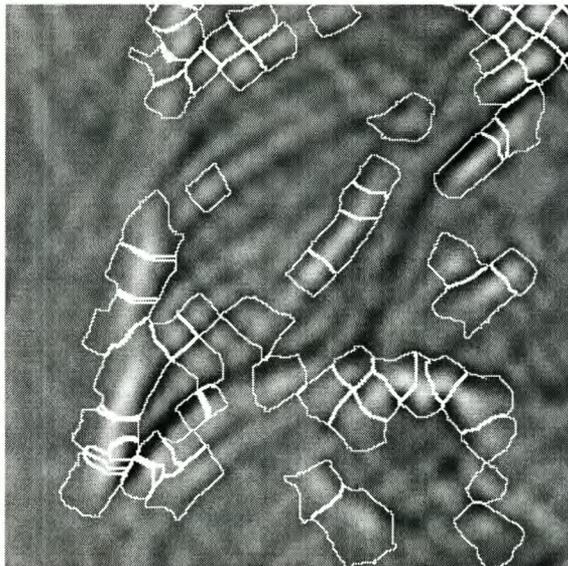


Figure 4.11 Thresholding the regions after the watershed algorithm by their peak values, using a threshold value t of 0.7.

A per-region postprocessing step is used next to improve the accuracy with which each ROI describes its peak. The grayscale histogram of the pixels inside each ROI is cropped to the top 0.5 intensity range of that ROI's histogram. For example, if a ROI initially includes pixel intensities from -0.3 to 1.1 , it keeps only those pixels with intensities greater than 0.6 . The result of this step is shown in Figure 4.12. This is a significant improvement on readability and usability.

The results using different threshold values are shown in Figure 4.13. As a result of the perfect discrimination, the adjacency of peaks is clearly visible. The same threshold values are used here (and in Figure 4.12) as for the global thresholding results in Figure 4.2. These two sets of results, when compared directly, clearly shows both the improved separation of peaks and the improved region descriptions.

The execution time of watershed-based thresholding is faster than that of the mean-based adaptive thresholding technique in Section 4.3: 1.55 s vs. 1.95 s on this 600×600 image (refer to Appendix A for implementation detail). While the raw global thresholding routine has negligible processing cost and the raw mean-based adaptive thresholding routine requires 1.5 s for the smoothing filter, they both need an additional 0.5 s to identify the contiguous regions of thresholded pixels. This step is automatically included in the watershed algorithm.

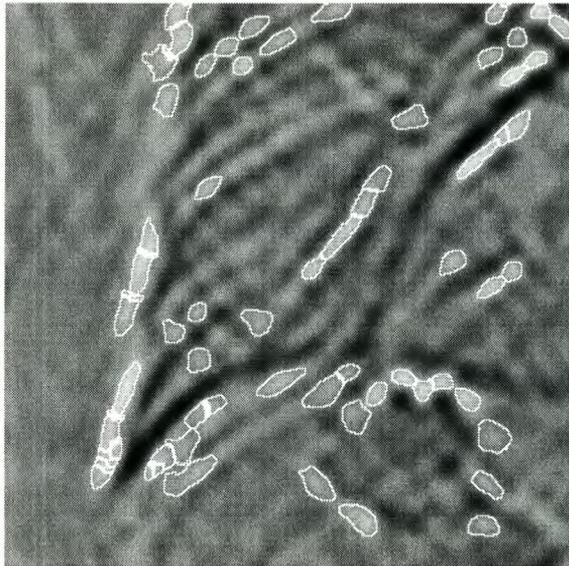


Figure 4.12 Determine ROI shapes after watershed-based thresholding: clip each ROI's grayscale histogram to top 0.5 of intensity range ($t = 0.7$).

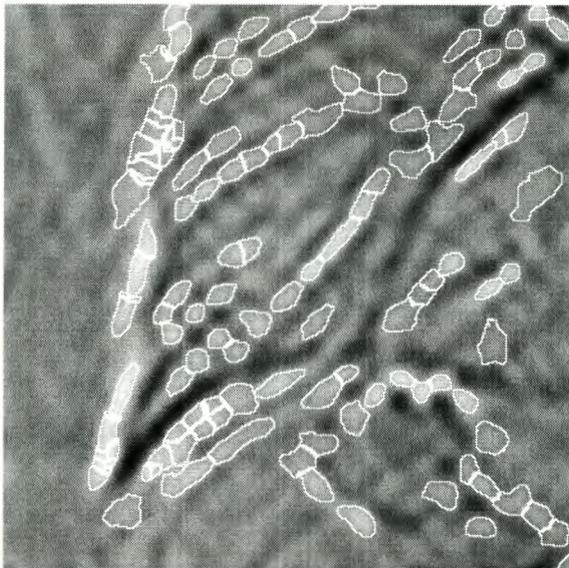
4.5 Conclusion

This chapter shows that peak detection as a step is not difficult to do well, and quite possible to do perfectly. The conceptual advantages of the watershed-based thresholding technique, as compared to the standard thresholding techniques, are the following:

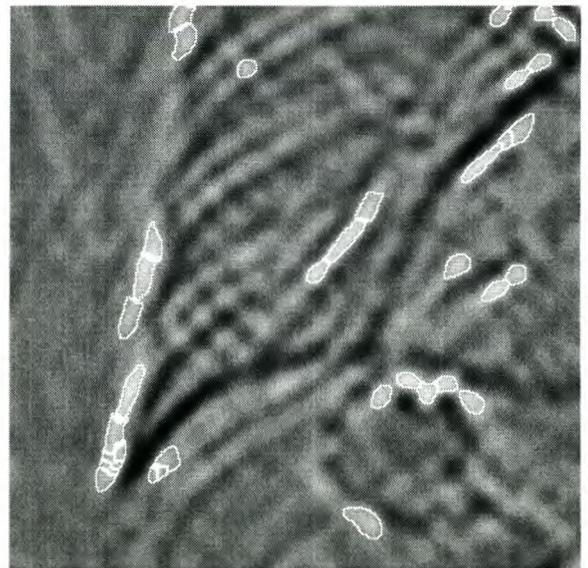
1. The shape of the ROI is threshold independent (as it is determined before the thresholding step in watershed-based thresholding). This can easily be exploited to yield a good representation of the underlying peak, consistently.
2. The separation of peaks is always perfect, for any threshold value (consider the range of very low threshold values for which the standard thresholding algorithms will yield exactly one region of the same size as the image).

While global thresholding has the advantage of being faster than watershed-based thresholding, there are no advantages to using adaptive thresholding. In fact, adaptive thresholding's detection of relative peaks introduces an uncertainty which may not be desired when processing correlation peaks.

These segmentation techniques will be compared further in Chapter 5, where truth data is used to generate statistics.



(a) $t = 0.5$



(b) $t = 0.9$

Figure 4.13 Investigating other threshold values when using watershed-based thresholding.

Chapter 5

Case Study: X-Ray Diamond Detection

This is a case study on the location of diamonds, hidden on a person, using an X-ray scanning device. The objective is to develop a non-intrusive, fast and accurate alternative to body searches. All the hardware for the system is already in place. At the time when the project started, the X-ray photographs were manually interpreted by an expert using enhancement filters.¹

The project is based on a collection of X-ray photographs on CD. The first part of this chapter examines these source images to illustrate the scope of the problem, and to develop a general preprocessing technique. Next is the detection stage, which consists of a MINACE object detection filter and a segmentation algorithm. A number of postprocessing techniques makes up the verification stage. The design choices are thoroughly investigated throughout, using detection rate statistics. The source images are very big, and therefore the design process often uses a more manageable sample cutout image for illustration.

The theoretical properties of the detection techniques suggest a good detection capability, but, in this highly cluttered environment, it is unlikely to achieve a fully automated detector.

¹The detail of these filters was withheld from the author to prevent bias in design.

5.1 Source images

5.1.1 Overview

There are 24 X-ray photographs on the source CD. These images are from three data generation sessions. Each session placed an increasing number of sample diamonds on the subject person, and created X-ray photographs with and without the diamonds. All the images are raw output from the Scannex machine, in 16 bit grayscale, and as Visualisation/Image File Format (VIFF²) files. The images are consistently 2400 pixels wide, and approximately 5215 pixels high. In complex, single precision, floating point format, used to calculate FFT's, it amounts to 95.5MB. A typical raw output image from the scanning machine (normalised, for visual clarity) is shown in Figure 5.1, complete with edge effects from the roof, the floor and even a broken scanning channel. These photographs possess remarkable resolution (which cannot be fully appreciated on paper) — reading the time on the worker's analog wristwatch is just short of possible. Note that the background has the highest intensity, because intensity is a measure of how many X-ray particles can pass through space.

²used by Khoros software, Khoral Inc.

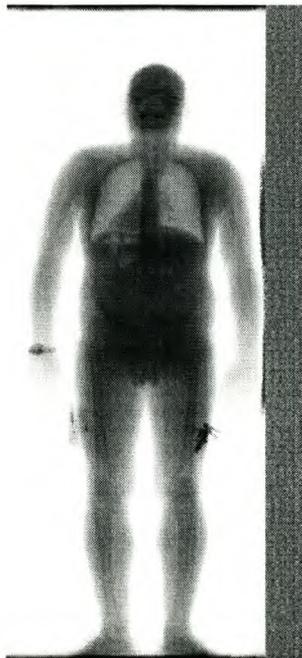


Figure 5.1 Typical raw output of X-ray personnel scanner.

The X-ray photographs are created by several X-ray sensitive modules aligned in a row, scanning in a top-down fashion much like a standard scanner. The result is several vertical X-ray strips (channels) which combine to form a full photograph. The scanning modules' individually imperfect X-ray-to-digital signal transformation curves lead to a marked discontinuity at every channel boundary. These vertical artifacts are visible in the close-up shown in Figure 5.2.

Two methods for detecting the channel boundaries are tested. The first detection algorithm subtracts every column element-wise from the previous, and calculates the mean of each resulting column. The discontinuities are quite visible when the one-dimensional result is plotted. The second algorithm calculates the mean of each column, performs 4th order polynomial linear prediction on the means, and keeps the error value as a result. Both techniques are demonstrated in Figure 5.3. Note the extra vertical glitch in the plots (the non-regular impulse near column 1500) and the noise on the right from the output of the broken channels. The predictive algorithm is quite sufficient for this task.

During the generation of the source photographs, the scanning machine had either 11 or



Figure 5.2 Close-up with several clear vertical scanning side-effects.

12 working channels. Each channel yields a width of between 170 and 190 pixels to the image. The variety of scanning side-effects found in these photographs are displayed in Figure 5.4. It is not known whether these effects can be remedied in an automatic fashion, since they are difficult to characterize. At least it is clear in Figure 5.3 that the channel boundaries can be detected automatically with relative ease.

Every image is stored in standard 16 bit grayscale format, although not all of the bits are used. The histogram of the image in Figure 5.1 (ignoring the edge effects) is shown in Figure 5.5. The large peak of high intensity found on the right side is due to the relatively large portion of the image that is background.

The VIFF file format is inconvenient, because it is rarely supported by image manipulation/viewing utilities. Using the VIFF information in the KHOROS programmer's

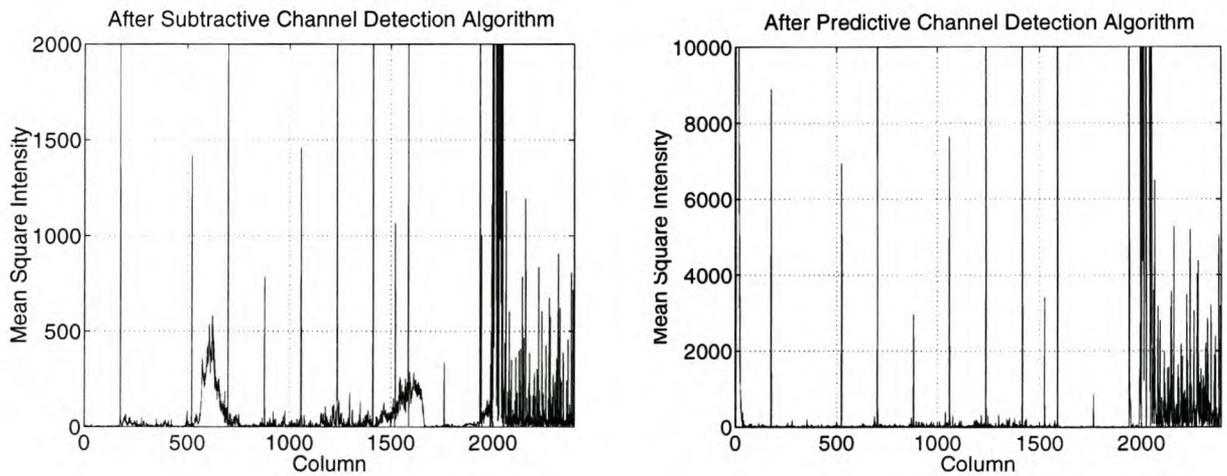


Figure 5.3 Column-wise outputs of the two channel boundary detection algorithms tested.

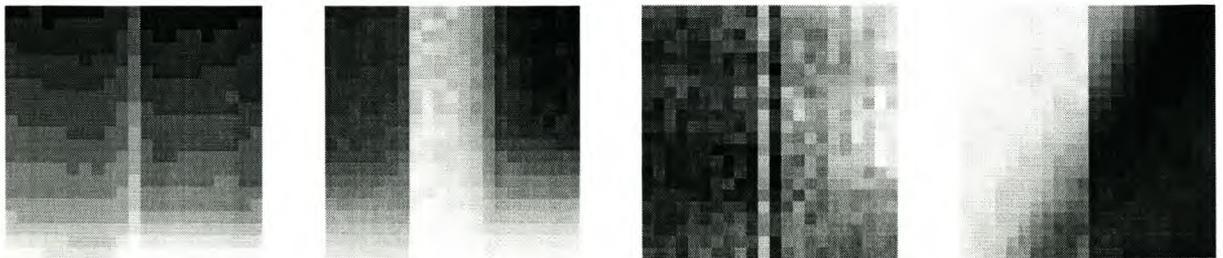


Figure 5.4 Typical side-effects of the X-ray scanning mechanism.

manual [29], the images are converted to the PNG³ format. PNG is chosen as it is a modern, non-proprietary format with good lossless compression.

Although the edge effects found on the scanned images can easily be ignored, the output from the MINACE filter in these areas is unpredictable. Since the problem can be serious, a simple routine is added which negates these edge effects at the top, left and right of an image. This routine searches through a few hundred rows or columns next to the top, left and right edges to find the smallest mean row or column. When the new edges are known the image can either be cropped, or the image intensity can be set to zero all the way to the edge. Cropping the image is useful to reduce the execution time of the filter.

Experimentation shows that the correlation filter performs best if the image is inverted, as this puts more emphasis on the regions of interest. This phenomenon is explained by interpreting grayscale values as levels of significance. Therefore, the background (which is easily detectable in the original images) should have the lowest grayscale value.

The same image from Figure 5.1 is shown again, in Figure 5.6. On the left is the un-normalized original, and on the right is the histogram adjusted and edge effect removed version. The right-hand image is what is used in the subsequent image processing stages.

³PNG (Portable Network Graphics) is the Open Source successor to GIF (CompuServe's Graphics Interchange Format)

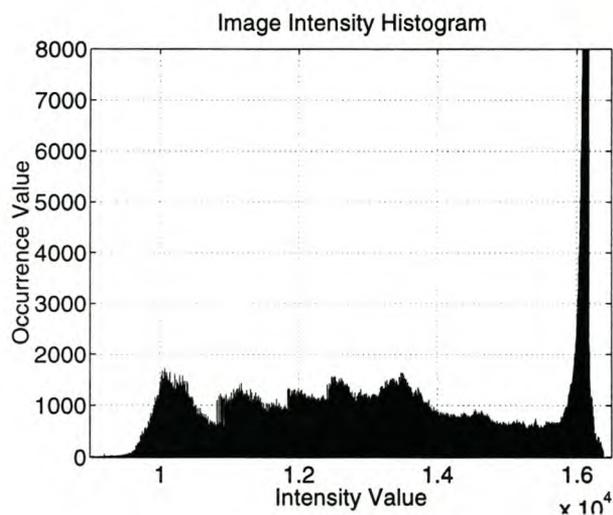


Figure 5.5 Intensity histogram of a typical X-ray photograph of a worker.

5.1.2 Diamond database

Among the 24 source images provided, 10 images contain diamonds placed on the scanned person, 9 images contain just a person and the remaining 5 contain either empty scans or just an arrangement of diamonds. In the second and third data generation sessions, the diamonds were placed on a regular grid. The organized structure of the placement has the additional benefit of easing the diamond location process, aiding manual data transcription.

The transcription of the diamond co-ordinates is done as accurately as possible, in accordance with Section 3.5. Only the diamonds occurring in their natural environment (i.e. on a person) are added to the diamond database (refer to Section 3.6.5). Descriptions of the three data generation sessions follow:

October 1998: Three images with diamonds are available, containing four diamonds in

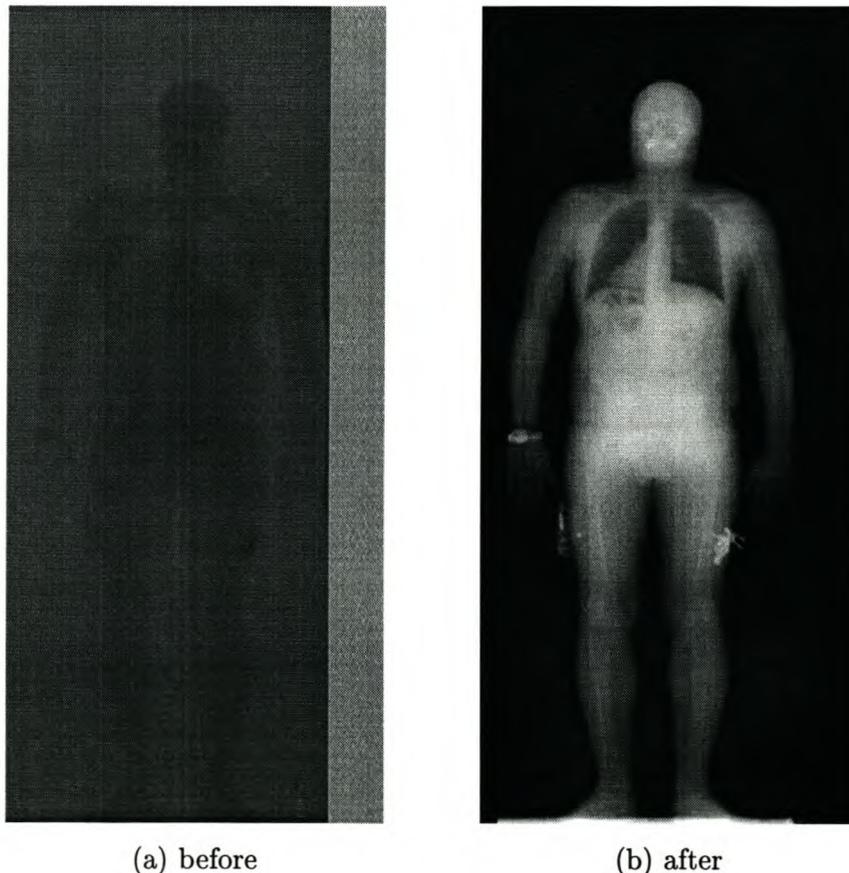


Figure 5.6 Illustrating the preprocessing adjustments on a raw source image.

each. It is only known that the diamonds are located in the torso region. The two smallest diamonds could not be located manually or otherwise in any of these images, even with the aid of several standard image enhancement tools (such as edge detection and histogram equalization).

February 1999: This was the first session to use a regular grid of diamonds. This session's grid contains 10 diamonds, ordered from large to small, from top to bottom. An example of the grid, from one of the four available images, is shown in Figure 5.7. Square boxes with side-length 51 pixels are drawn around each diamond.

The transcription of this session's data is not completely successful. In one image, the 3 smallest diamonds could not be found. In another, none was detectable manually or otherwise — it is known that in this case the diamonds were placed on the side of the subject's head.

July 1999: The grid in this session contains 32 diamonds. Three images are available,

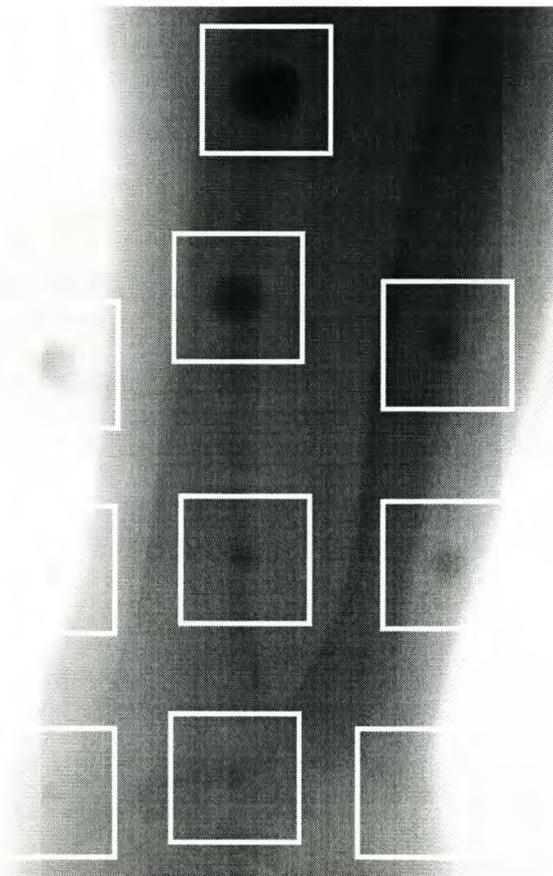


Figure 5.7 Layout of the mask of diamonds in February 1999 data generation session.

with the grid on various positions over the torso. All the diamonds are successfully located and included in the database. The layout of this grid is shown in Figure 5.8.

The collection of truth data totals 129 diamonds from a variety of sizes. Since this is a small amount of truth data for such a study, the database is enlarged by rotating the existing images. This also assists in rendering the images rotationally invariant.

Table 5.1 summarizes the various data sets. Training and testing sets are identified, and three size distinctions are made for experimental purposes. The size divisions are based on rough radius measurements done during the transcription process. These data sets are compiled as lists of co-ordinates, and physically realized into image cutouts using an extraction routine.

Figures 5.9 – 5.11 display the contents of the data sets. The size of these image cutouts is 61×61 . Each cutout is separately normalized for visual aid, therefore appearing to possess much greater contrast. The original contrast level of a cutout can be estimated roughly by its visible signal-to-noise ratio.

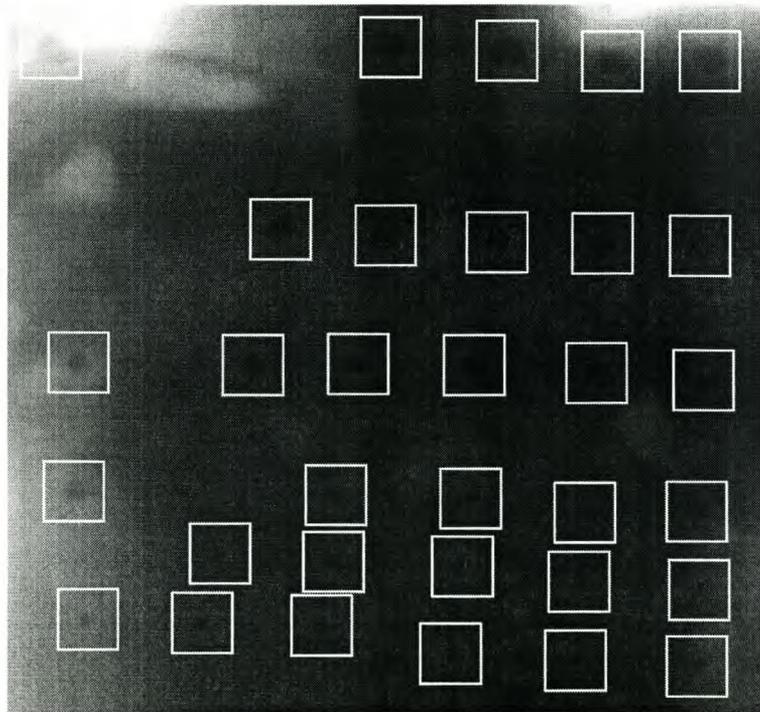


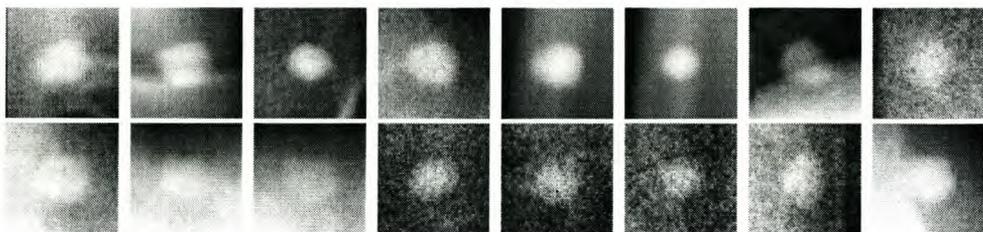
Figure 5.8 Layout of the mask of diamonds in July 1999 data generation session.

Table 5.1 Distribution of available unique diamonds between various true-class data sets.

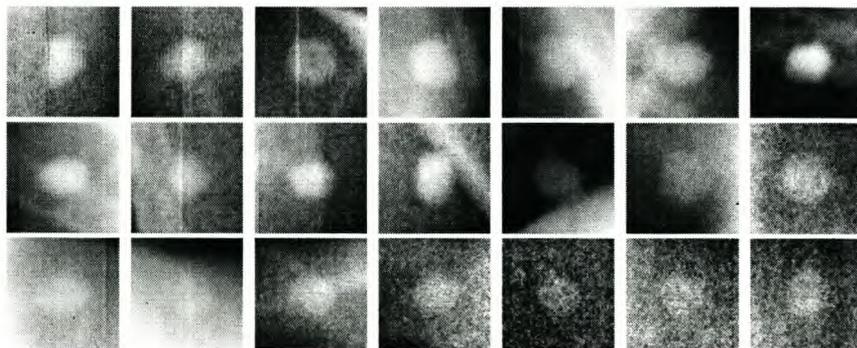
data set	approx. radii (pixels)	training	testing	total
large	≥ 11	16	21	37
medium	7 – 10	21	45	66
small	≤ 6	12	14	26
all		49	80	129

5.2 MINACE detection filter

The practical behaviour of the filter is explored by investigating the effect of the design choices. The investigation begins by recording the following filter observations for each change in the main filter parameters: the detection rate (as a percentage of the true-class testing set), false alarm rate (as a percentage of the false-class testing set), eventual size of the true-class training set, and condition number of a filter. After this statistical parameter investigation, a representative sample is taken from a source image and used to test the implementation of the MINACE filter.



(a) training set



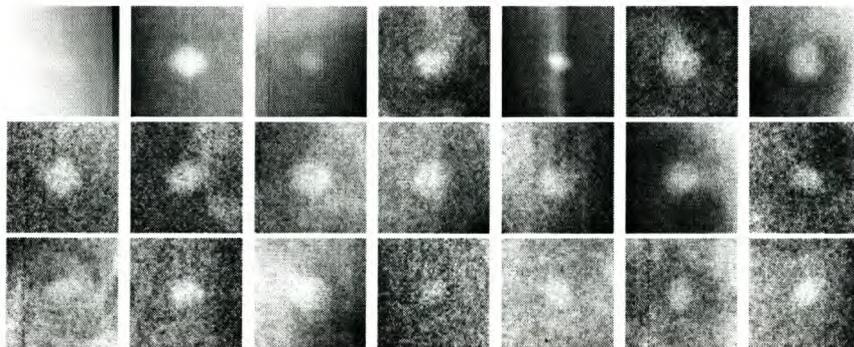
(b) testing set

Figure 5.9 Data set distribution of “large” diamonds (cutouts individually normalized).

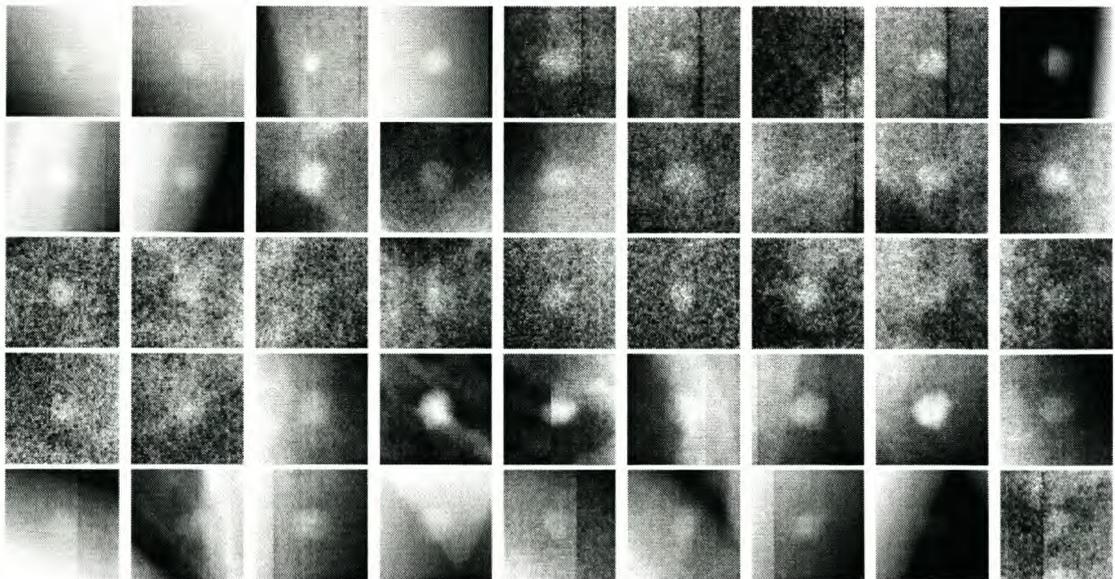
5.2.1 Investigating filter design choices

Some of the design choices controlling the MINACE filter (Section 3.6 and Section 3.7) must be tuned for every unique application. The straightforward choices are made first, and the filter is chosen to be a zero-mean filter, using the simplified model for the noise energy. False-class training is disregarded because the hard clutter in this application is so varied. The standard constraint and threshold values are used to generate and test the filter. The influence of the following filter choices are investigated in this section:

Filter radius: The simulated filter response is sampled for each of the two-dimensional



(a) training set



(b) testing set

Figure 5.10 Data set distribution of “medium” diamonds (cutouts individually normalized).

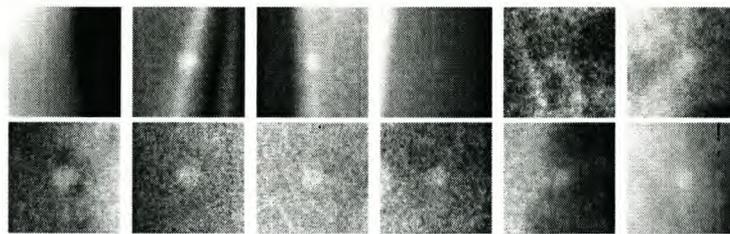
filters having radii in pixels between 1 and 30 (with the largest diamond having an estimated radius of 15, and the smallest, 3).

Energy minimization weight: A list of discrete sample values for the energy minimization weight is created. These values are for the chosen c_{em} -plane represented by (3.33) and (3.32). The selection follows an approximated logarithmic descent from 1, going smaller for 7 orders of magnitude before reaching zero. The list, containing 23 values, is as follows:

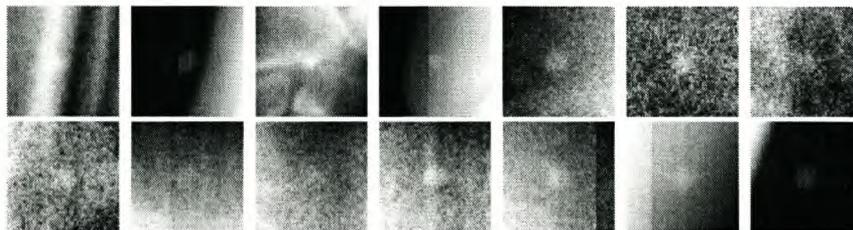
$\{1.0, 0.5, 0.2, 0.1, 0.05, \dots, 0.0000001, 0.0\}$.

Object size distribution: The experiment records data for the “all”, “large”, “medium”, and “small” data sets — as indicated in Table 5.1.

Statistics on the false alarm rate of a filter are generated using the false-class testing set. This data set is quite large, and is generated by randomly creating 200 cutouts from each of the 9 source images that contains a person and no diamonds. To prevent the background from influencing this testing set, no cutout is accepted into the set if it has an average intensity below 1% of the maximum source image intensity. The false alarm rate is therefore the percentage of these 1800 cutouts’ origin correlation peak values that exceed the testing threshold of 0.7.



(a) training set



(b) testing set

Figure 5.11 Data set distribution of “small” diamonds (cutouts individually normalized).

A large amount of raw data is generated, since filters are trained for each of the 23 chosen c_{em} -values, for each of the 30 chosen filter radii, and for each of the 4 object size distribution sets. The c_{em} -vs.-filter-radius matrix, in Table 5.2, contains the statistical detection rate percentages of the trained filters for the “all” data set. Using a graphical representation, such matrices will be shown more concisely from now. Figure 5.12 illustrates this, using the same filters as in Table 5.2, but displaying both the detection rate and false alarm rate matrices.

Recall the significance of the energy minimization weight c_{em} discussed in Section 3.4.2, and the implication that the leftmost columns of the matrices in Figure 5.12 represent data from an MVSDF filter (actually, with this simplified noise model, an ECP filter), and the rightmost, from a MACE filter. In between are samples of minimization weight choices for the MINACE filter. In accordance to the theory, note that the MACE filter consistently has the worst detection rate percentage, and the best false alarm rate percentage. Some performance trends in the filter radii are observed, but no significant deductions can be made.

Although both detection rate ($R_d\%$) and false alarm rate ($R_f\%$) are important observations, they are interpreted as the two components of a composite observation: “performance” (P). Performance is defined to possess a linear relation to both its components, with the value of 100 signifying the ideal result (100% detection rate, 0% false alarm rate).

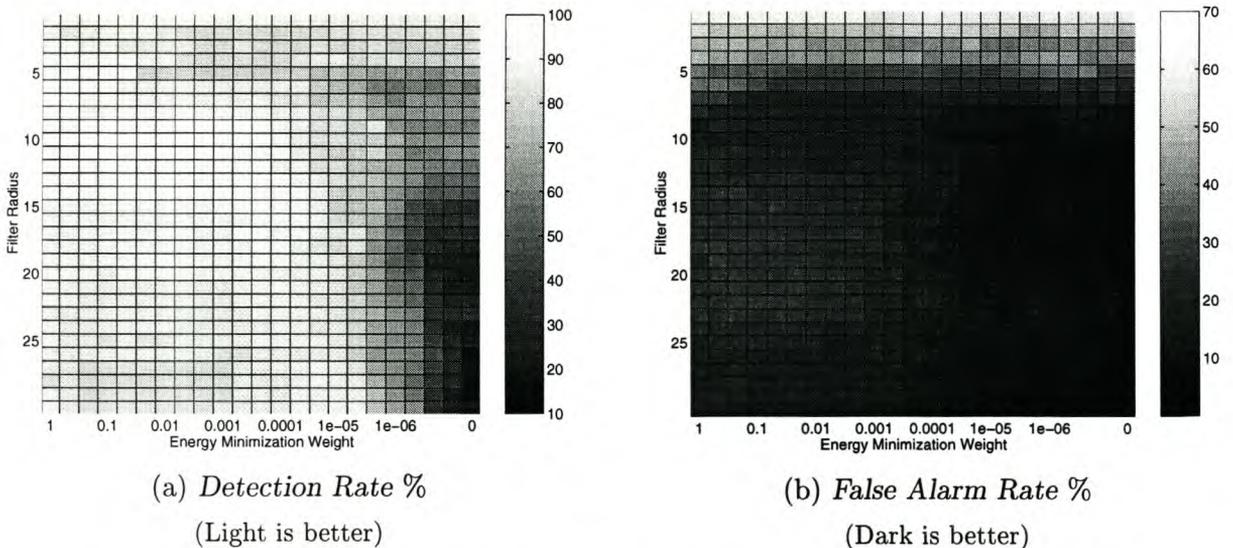


Figure 5.12 Components of *Performance* for “all” data set, as influenced by filter parameters.

Table 5.2 Raw matrix showing *Detection Rate %* for “all” data set, as influenced by filter parameters.

Filter Radius	Energy Minimization Weight c_{em}																							
	1.0	5× 10 ⁻¹	2× 10 ⁻¹	10 ⁻¹	5× 10 ⁻²	2× 10 ⁻²	10 ⁻²	5× 10 ⁻³	2× 10 ⁻³	10 ⁻³	5× 10 ⁻⁴	2× 10 ⁻⁴	10 ⁻⁴	5× 10 ⁻⁵	2× 10 ⁻⁵	10 ⁻⁵	5× 10 ⁻⁶	2× 10 ⁻⁶	10 ⁻⁶	5× 10 ⁻⁷	2× 10 ⁻⁷	10 ⁻⁷	0.0	
1	90.31	90.31	90.31	90.31	89.06	89.06	89.06	89.06	89.06	87.18	87.18	87.18	87.18	87.18	87.18	87.18	87.18	87.18	88.12	88.12	88.12	88.12	88.12	
2	87.18	84.68	85.62	86.56	86.25	86.25	84.37	81.25	81.56	80.93	81.56	81.56	80.00	80.00	84.06	84.06	84.68	82.50	82.50	85.31	85.31	85.31	85.31	
3	91.56	91.25	88.43	87.18	84.37	84.37	84.37	84.37	80.93	81.87	82.18	81.56	81.56	81.56	83.12	81.87	81.25	82.81	80.31	80.93	80.93	80.93	78.75	
4	94.68	95.00	91.87	90.31	90.31	87.81	87.50	81.25	79.68	80.00	81.87	79.68	79.06	83.12	86.56	86.25	78.43	82.81	82.81	80.62	81.25	80.62	81.56	
5	95.31	95.00	94.68	93.75	93.75	77.50	77.18	76.87	76.87	77.18	77.18	77.18	77.50	76.87	71.25	72.18	67.81	66.25	65.31	61.56	60.31	65.00	65.00	
6	97.50	96.87	95.62	94.06	89.06	89.06	89.06	88.75	88.75	88.75	88.75	88.75	87.18	85.93	79.68	74.68	70.31	59.68	55.31	58.12	56.25	57.81	56.56	
7	97.50	97.50	90.62	89.37	89.37	89.06	89.06	88.75	88.75	88.75	88.12	86.87	84.68	82.50	78.12	76.56	73.43	66.87	58.75	64.06	61.56	61.87	62.50	
8	94.06	93.43	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	90.62	88.43	86.87	83.43	83.75	79.37	69.37	64.06	63.12	61.56	63.12	62.18	
9	95.31	95.93	95.31	95.31	95.31	95.31	95.31	95.31	95.31	95.31	94.37	93.12	89.68	85.62	83.75	83.12	79.37	85.62	66.87	57.81	57.50	57.81	58.43	
10	97.18	96.56	97.18	96.87	96.87	96.87	96.87	96.87	96.87	95.62	95.31	94.06	90.93	86.87	82.81	82.18	79.68	78.43	74.37	63.75	61.87	61.87	62.50	
11	96.25	96.56	96.56	96.56	96.56	96.56	96.56	96.56	96.56	96.56	96.25	94.37	93.12	87.50	82.81	80.93	79.06	83.12	68.75	60.31	60.00	61.25	61.25	
12	96.87	96.56	95.93	95.93	95.93	95.93	95.93	95.93	95.93	95.93	96.25	94.37	94.06	92.50	92.50	83.12	82.50	77.50	71.56	68.75	61.25	58.12	57.18	55.93
13	96.25	95.62	95.31	95.31	95.31	95.31	95.31	95.31	95.31	95.31	95.93	95.31	94.37	94.06	93.12	92.50	90.00	82.50	78.75	75.00	68.43	62.81	50.62	50.62
14	93.75	94.06	93.75	93.75	93.75	93.75	93.75	93.75	93.75	94.06	94.37	94.06	93.43	95.00	95.00	90.00	79.68	75.93	74.37	65.62	56.25	46.87	47.81	48.12
15	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.81	93.43	93.12	95.00	93.75	91.25	79.37	77.18	75.62	65.00	44.06	37.50	35.00	34.37
16	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.18	92.50	92.50	95.00	95.00	90.62	85.31	79.37	74.68	59.68	45.93	33.75	34.37	36.25	36.25
17	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	92.50	91.87	91.56	93.43	96.25	96.25	95.00	92.50	87.50	87.81	67.81	57.81	42.50	30.93	28.75	28.75
18	91.25	91.25	91.25	91.25	91.25	91.25	91.25	91.25	91.25	90.93	91.25	92.50	96.25	94.68	91.87	89.06	84.68	88.75	67.50	53.43	41.56	29.37	26.56	26.25
19	90.93	90.93	90.93	90.93	90.62	90.62	90.62	90.62	90.00	90.62	90.93	95.93	94.06	91.87	89.37	86.25	81.56	65.93	51.87	44.06	23.75	22.18	22.50	22.50
20	90.31	90.00	89.37	89.06	89.37	89.37	89.37	89.37	89.37	89.37	89.37	90.62	91.87	92.50	91.87	90.00	87.81	80.93	69.06	55.00	47.18	25.00	21.25	23.43
21	90.62	90.00	89.37	89.06	89.06	88.75	89.06	89.06	89.06	89.06	89.06	90.31	91.25	93.12	92.18	90.62	87.18	78.75	68.75	59.06	51.56	30.62	23.75	21.87
22	90.00	89.37	88.43	88.43	88.43	88.43	88.43	88.43	88.43	89.06	89.06	90.31	92.18	92.50	92.18	88.75	86.56	82.50	68.43	63.12	52.50	32.18	25.31	27.18
23	89.06	89.06	88.75	88.75	88.75	88.75	88.75	88.75	88.75	88.43	88.43	89.06	91.56	92.18	92.18	91.56	86.87	81.25	72.81	65.31	57.50	35.00	31.56	30.00
24	89.37	89.06	89.06	88.75	89.06	89.06	89.06	87.81	87.18	88.12	89.06	88.75	91.56	92.18	89.06	86.25	83.12	70.62	66.87	58.43	33.75	23.75	18.75	18.75
25	89.37	85.93	85.62	85.31	85.62	85.31	85.31	86.56	86.56	86.25	86.56	89.37	90.62	90.31	89.37	86.56	83.43	70.31	61.56	58.43	37.50	25.00	21.87	21.87
26	86.25	85.93	85.93	85.93	85.93	85.93	85.93	85.93	85.93	85.93	84.68	84.68	87.50	88.43	88.75	87.81	87.50	79.68	65.00	61.56	56.87	40.00	25.31	16.87
27	86.56	86.25	84.68	84.68	85.00	85.00	85.00	85.00	85.00	85.00	85.00	86.25	89.06	90.31	90.31	92.81	92.18	85.93	68.75	60.93	56.56	38.75	27.50	18.75
28	85.00	85.93	85.00	84.68	84.37	84.37	84.37	84.68	84.06	85.00	86.25	88.12	88.75	91.87	93.43	90.62	90.62	74.06	60.93	52.50	37.18	25.00	13.75	13.75
29	89.68	85.62	85.00	85.00	85.00	85.00	85.00	85.00	85.00	85.00	85.00	86.56	88.75	89.06	89.37	91.25	90.93	90.31	77.18	66.25	55.31	34.37	22.50	15.00
30	89.37	86.25	85.00	85.00	85.00	84.68	85.00	85.00	84.68	85.62	85.93	88.43	89.06	90.31	91.25	92.18	90.00	77.18	69.06	60.62	35.31	25.62	21.25	21.25

In equation form it is

$$P = \frac{100 + R_d\% - R_f\%}{2}. \quad (5.1)$$

In Figure 5.13, the performance matrix is shown for the same group of filters as in Table 5.2 and Figure 5.12.

The MACE filter's generally poor performance on this data is quite clear from Figure 5.13. The most significant features in this figure are the bands of better performance: a horizontal one around the filter radius 10, and a vertical one around the c_{em} -value of about 5×10^{-5} . The area where the bands cross, however, does not appear to benefit from both influences.

The statistics on the training set size required to create each sample filter are displayed in Figure 5.14. Recall that true-class training images are singly added to the filter, until the filter reaches a certain threshold detection rate percentage on the whole training set. The eventual training set size is, therefore, closely tied to the filter parameters and capable of confirming some of the theoretical predictions. The figure shows that very large training sets are required to train a MACE-like filter. This is a direct consequence of noise vulnerability and poor intraclass recognition, and explains why the detection rates of this filter are so poor. Conversely, the figure shows an expected decrease in eventual training set size as the minimization weight nears the MVSDF filter. It is also notable how the

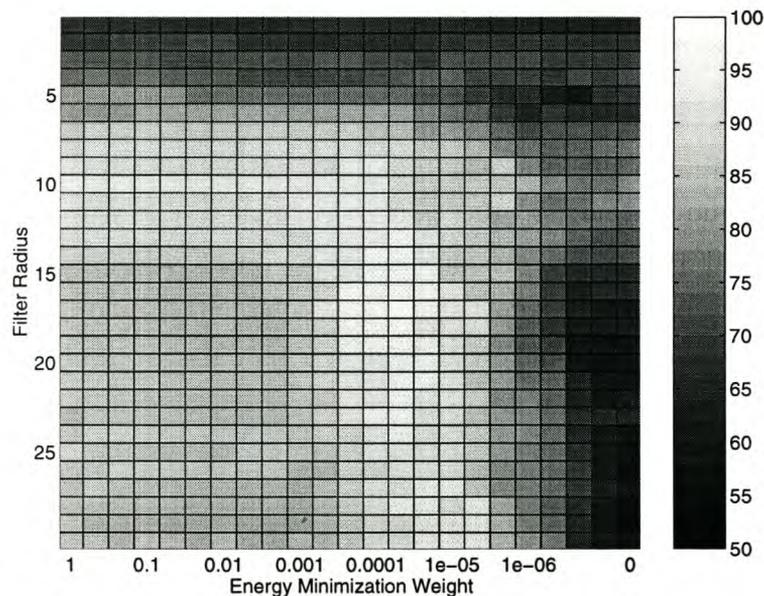


Figure 5.13 Performance for “all” data set, as influenced by filter parameters.

plotted curves of the detection rate (in Figure 5.12) and the training set size are, inversely, of almost identical shape. All these observations show that the eventual training set size is a direct indication of the ease with which unseen images will be detected.

The last observation recorded in this experiment is the condition number of the filters. Theoretically, a large condition number indicates a sensitivity to noise. Figure 5.15 shows that this number lies between 10^2 and 10^6 for the usable part of the experiment range. In this range it does not seem to have any influence on the components of performance.

The influence of the object size distribution is examined using all of the statistical observations, but only the performance observation changes significantly between the four data sets described in Table 5.2. The “large” performance matrix is shown in Figure 5.16, the “medium” performance matrix is shown in Figure 5.17, the “small” performance matrix is shown in Figure 5.18, and compare them also to the performance matrix of the “all” data set in Figure 5.13. These figures show that the data sets can be arranged in a decreasing order of overall performance, and simultaneously in an increasing order of sensitivity to c_{em} -changes: “large” – “medium” – “all” – “small”. The only conclusion is that the smaller diamonds included in the “all” data set must have a significantly adverse effect on the data set’s overall performance.

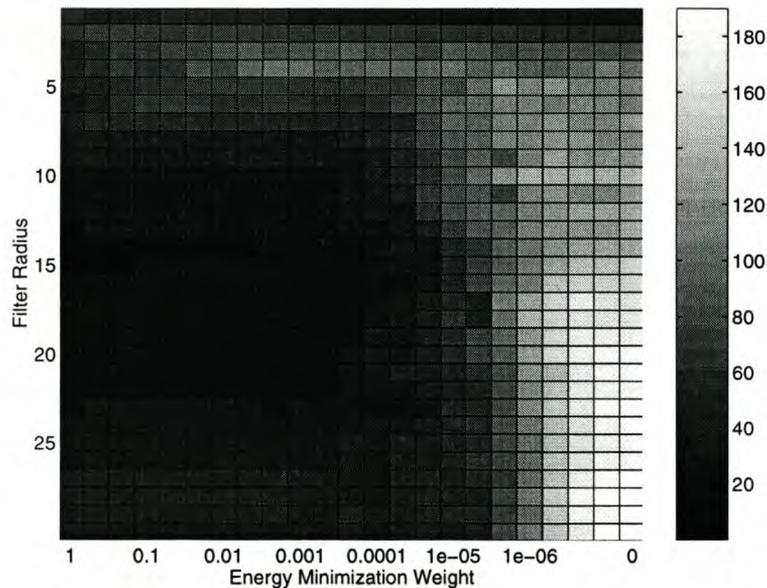


Figure 5.14 Training Set Size for “all” data set, as influenced by filter parameters.

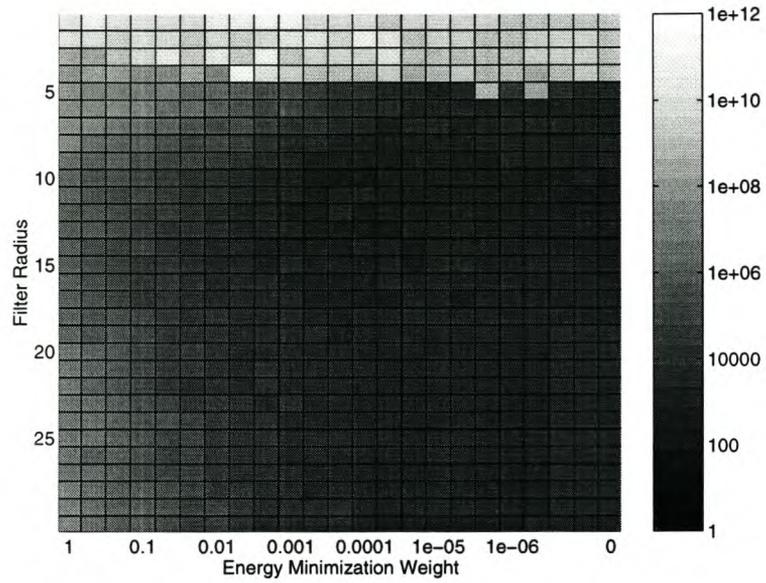


Figure 5.15 Condition Number for "all" data set, as influenced by filter parameters.

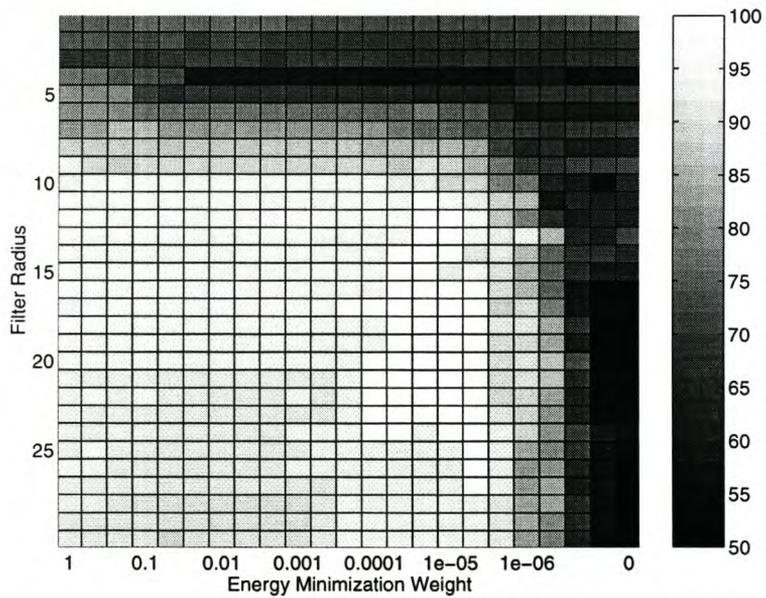


Figure 5.16 Performance for "large" data set, as influenced by filter parameters.

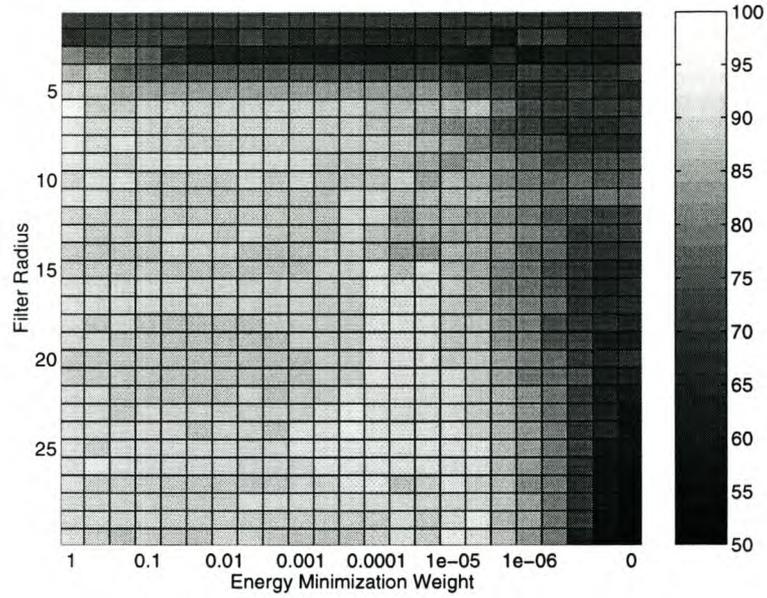


Figure 5.17 Performance for “medium” data set, as influenced by filter parameters.

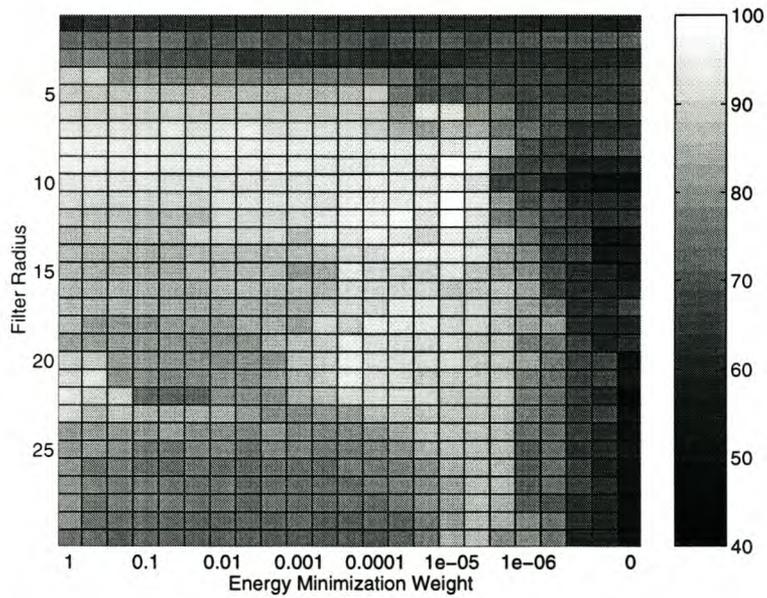


Figure 5.18 Performance for “small” data set, as influenced by filter parameters.

5.2.2 Case studies

A full source image is very big, and therefore a cutout is made of a region containing diamonds from the testing set. The cutout is shown in Figure 5.19. The grid of diamonds is marked on its 4 corners by 5 coins.

No guarantee can be given on the intensity range of the output generated by the MINACE filter. To allow for the direct comparison of the outputs, histogram clipping is used to display only the grayscale range between 0 and 2 in these figures. This is a useful range, since one of the true-class images from the eventual training set will yield an origin correlation peak intensity of 1, and the detection rate statistics are generated using a threshold of 0.7. The output area containing edge effects is set to zero in each image.

Best filter from “all” data set

This filter is chosen from the performance statistics in Figure 5.13, and the results are summarized in Table 5.3. The filtered result using the test image is shown in Figure 5.20.

Although the grid containing the diamonds is highlighted, so unfortunately is much of the

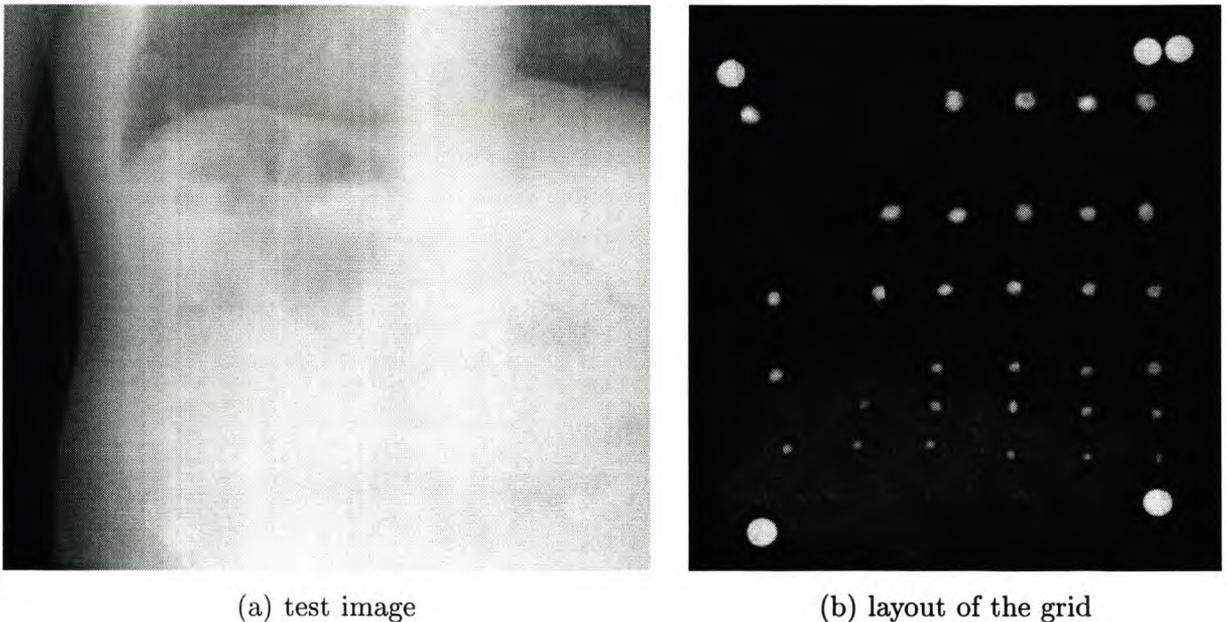


Figure 5.19 The image used for testing the filters. This a cutout from the source image “AddJuly1999/ADDtestImage5”.

Table 5.3 Parameter summary for best filter from “all” data set.

filter size	c_{em}	performance	detection rate %	false alarm rate %	true-class training set size	condition number
17	10^{-4}	93.93	96.25	8.39	17	6.7×10^2

background clutter. Since diamonds have no distinguishing features, other than a localized shadow, all features with these characteristics are highlighted by the filter. Figure 5.20 also shows that, although the image channel boundaries do affect the filter, the detection outcome is not significantly influenced.

Energy minimization weight investigation

This example illustrates the effect of c_{em} . This parameter is the only difference from the previous example. The results are summarized in Table 5.4, and the filtered test images are shown in Figure 5.21.

The figure shows that filter (a) is less sensitive to small changes in the input (like noise)

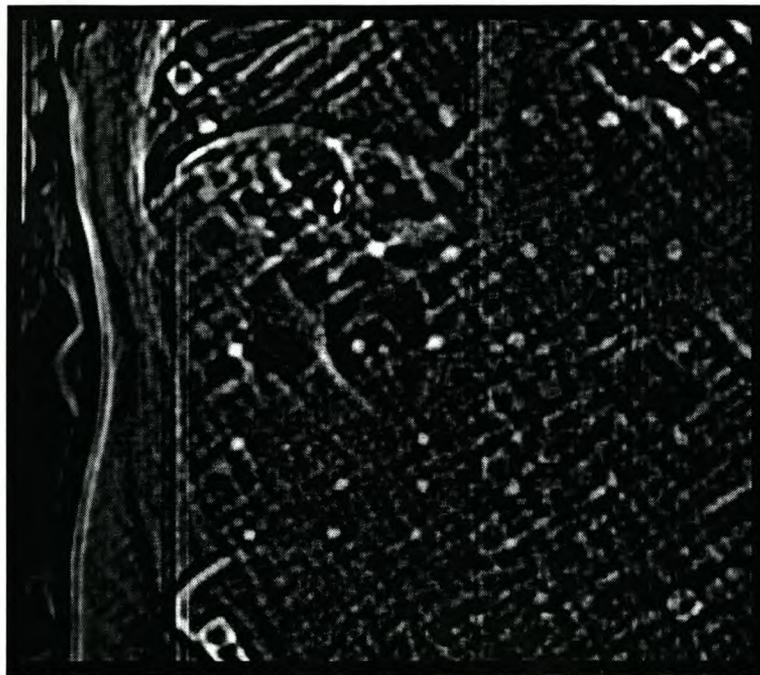
**Figure 5.20** Positive test image output for best filter from “all” data set.

Table 5.4 Parameter summary for energy minimization weight investigation.

	filter		performance	detection rate %	false alarm rate %	true-class	condition number
	size	c_{em}				training set size	
(a)	17	10^{-3}	86.64	91.56	18.28	10	9.18×10^2
(b)	17	10^{-5}	91.8	87.5	3.89	46	1.1×10^3

than the filter in Example 1, and also emphasizes clutter more readily. Filter (b) differs in the opposite way from the filter in Example 1. It can be concluded that filter (a) is a more MVSDF-like filter than the filter in Example 1, and filter (b) is more MACE-like. The best MINACE filter indeed seems to be where c_{em} has the best balance between signal energy minimization and noise energy minimization.

Best filter from “large” data set

This filter is chosen from the performance statistics in Figure 5.16. The results are summarized in Table 5.5, and the filtered test image is shown in Figure 5.22.

The larger diamonds are indeed well marked, and with narrow peaks — the advantage of a low c_{em} -value. Interestingly, this filter is better at clutter rejection than the filter from the “all” data set. The only conclusion is that the size and shape of the gradient for

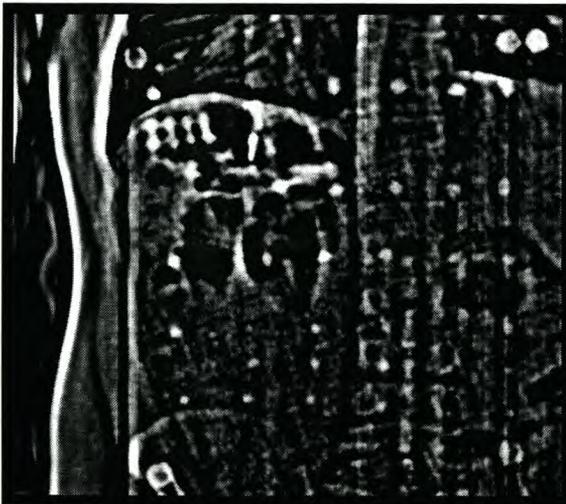
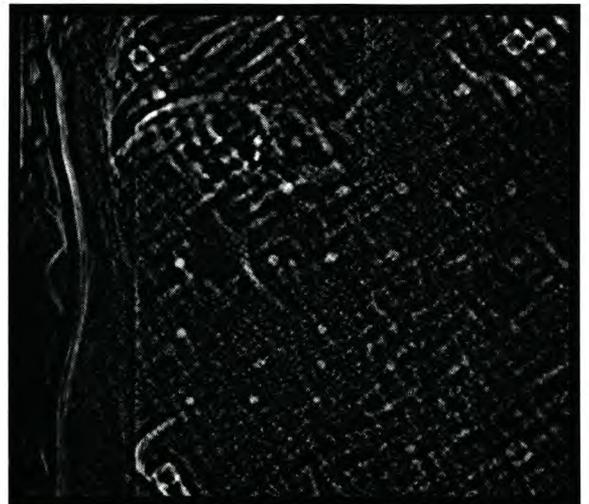
(a) $c_{em} = 10^{-3}$ (b) $c_{em} = 10^{-5}$ **Figure 5.21** Positive test image outputs for energy minimization weight investigation.

Table 5.5 Parameter summary for best filter from “large” data set.

filter size	c_{em}	performance	detection rate %	false alarm rate %	true-class training set size	condition number
19	5×10^{-6}	99.56	100	0.89	19	2.1×10^2

which this filter is trained, correlates less well with the clutter in this image.

Best filter from “medium” data set

This filter is chosen from the performance statistics in Figure 5.17. The results are summarized in Table 5.6, and the filtered test image is shown in Figure 5.23.

Again, the diamond size range of this filter is well highlighted, and the clutter performance is better than that of the “all” data set. Since this is the middle one of the three diamond size ranges, the diamonds from the “big” and “small” data sets are also fairly well highlighted.

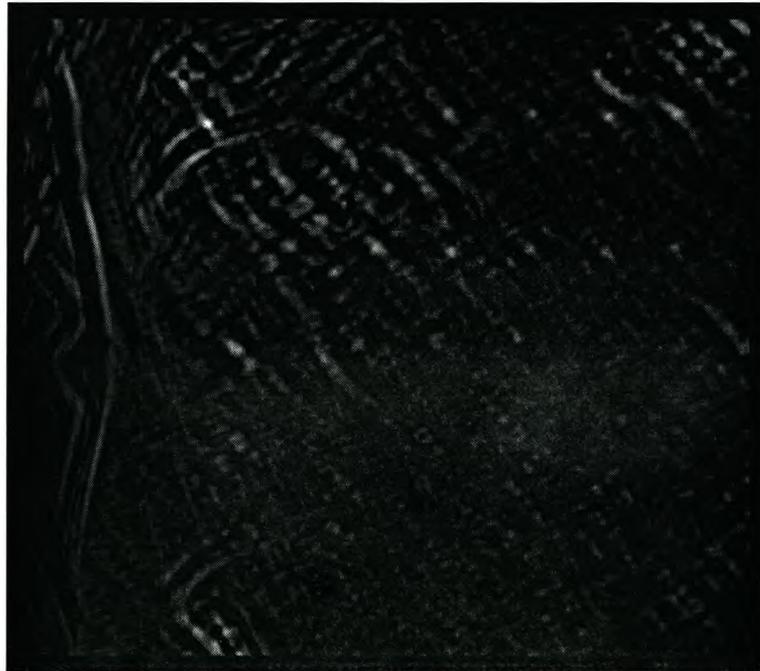
**Figure 5.22** Positive test image output for best filter from “large” data set.

Table 5.6 Parameter summary for best filter from “medium” data set.

filter size	c_{em}	performance	detection rate %	false alarm rate %	true-class training set size	condition number
23	2×10^{-4}	94.11	91.11	2.89	20	5.8×10^2

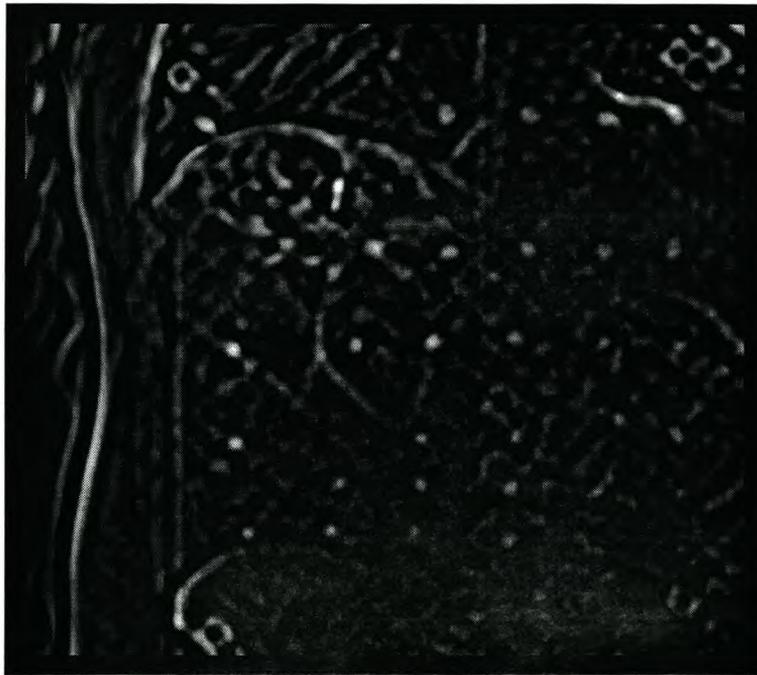
Best filter from “small” data set

The results of using filters trained for the “small” diamond size range are summarized in Table 5.7. The filtered test images are shown in Figure 5.24.

Table 5.7 Parameter summary for best filters from “small” data set.

	filter size	c_{em}	performance	detection rate %	false alarm rate %	true-class training set size	condition number
(a)	9	10^{-5}	98.67	100	2.67	13	2.9×10^2
(b)	9	1.0	94.4	92.86	4.06	12	3×10^6

Filter (a) is the filter of choice as indicated by Figure 5.18, but here reality deviates

**Figure 5.23** Positive test image output for best filter from “medium” data set.

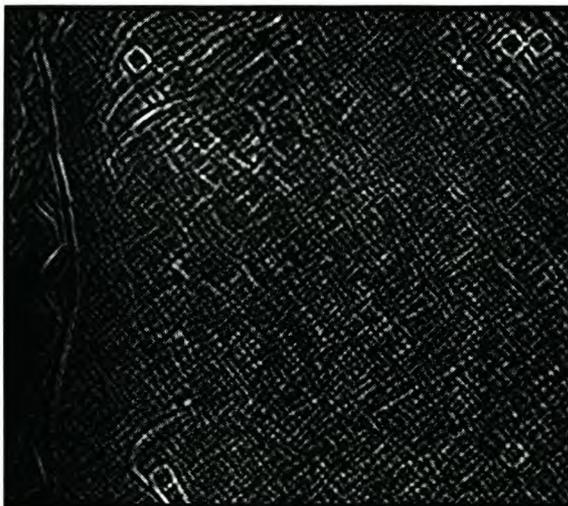
significantly from the statistical results. In the output, the features of the background clutter completely dominate the scene. Choosing a filter with a lower sensitivity for detail, Filter (b) is chosen as a full MVSDF filter. The larger c_{em} -value, as expected, improves the filter's handling of the fine clutter found throughout the body, but worsens the filter's handling of the edge-type clutter.

Test filter from “large” & “medium” combination data set

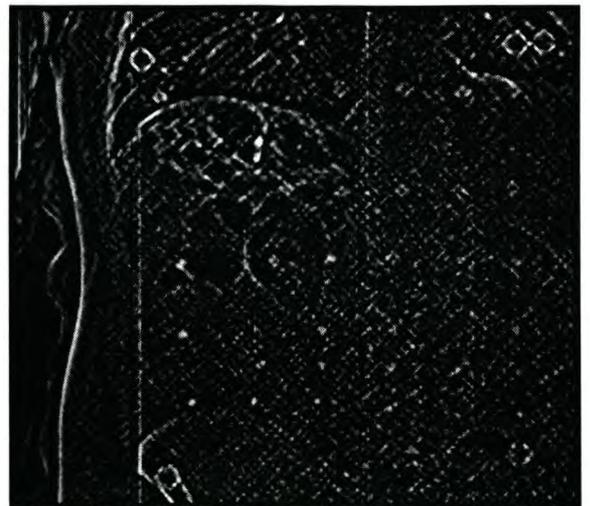
This filter is chosen in response to the observations so far. The idea is to make a new data set where the difficult diamonds have been excluded. The “large” and “medium” training sets are simply combined for the training of this filter, and the “all” testing set is used for the detection rate statistic. The size and c_{em} -value of the filter are chosen to be the same as for the best filter from the “all” data set. This filter is summarized in Table 5.8, and the filtered test image is shown in Figure 5.25.

Table 5.8 Parameter summary for test filter from “large” & “medium” combination data set.

filter size	c_{em}	performance	detection rate %	false alarm rate %	true-class training set size	condition number
17	10^{-4}	88.71	80.31	2.90	26	7.84×10^2



(a) $c_{em} = 10^{-5}$



(b) $c_{em} = 1.0$

Figure 5.24 Positive test image outputs for best filters from “small” data set.

The filtered output is good, although it has a slightly worse clutter performance than the filter from the “medium” set. Even so, it is a safer choice for detection purposes to have the “large” data set included in the training. This filter is therefore chosen to be used in the final detection system. The filter is shown in Figure 5.26.

It is no surprise that even the final detection filter has difficulty in distinguishing the



Figure 5.25 Positive test image output for test filter from “large” & “medium” combination data set.

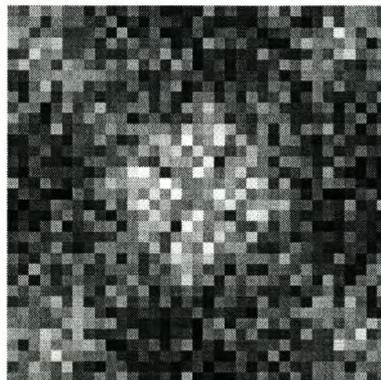


Figure 5.26 Normalized space-domain representation of the chosen MINACE filter, trained with the “large” & “medium” combination data set and a c_{em} -value of 10^{-4} .

featureless diamonds from the image background. Although this filter, as it is, can quite conceivably be the primary enhancement filter that an operator uses, there is benefit in segmentation and postprocessing.

5.3 Segmentation

All of the segmentation algorithms presented in Chapter 4 except for median-based local thresholding are considered for use in this application. There is not a big difference between the mean and the median characteristics, and calculating the median is computationally too expensive with a straightforward algorithm. The segmentation algorithms are tested on the chosen MINACE filter's output from Figure 5.25. In each case, the filtered output is first smoothed with a 7×7 averaging filter to avoid excessive region formation due to single pixels standing out from their environments. Regardless of the algorithm, a threshold value of $T = 0.7$ is used in the first part of this investigation, and a local window size of 35×35 is used where applicable. This initial choice of window size is to keep the effect of local thresholding modest, to see whether it improves on global thresholding at all.

The superimposed result of global thresholding is shown in Figure 5.27. At this threshold value, there are no problems with region lumping, and only 6 diamonds are undetected (3 medium and 3 small). When viewing the full grayscale histogram of the output, the large negative peaks become visible. These areas tend to be close to the positive peaks, and the solitary diamond peaks are almost surrounded with a negative region. Therefore, the detection rate is expected to suffer little from region lumping except at very low threshold values. This was the main concern with global thresholding.

The superimposed result of mean-based local thresholding is shown in Figure 5.28. The peaks in the output tend to be fairly wide, and this influences the mean value of the local neighbourhood so that the effective threshold is somewhat higher. Likewise, close to the lower regions of the output, there is a visible increase in region formation.

The superimposed result of Niblack local thresholding is shown in Figure 5.29. $c = 0.3$ is used for the standard deviation weight defined in (4.3). Its value is always positive and pushes the effective threshold higher, depending on the local neighbourhood. It also counters the impact of the lower areas on the threshold, because of the greater landscape variance there.

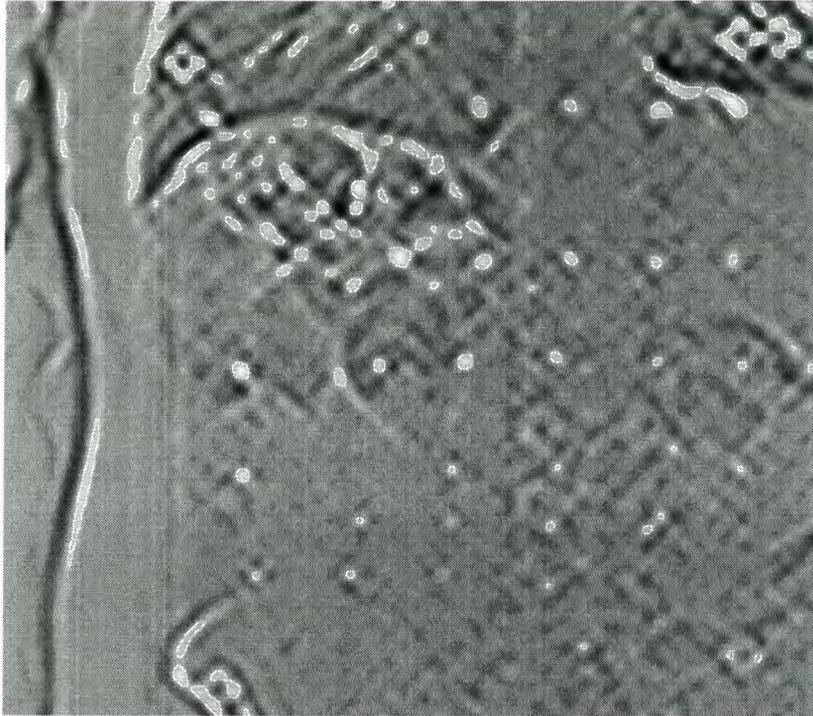


Figure 5.27 Demonstrating global thresholding on filtered output.

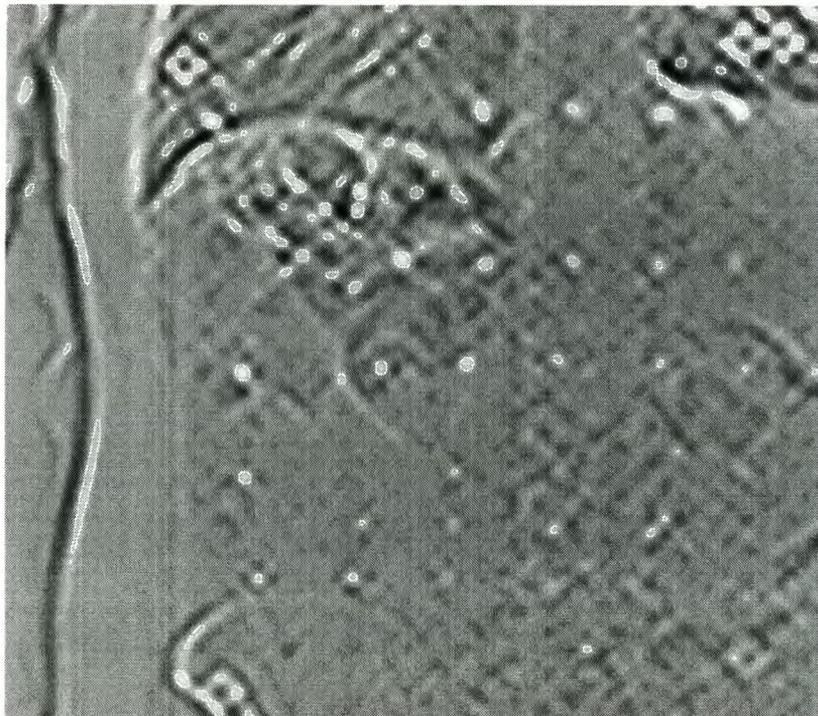


Figure 5.28 Demonstrating mean-based local thresholding on filtered output.

The superimposed result of watershed-based thresholding is shown in Figure 5.30. Each region r is histogram clipped at an intensity distance of $H = 1.0$ from its peak value p_r . This means that all region pixels with intensities $u \notin [p_r - 1.0, p_r]$ are discarded. This intensity distance is chosen so that even regions with sharp peaks still identify some of the surrounding slope (the influence zone).

In this improved representation of the watershed regions, there is only one diamond region with an adjacent region. It seems as if the low correlation values around diamond peaks really are useful in isolating the peaks, and this knowledge can be exploited. Discarding all regions that have neighbours may be a bit overzealous at first, so all regions with neighbours that have higher peaks are discarded. No diamonds are lost, and the false alarm rate is reduced. Since the focus will now shift to the shape of the correlation peaks instead of the influence zone, the histogram width of each region is reduced to $H = 0.5$. The result from these two steps is shown in Figure 5.31.

To generate detailed statistics on these techniques, the full source image of the testing cutout is now considered. The chosen filter and segmentation algorithms are evaluated for the range of threshold values $T = \{0.2, 0.3, \dots, 1.2\}$. Figure 5.32 shows the curves

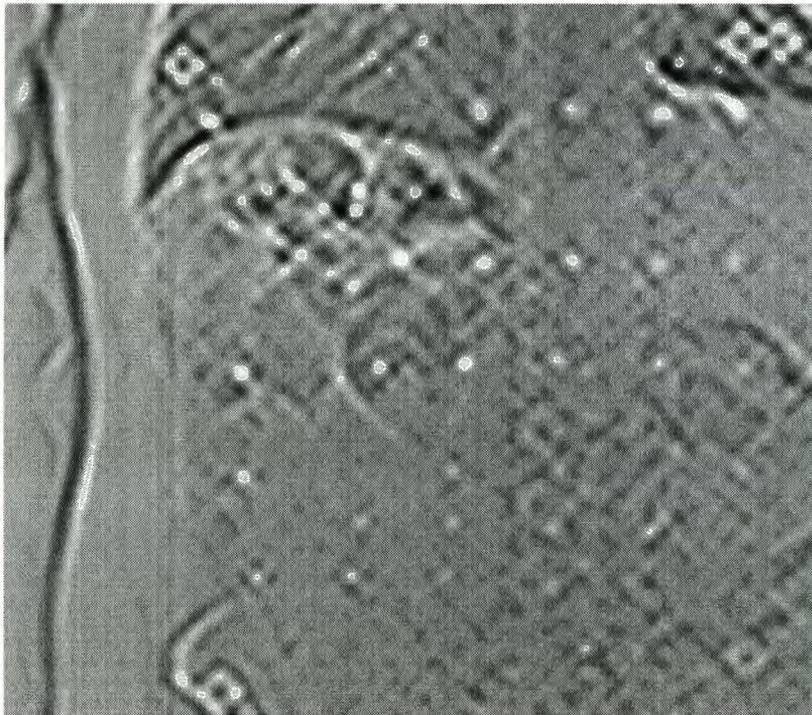


Figure 5.29 Demonstrating Niblack local thresholding on filtered output.

for the four segmentation algorithms when plotting detection rate against the number of false alarms. The lower threshold values are at the top right with high detection rates and many false alarms, and vice versa. A diamond is considered to be detected if a region peak is within a certain distance of the diamond centre. This distance is 4 for “small” diamonds, 6 for “medium” diamonds and 11 for “large” diamonds, in each case the smallest diamond radius expected from the size class.

In Figure 5.32, the watershed algorithm’s curve does not reach 100% detection because of the cluster elimination postprocessing, but this is the price of an improved false alarm rate. It performs very similar to global thresholding, while the two local thresholding techniques are somewhat poorer. The turning point in the curve at the top left is the best trade-off, where the algorithms achieve around 85% detection with 750 false alarms. Note that the detection rates for the global and local thresholding algorithms have not yet started to drop for the low threshold of 0.2 at the top right, as could be expected from the region lumping effect. There is no doubt that region lumping does occur at that threshold, and the only explanation is that the diamond peaks do indeed tend to stand out from the peaks caused by their cluttered surroundings. Also, the isolating effect of the

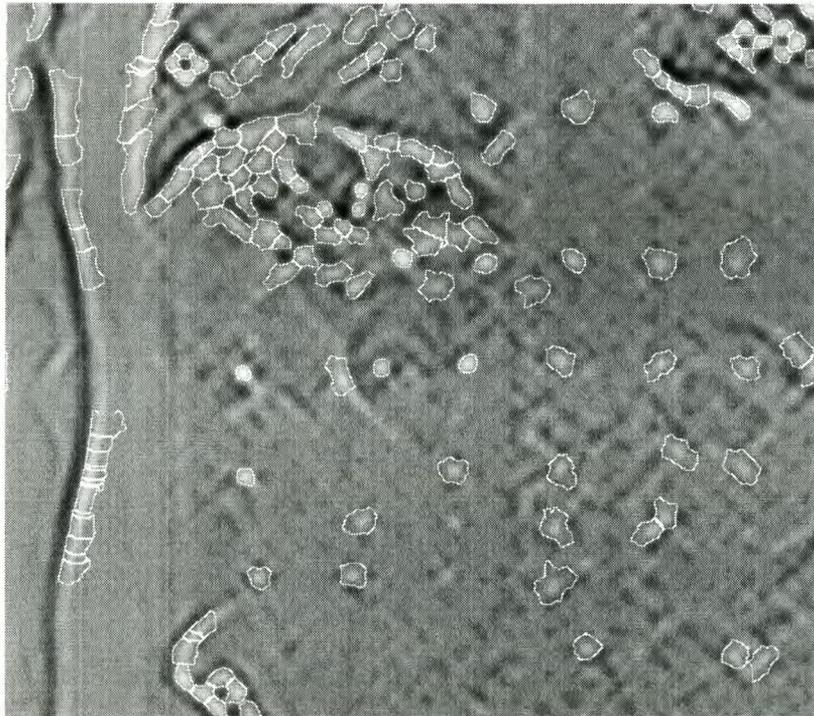


Figure 5.30 Demonstrating watershed-based thresholding on filtered output. Each region has a histogram width $H = 1.0$ to indicate region influence.

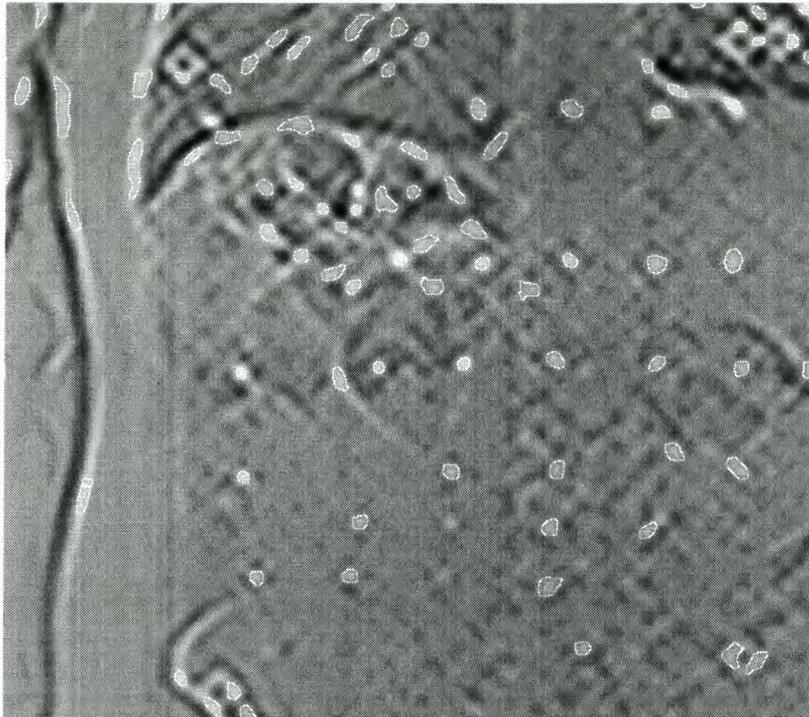


Figure 5.31 Demonstrating controlled cluster elimination after watershed-based thresholding. Each region's histogram width is now reduced to $H = 0.5$ to focus on peak shapes.

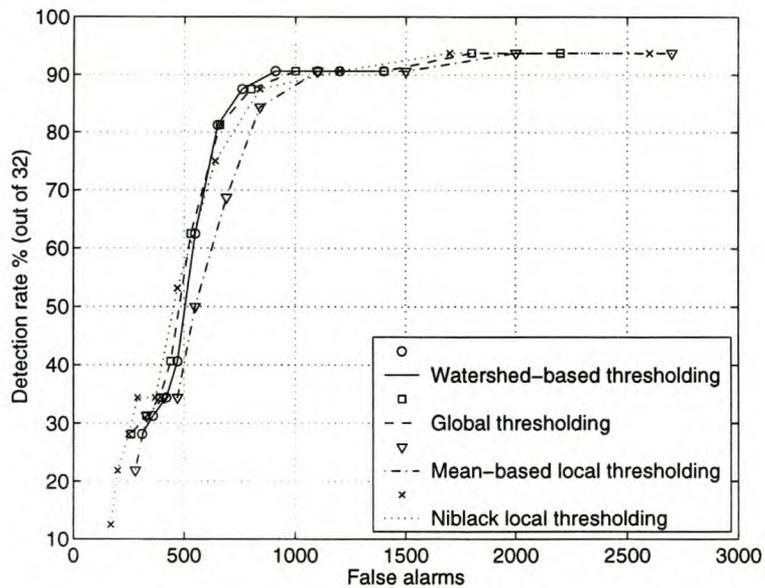


Figure 5.32 Comparing the performance of the four segmentation techniques for various threshold values.

lower correlation values observed around diamond peaks contributes to this phenomenon.

At this stage the detector has a choice of four algorithms with which to segment the MINACE peaks. The segmentation algorithms perform much alike so far, and a final decision between them cannot yet be made. A segmented image has the additional advantage of allowing for more accurate statistics than those available from the training process. These statistics reveal a concerningly high false alarm count of 750 for a normal image, and the focus of the design now shifts from detecting diamonds, to reducing false alarms.

5.4 Reducing false alarms

Two techniques are tested here. They examine the shape of each region, as identified by any of the segmentation algorithms, and are designed to have a minimal impact on the detection rate. The first technique examines the spatial distribution of a region by using principle component analysis, and the second technique examines region contours. Both techniques test the assumption that a desirable region has the shape of a disc. This assumption cannot be applied too strictly, because the correlation peaks of larger diamonds tend to be somewhat irregular. Also, the presence of clutter influences the formation of correlation peaks adversely. Once both techniques are implemented, the cluster elimination addition to watershed-based thresholding is re-examined.

5.4.1 Principle components

Using the Karhunen Loeve (KL) transform [22], the principle components of each region is analyzed. The reason for using the KL transform can be understood if the set of pixel co-ordinates that makes a region is interpreted as the discrete sample vector sequence of a 2-dimensional process. The basis vectors of the region, and of the KL transform of the process, are the orthonormal eigenvectors of the covariance matrix of the co-ordinate sequence. (The covariance matrix can only be used if the vector sequence is made zero mean, but this is trivial to do.) It is not necessary to carry through the transform however, because this routine only requires the result of the eigenanalysis.

The two eigenvalues of the covariance matrix indicate the weight (or length) of the basis vectors, and the relation between the two eigenvalues (the biggest, λ_b , and the smallest, λ_s) is calculated as $\frac{\lambda_s}{\lambda_b}$. For a perfectly circular object this relation is 1. If it is less than

0.25, the region is discarded. This decision threshold is experimentally determined to accommodate larger diamonds in clutter.

The performance comparison plot of Figure 5.32 is repeated in Figure 5.33 for the detection and false alarm statistics after this postprocessing stage. The same range for the threshold is used. Although the asymptote of the detection rate is about 7% poorer, the approximate number of false alarms at the turning point of the curves is now 350. This is less than half of the original number of 750. Two other observations are also of interest. Firstly, the global thresholding algorithm has the same detection rate value for most part of the threshold value curve as watershed-based thresholding, but a better false alarm rate. Secondly, the global and local thresholding curves show a breakdown from the detection rate asymptote. At low threshold values the inevitably poor region shapes of those algorithms do not pass the principle component postprocessing.

5.4.2 Contour variance

The principle component postprocessing only considers the pixel distribution along the basis vector axis, and can be fooled by curved regions which are obviously wrong to the eye. This routine specifically addresses that possibility. It calculates the distances between

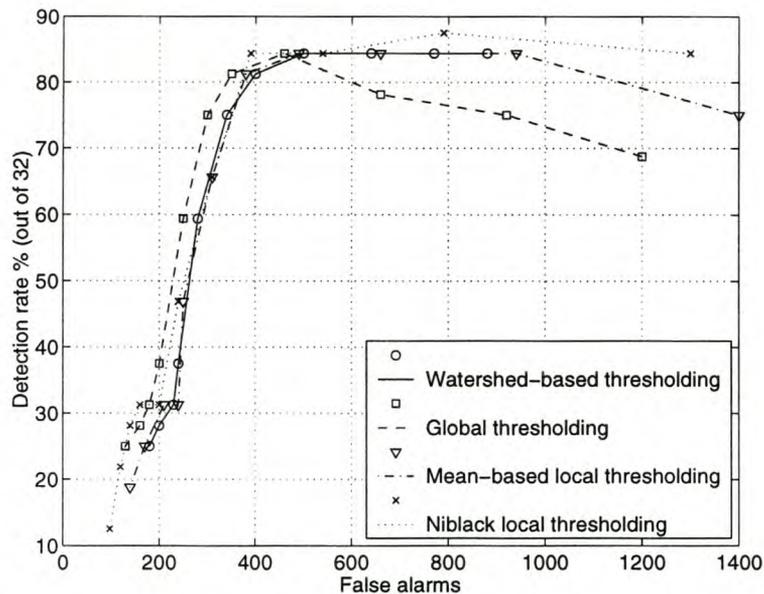


Figure 5.33 Comparing the performance of the four segmentation techniques for various threshold values, after principle component postprocessing.

the region centroid and every pixel along the region contour, and inspects the variance of this set of distance values. If the variance is greater than 4, the region is discarded. Again, this value is experimentally determined to minimize the loss of true-class regions.

The statistical performance analysis is repeated, and the results are shown in Figure 5.34. This addition to the postprocessing does not make a big difference. Closer inspection reveals that the false alarm rates of all plot points have improved somewhat. Also, watershed-based thresholding benefits a little bit more than the other algorithms, especially for the lower threshold values.

These two regional postprocessing steps improve the performance of the detector regardless of the segmentation algorithm used. It is possible to achieve different trade-offs by adjusting the two postprocessing parameters, but the first diamonds to disappear are some of the large diamonds with irregular correlation peak shapes. Apart than that, the only other adjustable parameter, without introducing new processing concepts, is the aggressiveness of the cluster elimination after watershed-based thresholding.

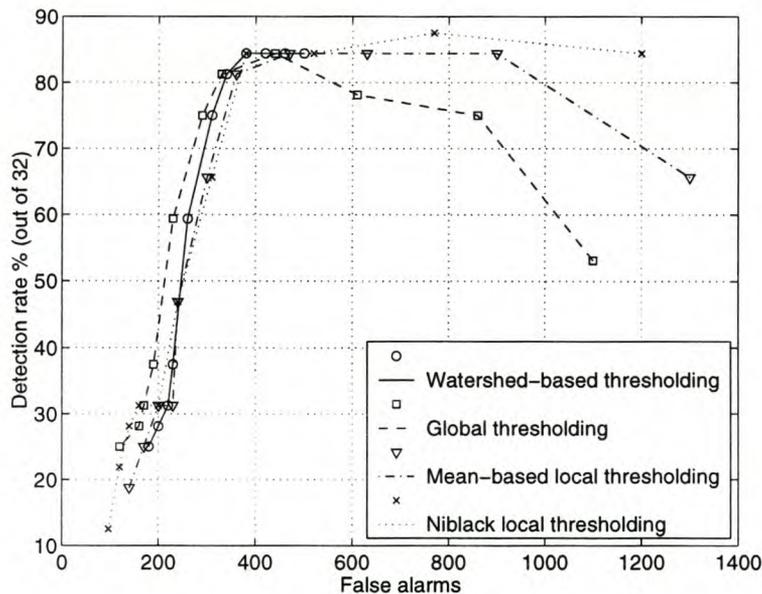


Figure 5.34 Comparing the performance of the four segmentation techniques for various threshold values, after principle component and contour variance postprocessing.

5.4.3 Cluster elimination

It is clear from Figure 5.30 that the assumption about diamonds and region clustering is valid for this image and for this threshold value, because only one diamond has a neighbouring region. Here, the watershed algorithm's ability to separate neighbouring peaks is now a liability. To rectify this situation, and to start testing the real validity of this observation, a variation of the cluster elimination routine is examined. This variation rejects all regions that have neighbours. Since the concept of a region neighbour depends on the intensity distance H with which a region's histogram is clipped (Section 5.3), full cluster elimination is also tested with an alternate choice for H , namely 0.7. Threshold value curves are plotted in Figure 5.35 for the performances of the various cluster elimination approaches after watershed-based thresholding and after principle component and contour variance postprocessing.

Figure 5.35 shows the severe penalty in false alarm numbers if cluster elimination is not done. Interestingly, the new cluster elimination curves do not follow the previous paradigm, and reach their peaks in performance at the threshold values of 0.6 and 0.7. This happens because for low threshold values the diamond regions start acquiring neigh-

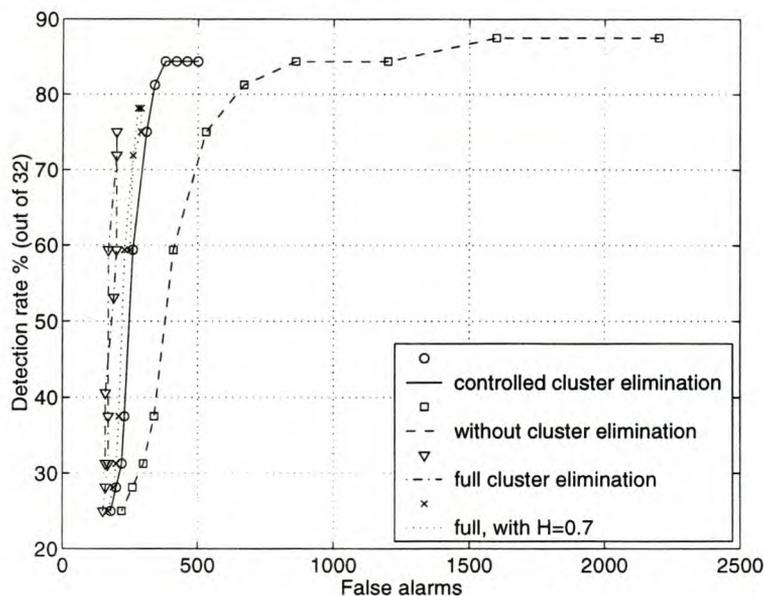


Figure 5.35 Comparing the performance of the four mentioned cluster elimination approaches after watershed-based thresholding, after principle component and contour variance postprocessing.

hours and are rejected. These performance peaks makes a new level of trade-off between detection rate and false alarm numbers possible, and the original histogram width of $H = 1.0$ is clearly the better trade-off.

These regional postprocessing routines seem like valuable additions to the detector so far. Some of the assumptions, especially those regarding the cluster elimination, still need to be tested on other source images to verify their validity. However, the discussion on the components of the detector is now complete.

5.5 Final evaluation

The detector so far consists of basic histogram preprocessing, a zero-mean MINACE filter trained with a combination data set of “large” and “medium” diamonds (as shown in Figure 5.9 and Figure 5.10) and a c_{em} -value of 10^{-4} , a 7×7 averaging filter, and one of several segmentation algorithms. The execution times of the various components of the detector are evaluated in Table 5.9. The improvement when using the simple cropping option mentioned with the preprocessing in Section 5.1.1, is also investigated. Note that modern CPUs can improve on these execution times significantly (refer to Appendix A for implementation detail).

Cropping the source image is simple, but makes a big difference in the execution time of the detector since most of this time is used on FFT's. The execution time of Niblack local thresholding is not shown because the straightforward implementation is inefficient and not competitive. If the results of this technique proved to be superior, a more efficient implementation could be used. One such alternative, although less accurate, is to subdivide the image into blocks and to calculate the adaptive threshold for each block only. However, in this application, global thresholding is superior to the two local thresholding algorithms in every aspect. Watershed-based thresholding and mean-based local thresholding are much slower than global thresholding, but still twice as fast as the rest of the detection routine. Watershed-based thresholding only seems to have an advantage in this application when postprocessing like cluster elimination is used.

As a final performance evaluation, threshold value curves are plotted for the other two source images with 32 diamonds, in Figure 5.36 and Figure 5.37. Both of these two images (especially “AddJuly1999/ADDtestImage3”) contain diamonds used to train the MINACE filter. Although this fact has very little effect on the final detection rates (they

Table 5.9 Approximate execution times of the various stages of the detector, using the “AddJuly1999/ADDtestImage5” source image and a threshold value of 0.7 (refer to Appendix A for implementation detail).

	execution time (s)	
	without cropping (image size 5218×2400)	with cropping (image size 5096×1940)
preprocessing	17	18
MINACE & smoothing filter	93	70
<i>thresholding routines:</i>		
global	12	10
mean-based local	79	58
watershed	55	46
principle component postprocessing	4.4	3.3
contour variance postprocessing	7	5.5
<i>totals:</i>		
using global	133	107
using mean-based local	200	155
using watershed	176	143

are much more dependent on the postprocessing trade-offs), these plots are useful mainly to confirm the trends observed in the various segmentation techniques, and to gather more false alarm statistics.

It turns out that the only unexpected result in these two figures is in Figure 5.36, where the detection rate breakdown for low threshold values occurs very early for both global and mean-based local thresholding. Otherwise, the figures show that the assumption on which full cluster elimination is based, is correct. It also looks as if the original “AddJuly1999/ADDtestImage5” has the poorest false alarm performance of them all. Looking at the output, this is explained by the presence of the subject’s watch, keys and something else (which may be a notebook), neither of which is present in the source images “AddJuly1999/ADDtestImage3” and “AddJuly1999/ADDtestImage4”.

To summarize these results, a final filter is chosen for each thresholding approach and these filters’ detection statistics are displayed in Table 5.10. The error bars in this table are calculated at a 90% confidence interval [30]. Although the detection rate statistics between the thresholding approaches are very similar, the best false alarm performance is obtained using watershed-based thresholding with full cluster elimination. The physical

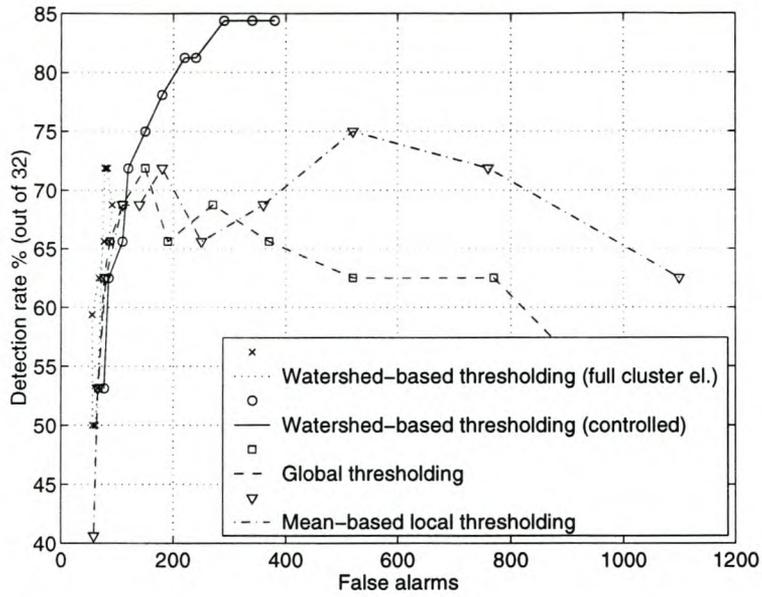


Figure 5.36 Comparing the performance of the four final segmentation techniques for various threshold values on source image “AddJuly1999/ADDtestImage4”, after principle component and contour variance postprocessing.

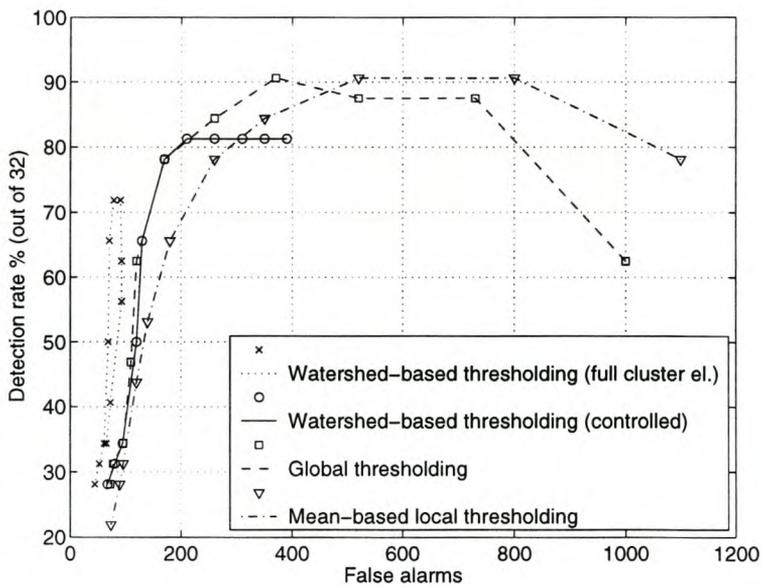


Figure 5.37 Comparing the performance of the four final segmentation techniques for various threshold values on source image “AddJuly1999/ADDtestImage3”, after principle component and contour variance postprocessing.

results of applying this choice of detector on a source image is shown in Figure 5.38. The filtered result shows a remarkable level of structural detail of the subject. Since the filter is trained on the surrounding edges of the diamonds, it emphasizes all edges. Not surprisingly, these edges tend to have both positive and negative output, appearing along opposing sides of the edge. When comparing the contrast of the grid of diamonds between the output image and the preprocessed source image (shown in Figure 5.6b), diamonds do seem to benefit the most from this filter.

5.6 Conclusion

This case study of diamond detection in X-ray images was expected to be challenging. The main reasons for this, i.e. diamonds' lack of prominent features and the clutter, are discussed extensively throughout this chapter. The approach proposed in this thesis is simple, robust, and fast.

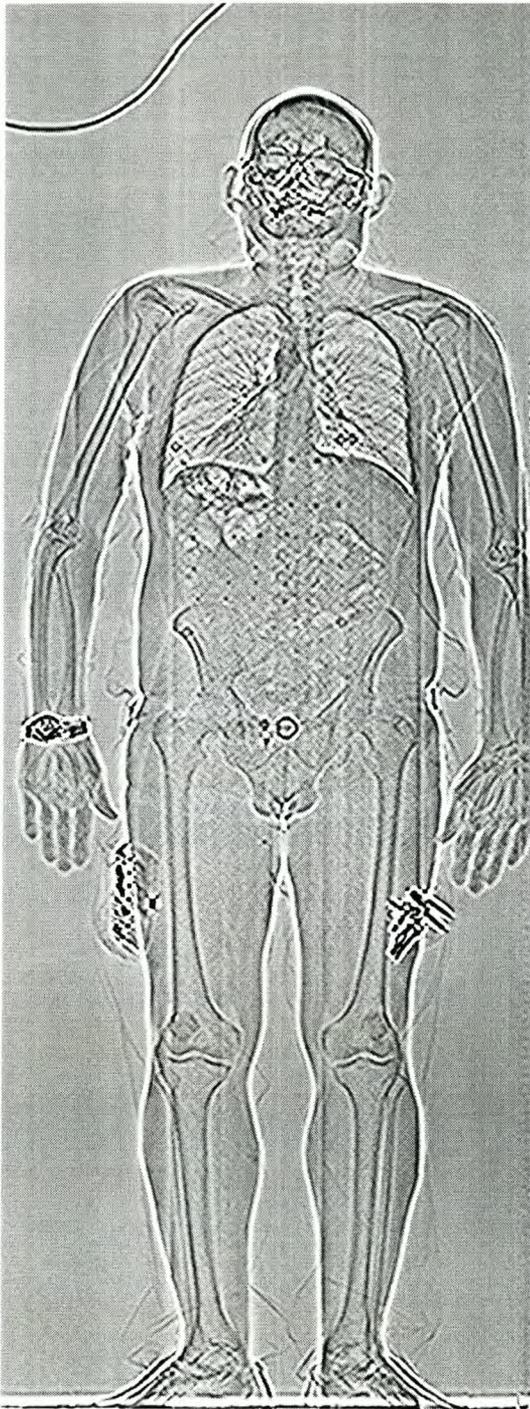
The design of this automatic diamond detector is centered around the MINACE correlation filter, with the necessary amount of processing between the various stages for functionality. The filter performs exactly as expected from a good correlation filter with a vague feature-description. A number of segmentation techniques with which the correlation output can be mapped as ROIs, is tested. In the course of this investigation, it is observed that the diamond peaks tend to be surrounded by low correlation value areas. Although this negates global thresholding's main disadvantage of poor peak separation, watershed-based segmentation can exploit this observation even further to achieve the best false alarm rates in this investigation. Regarding local thresholding, it must be concluded that this is not the right application for such a tool. Local thresholding simply generates too many peaks.

Once the segmentation stage provides a region mapping, the region shapes are processed to reduce the false alarms. Elongated or curved regions are discarded by the respective use of principle component analysis, and the examination of the variance of the region contour's distance from the region centroid. Although all of these false alarm reduction techniques are effective, they are far from sufficient for the detector to run unsupervised.

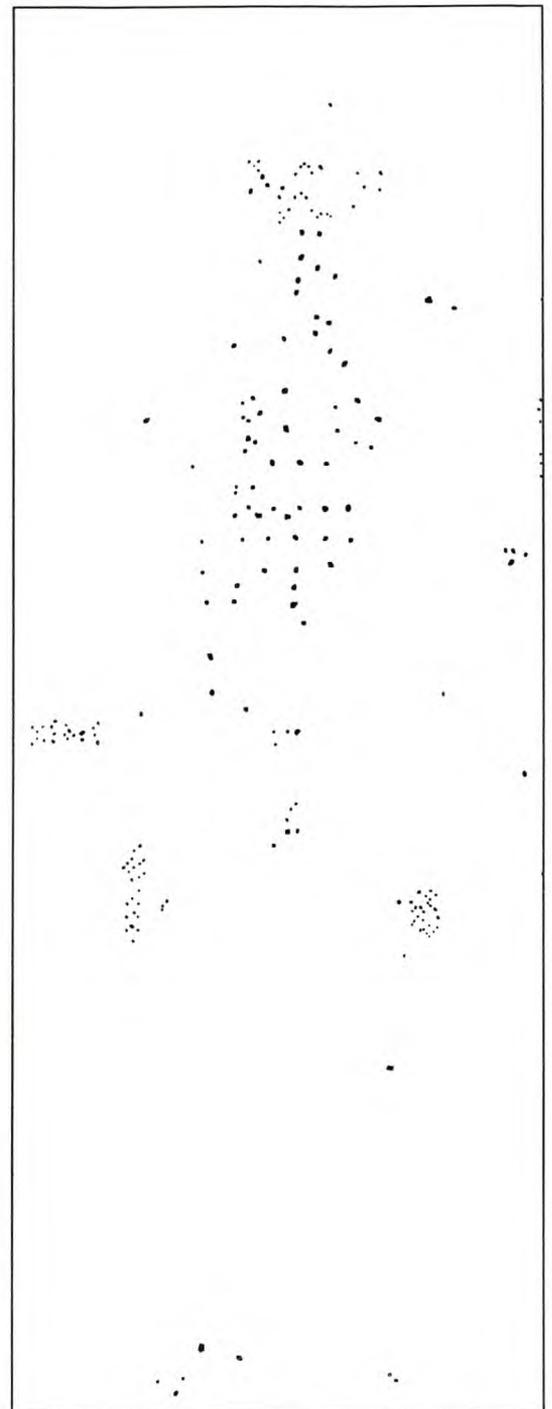
This case study is a good showcase of what the MINACE filter is capable of under difficult circumstances. Lack of features, abundance of clutter and very large images characterizes this problem. Although the output of the MINACE filter alone can be very useful to a

Table 5.10 Performance statistics for sample filters of the four segmentation techniques, on the three source images with 32 diamonds named “AddJuly1999/ADDtestImage i ” (with $i = \{3, 4, 5\}$). The threshold value T is 0.7 except where indicated, and the error bar percentages are at a 90% confidence interval.

segmentation technique	image no.	data set					false alarms
		large	medium	small	all	lrg. & med.	
watershed-based	3	7/9	13/17	3/6	$\frac{23}{32}$ (72% \pm 10%)	$\frac{20}{26}$ (77% \pm 11%)	79
thresholding	4	8/10	11/15	3/7	$\frac{22}{32}$ (69% \pm 11%)	$\frac{19}{25}$ (76% \pm 11%)	91
(full cluster el.)	5	9/10	13/18	1/4	$\frac{23}{32}$ (72% \pm 10%)	$\frac{22}{28}$ (79% \pm 10%)	198
	3, 4, 5	$\frac{24}{29}$ (83% \pm 9%)	$\frac{37}{50}$ (74% \pm 8%)	$\frac{7}{17}$ (41% \pm 15%)	$\frac{68}{96}$ (71% \pm 6%)	$\frac{61}{79}$ (77% \pm 6%)	
watershed-based	3	8/9	14/17	3/6	$\frac{25}{32}$ (78% \pm 9%)	$\frac{22}{26}$ (85% \pm 9%)	167
thresholding	4	9/10	13/15	3/7	$\frac{25}{32}$ (78% \pm 9%)	$\frac{22}{25}$ (88% \pm 8%)	185
(controlled)	5	9/10	14/18	1/4	$\frac{24}{32}$ (75% \pm 10%)	$\frac{23}{28}$ (82% \pm 9%)	306
	3, 4, 5	$\frac{26}{29}$ (90% \pm 7%)	$\frac{41}{50}$ (82% \pm 7%)	$\frac{7}{17}$ (41% \pm 15%)	$\frac{74}{96}$ (77% \pm 6%)	$\frac{67}{79}$ (85% \pm 5%)	
global	3	7/9	14/17	4/6	$\frac{25}{32}$ (78% \pm 9%)	$\frac{21}{26}$ (81% \pm 10%)	172
thresholding	4	8/10	10/15	3/7	$\frac{21}{32}$ (66% \pm 11%)	$\frac{18}{25}$ (72% \pm 12%)	191
	5	9/10	14/18	1/4	$\frac{24}{32}$ (75% \pm 10%)	$\frac{23}{28}$ (82% \pm 9%)	286
	3, 4, 5	$\frac{24}{29}$ (83% \pm 9%)	$\frac{38}{50}$ (76% \pm 8%)	$\frac{8}{17}$ (47% \pm 16%)	$\frac{70}{96}$ (73% \pm 6%)	$\frac{62}{79}$ (79% \pm 6%)	
mean-based	3	6/9	15/17	4/6	$\frac{25}{32}$ (78% \pm 9%)	$\frac{21}{26}$ (81% \pm 10%)	259
local thresholding	4	9/10	10/15	2/7	$\frac{21}{32}$ (66% \pm 11%)	$\frac{19}{25}$ (76% \pm 11%)	246
($T = 0.6$)	5	9/10	16/18	1/4	$\frac{26}{32}$ (81% \pm 9%)	$\frac{25}{28}$ (89% \pm 8%)	363
	3, 4, 5	$\frac{24}{29}$ (83% \pm 9%)	$\frac{41}{50}$ (82% \pm 7%)	$\frac{7}{17}$ (41% \pm 15%)	$\frac{72}{96}$ (75% \pm 6%)	$\frac{65}{79}$ (82% \pm 6%)	



MINACE filtered source image, output clipped to $[-0.5, 1.0]$ and inverted



After all regional processing

Figure 5.38 Physical results when applying the chosen MINACE filter and watershed-based segmentation with full cluster elimination and regional postprocessing. The image is “AddJuly1999/ADDtestImage5” and the threshold value is 0.7.

trained operator, the segmentation stage may be a useful additional indication. It also allows the detector to be augmented easily with further ROI-based postprocessing.

The next case study will demonstrate the difference when using the now familiar object detection technique on a new problem.

Chapter 6

Case Study: Reading A Stamped Code

2D barcode technology is not a new commercial concept, and readers are available for purchase from 500\$ to 1500\$. The market supplies readers designed for print quality, static barcodes. The concept of dynamic barcodes has not yet evolved commercially, and the goal of this case study is to design a robust reader for a dynamic 2D barcode symbology. The design process is also a test of the ease of portability of the object detection approach in this thesis. An example of the dynamic stamp is given in Figure 6.1. The stamps are made with a handheld device and ink. The rotation of each square is adjustable, and this ability provides the dynamic variable.

The reader developed here consists of a preprocessing-, a detection- and a postprocessing stage. It is tested on a data set created with a prototype stamping device. The stamps

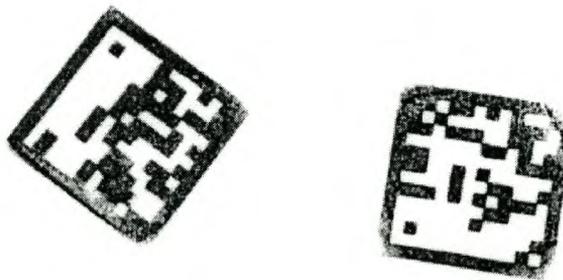


Figure 6.1 A scanned stamp.

for the data set are made on a variety of backgrounds, and are read with a scanner (to approximate data from a digital camera). The postprocessing stage uses the ROIs given by the detection stage, and relies on the principles of the Hough transform for the fine location and verification of the stamped codes. It measures the two co-ordinates, two rotations and appropriate side distances necessary to describe the detected stamp, and also estimates a confidence that the two ROIs indeed form a stamp.

A significantly more automated detection system is expected from this case study. The chapter starts with detailed descriptions of the stamps and the design environment.

6.1 Initial project stages

6.1.1 Introduction

The type of 2D codes used in this application is a variation of the DataMatrix¹. 2D codes are better space-optimized than their 1D counterparts and can carry more information. Better error-correction can therefore be used.

The stamp consists of two, square, outlined, binary matrices where each matrix can be at any of 12 distinct rotational positions. The pattern and rotation of each stamp carry information. Consider the example of a fruit-packing installation: the rotational information can be used to indicate the number of fruit in a box, and every packer should use a stamping device with a unique identification code. A computer connected to a digital camera positioned over a conveyer belt then records the packing statistics — type of product, date and the packer. The advantage of the rotational information is that it can be changed easily if for instance a different product has to be packed, simplifying the logistics.

The stamping device used for this design (shown in Figure 6.2), uses ink to make the stamp, and simply needs to be pushed down on an applicable surface. When pushed down, the stamping surface swivels outwards from an ink cushion. This allows the device to be used many times with a single application of ink, and also allows the ink to be stored for extended periods. The rotational positions of the two squares in the stamp are clearly visible in Figure 6.2. Overall, the functioning of the device is self-explanatory.

¹More detail on various 2D codes is readily available on the Internet [31]. Software is available (e.g. at <http://www.encode-solutions.com>) for the encoding of typical codes.

In this application the severity of the clutter that can be tolerated by the system is investigated, and a variety of cluttered packaging backgrounds is used. In practise, the stamps can be applied on a clear surface, indicated for this purpose.

The stamping device relies on ink to make the stamps, and therefore suffers from the expected range of distortions caused by drying ink. This effect is visible in Figure 6.1 and Figure 6.3. Fortunately, the edges, especially the corners, suffer the most from this distortion. This implies that, if a square can be located correctly, the chances are good that the inner part of the square (which contains the data) is healthy. Figure 6.3 shows two examples where one of the squares in each stamp is very poor. In the left stamp, the left square cannot be located correctly with the proposed detection algorithm (but the edge may be corrected with a black marker pen). In the right stamp, the left square has a damaged datamatrix (but if sufficient error correction is used in the code it may still be usable).

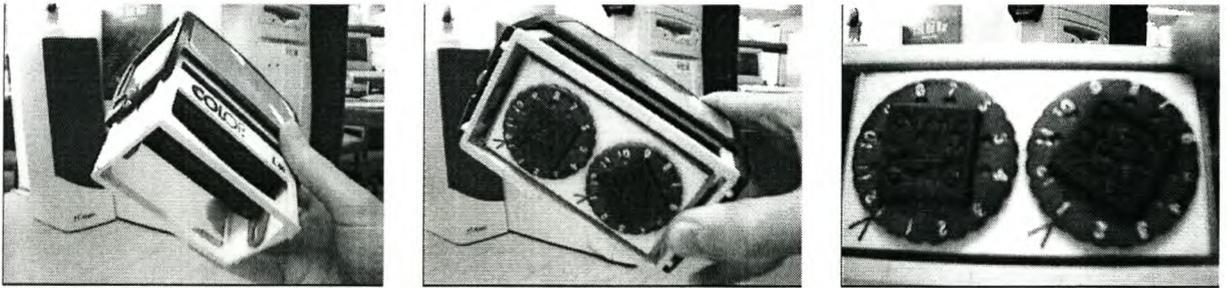


Figure 6.2 The stamping device.

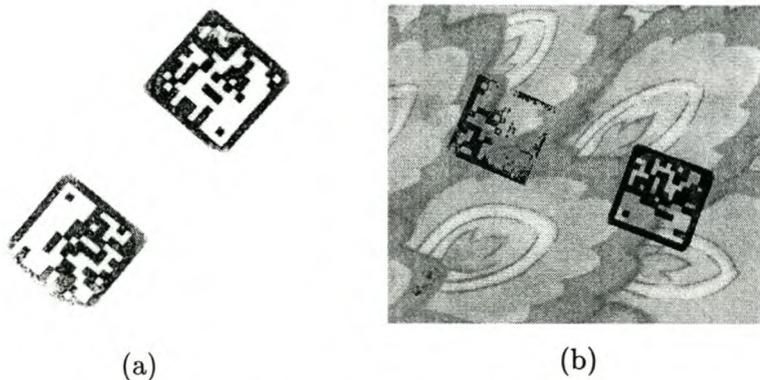


Figure 6.3 Some poor quality stamps.

6.1.2 Summarizing the project differences

- The two case studies in this thesis are dramatically different in operational environment and object type. In this design, the environment is less cluttered, and the objects possess more features. The clearer features of the stamps make it possible to implement a verification stage which fully automates the detection process.
- There is freedom in this prototype project to specify design criteria, while the diamond detection project is completely based on the given source image CD.
- The design process with this application is much faster. The MINACE filter is now familiar and there is no need to experiment as much with the different parameters.

6.1.3 Data collection

The data set needs to contain samples of stamps as they would appear in the application, in order to judge the success of this system accurately. Since this is a prototype, some of the operational characteristics, such as the digital resolution and the severity of the clutter, are arbitrarily chosen. The simplest way of creating this data set is by scanning in stamps from paper or carton. This is realistic, because the differences between scanning and using a digital camera are conditions like lighting and viewing angle, which can be controlled.

The physical data set includes two pieces of clean paper (Figure 6.4), two pieces of gift paper (Figure 6.5) with a continuous background, or light clutter, two pieces of packaging (Figure 6.6) with lettering and designs, or heavy clutter, two pieces of clear packaging where the one has normal stamps and the other very clearly-pressed stamps (Figure 6.7) and three more pieces of packaging where the ink was running low and the ink degradation can be tested (Figure 6.8). The rotational variable (Section 6.1.1) on the stamp is not adjusted much, but the stamps are made in any orientation. It is not necessary to have representative data on the rotational variable, because the detection filter looks for single squares and does not rely on the relative difference in rotation between the two squares.

The scanning process assumes a certain camera resolution and distance, because this is a prototype and the digital camera is not yet chosen. The squares are chosen to have sidelengths of about 100 pixels. After adjusting the scanner somewhat, a sidelength of 105 is achieved. Since each square consists of a 14×14 grid, every block of data now has a sidelength of 7.5 pixels. This is a good safety margin for the accurate reading of

a block's binary value, because a few pixels can be used to find an average value while staying away from the edges. This resolution can easily be achieved in practice with a low-cost camera when either the location of the packages is well known (and the camera can either zoom in on the packages or be located close to them), or when the camera is handheld.

The data set now contains 100 distinct stamps, of which 23 have clean background, 13 have light clutter, 14 have heavy clutter, 20 are used for the varying pressure test and 30 are used for the ink degradation test. The whole data set is used for testing purposes.

6.1.4 Training set: Modelling the squares

The training set consists of artificial models of the stamps, because it must be useful for any combination of data in the matrix (not just for the prototype stamping device supplied for this project). Modelling also has the advantage in that it allows the generation of an arbitrary number of rotations and sizes of high-quality squares for a well-representative training set.

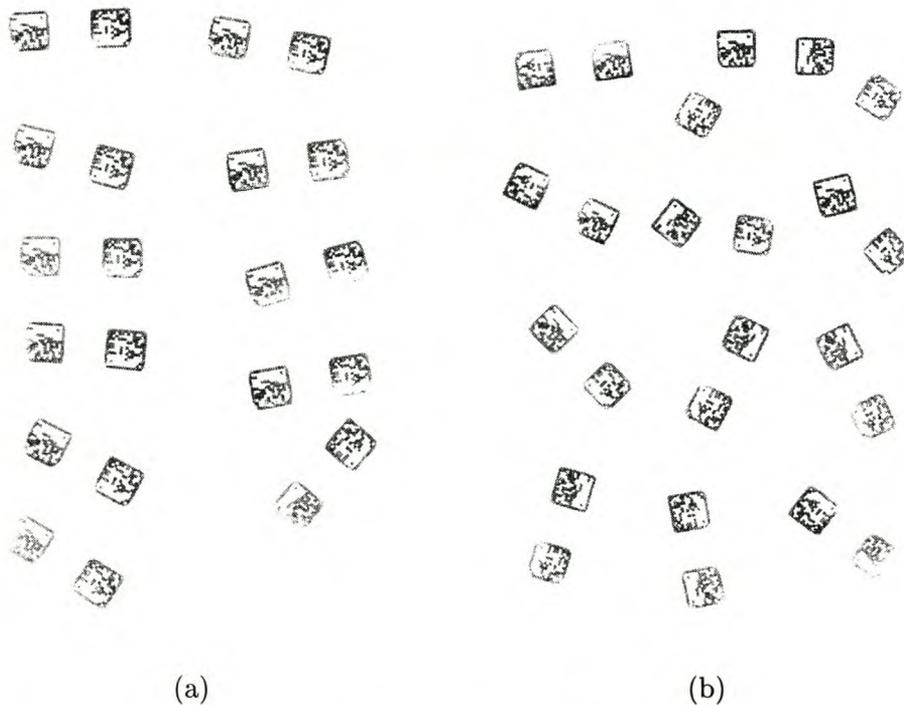


Figure 6.4 The data set part 1: Two sheets with clean background.

The model is designed to convey the following object information to the filter:

1. a square is a 16×16 matrix
2. the outer blocks of the matrix are black (forming an outline)
3. the inner blocks can be any binary matrix (black and white pattern)

The third characteristic above is trained by using two base images in the training set, one with a black-and-white checkerboard pattern and the other with the pattern reversed. Two possible distortions are included in the training set:

1. 360° Rotation: The symmetry of the two base images require that they each be rotated through 90° only. 18 rotated images are generated for each base image.

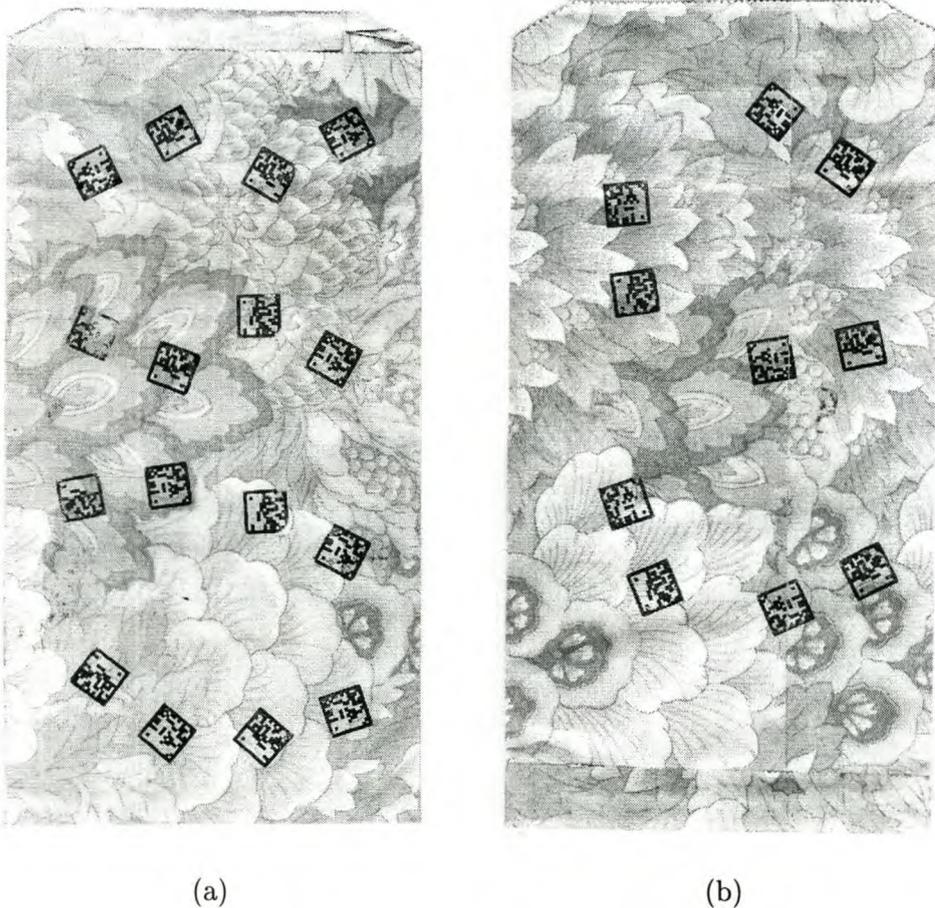


Figure 6.5 The data set part 2: Two sheets with light clutter.

2. Scale: An approximate 10% variation in scale is trained into the MINACE filter. The testing set contains squares 105 pixels wide, and the training set is made up of 6 different sizes. These sizes range linearly from 90 pixels to 110 pixels.

The final training images are created by representing the 16×16 matrix with 4 pixels per matrix cell, upsampling this 64×64 figure to the required size (between 90 and 110 pixels), and rotating the result. Note that the upsampling and rotation steps are done using bilinear interpolation, because when using bicubic interpolation the sharp edges between the matrix cells produce a ringing effect.

The final training set contains $2 \times 18 \times 6 = 216$ training images.

6.2 Preprocessing

The preprocessing consists of four main steps:

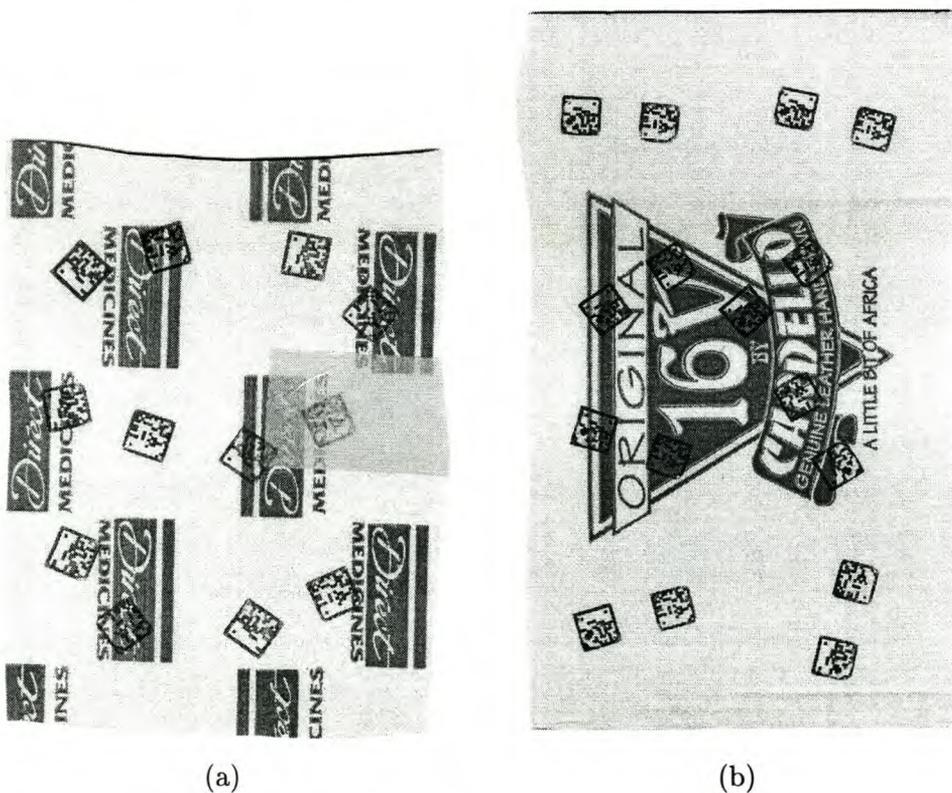


Figure 6.6 The data set part 3: Two sheets with heavy clutter.

1. Inversion and normalization

The 8-bit input images are inverted to let the ink be the higher, more significant grayscale values, and then normalized to 255.

2. Lower histogram threshold

This threshold tries to remove a possible uniform background from the image, and the threshold value is the mean grayscale value of all the pixels in the image.

3. Sine histogram transformation

This step moves most of the pixels belonging to the ink of the stamp closer to the ideal value of 255 (white), and leaves the rest of the histogram range relatively unchanged. If u is the grayscale level, the transformation is written as

$$f(u) = \sin\left(\frac{u}{255} \times \frac{\pi}{2}\right), \quad u = [0, 1, 2, \dots, 255]. \quad (6.1)$$

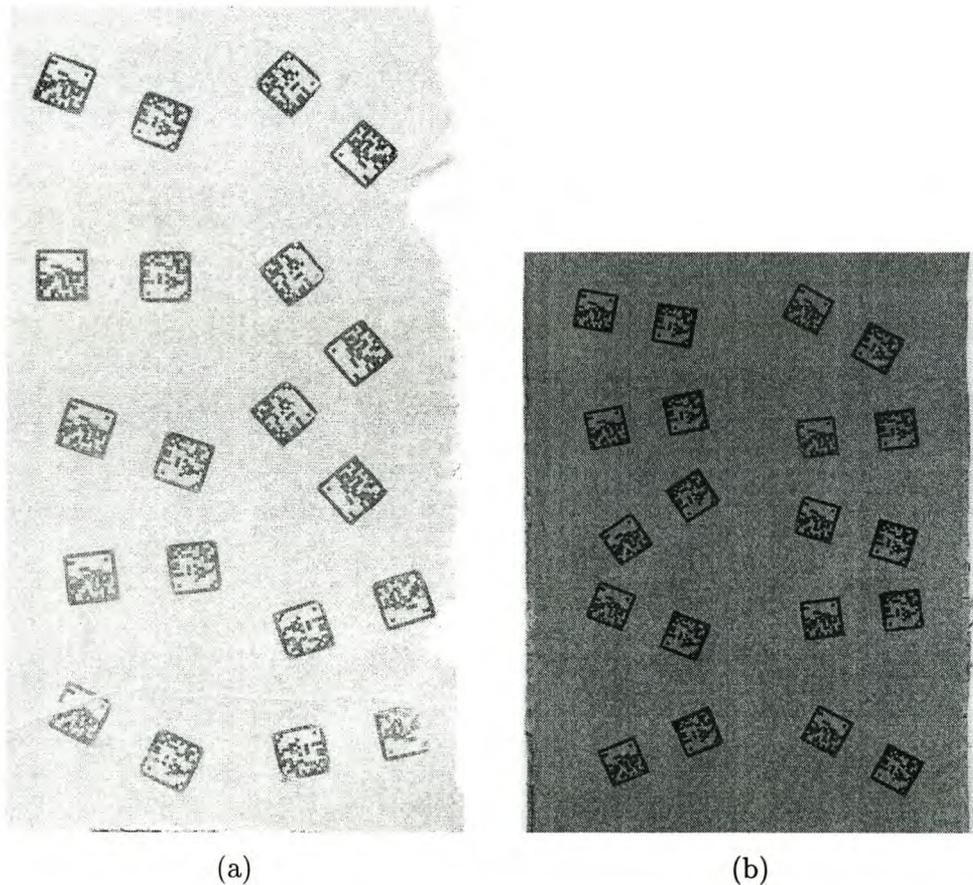


Figure 6.7 The data set part 4: Two sheets with different stamping pressure.

4. Median smoothing filter

A 3×3 median smoothing filter reduces the speckle noise caused by the physical stamping mechanism and the ink, and also preserves edge detail. The benefit of this operation is intended for the eventual reading of the data contained in the stamp.

The first step is basic preprocessing, required for the MINACE filter to function correctly, and the other three steps attempt to improve the stamps. These steps are demonstrated in Figure 6.9 on an uncluttered stamp, in Figure 6.10 on a lightly cluttered stamp and in Figure 6.11 on a stamp in heavy clutter. The stamps are improved in each case, but the worst clutter still remain. As it is, the preprocessing steps are very general and the only significant application-specific assumption is the intensity of the ink. Improving the performance in heavy clutter will require more biased approaches which may not work as readily in all scenarios.

6.3 MINACE detection filter

Now that a training set is available, the second part of implementing a MINACE filter is examined, i.e. determining the filter parameters. Most of these decisions are straightforward to set up in this application. In fact, most are already set.

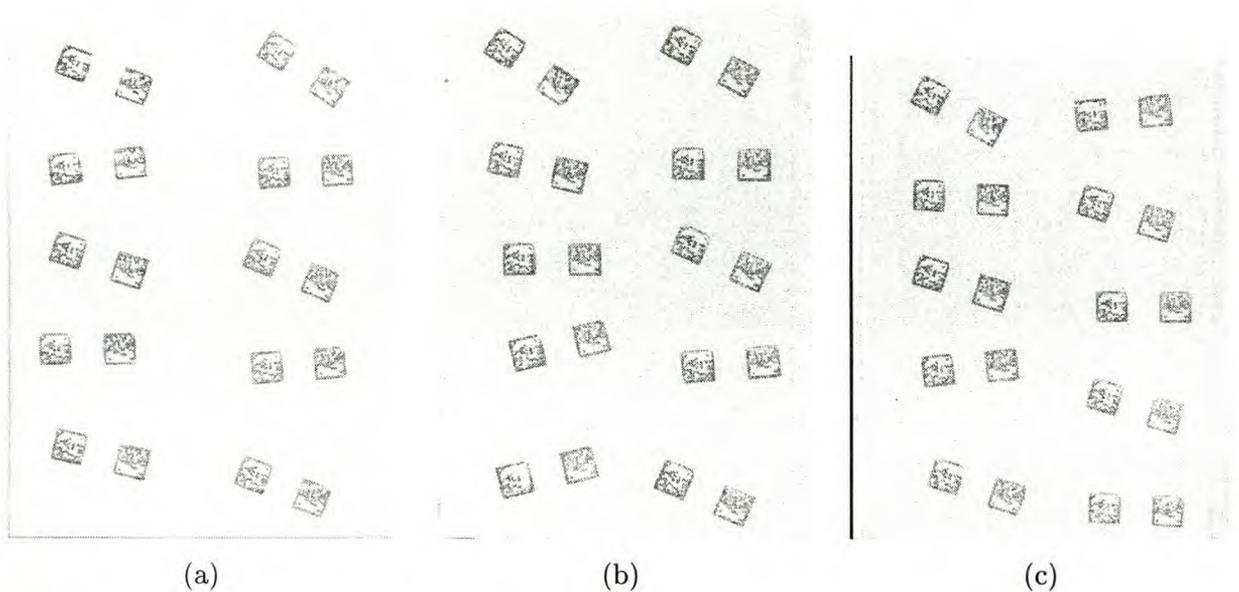
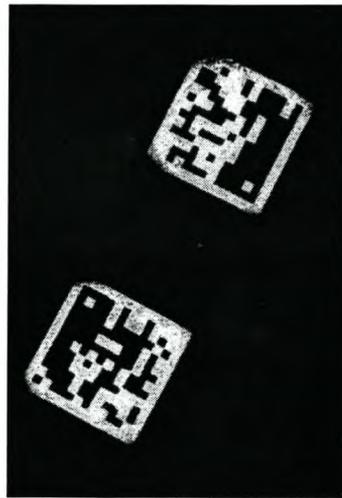


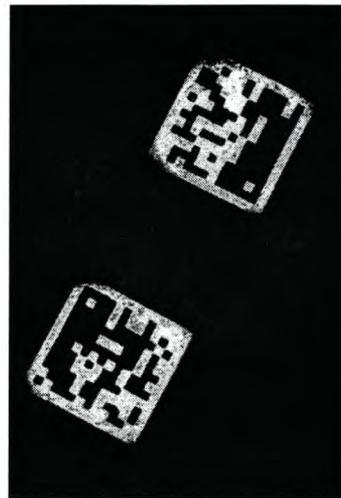
Figure 6.8 The data set part 5: Three sheets with ink degradation.

6.3.1 Parameter decisions

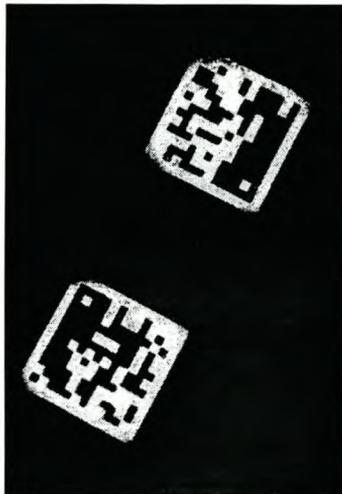
1. The filter size is 158 pixels square. This is enough to fit a square of sidelength 110 pixels, rotated 45 degrees.
2. The testing set is not used for statistics in the design of this MINACE filter. The training set model is simple and straightforward, and visual evaluation is sufficient for this stage.



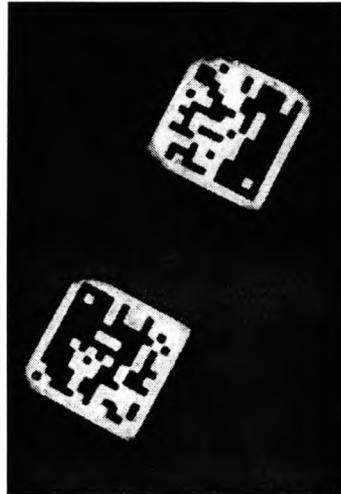
(a) after inversion and normalization



(b) after lower histogram threshold

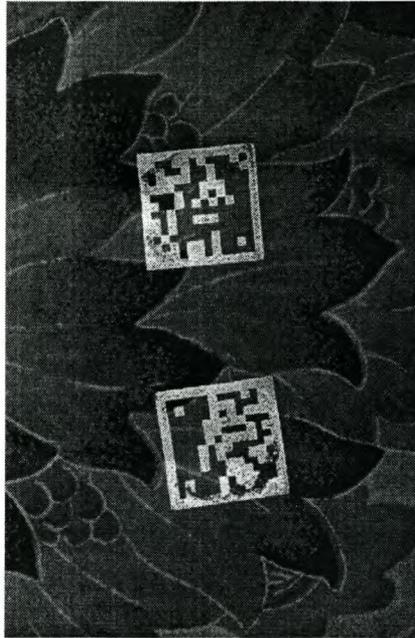


(c) after sine histogram transformation



(d) after median smoothing filter

Figure 6.9 Preprocessing stages on an uncluttered image.



(a) after inversion and normalization



(b) after lower histogram threshold



(c) after sine histogram transformation



(d) after median smoothing filter

Figure 6.10 Preprocessing stages on a lightly cluttered image.

3. The standard training threshold value of 0.8 is used again.
4. c_{em} is the only parameter not already determined. Its value should balance the two minimization properties of the MINACE filter. A good c_{em} -value is experimentally determined to be around 0.02. The rationale behind this value is demonstrated next.



(a) after inversion and normalization



(b) after lower histogram threshold



(c) after sine histogram transformation



(d) after median smoothing filter

Figure 6.11 Preprocessing stages on a heavily cluttered image.

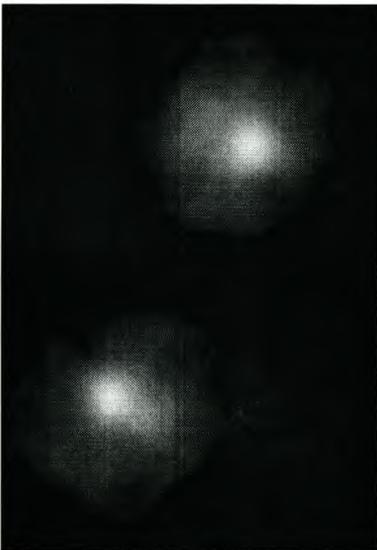
6.3.2 Determining c_{em}

The stamp image shown in Figure 6.9 is used to determine c_{em} . It is not important to consider clutter for this; the desired c_{em} -value is the one which subjectively gives the best type of MINACE filter emphasis on the stamps. In other words, the filter should generalize well enough to minimize the sensitive behaviour of low c_{em} -values, but should not be too general and give large peaks. Experience with the diamond detection application indicates that acceptable c_{em} -values will use roughly around 10% of the total training set.

The effect of extreme c_{em} -values on the stamps is examined before choosing the final value. Table 6.1 summarizes the results of this procedure, and the filtered test images are shown in Figure 6.12. Again, for comparison, histogram clipping is used, displaying the grayscale range between 0 and 1 in these filter outputs.

Table 6.1 Filter observations for energy minimization weight investigation.

	c_{em}	true-class training set size	condition number
(a)	1	6	1.9×10^1
(b)	10^{-4}	53	1.0×10^2
(c)	2×10^{-2}	14	2.6×10^1



(a) $c_{em} = 1$



(b) $c_{em} = 10^{-4}$



(c) $c_{em} = 2 \times 10^{-2}$

Figure 6.12 Filter outputs for energy minimization weight investigation.

The peaks for detected objects tend to be wide, and a smoother peak is preferred to a sensitive output with many local maxima. This is a difficult application for training size-invariance, and the wide peaks are caused by the 10% variation in the size of the squares included in the training set. The c_{em} -value of 0.02 is determined to be a well-balanced choice. The filter is shown in Figure 6.13.

6.3.3 Performance in clutter

The lightly and heavily cluttered images from Figure 6.10 and Figure 6.11 are used to examine the chosen filter further. Figure 6.14 shows the MINACE filter performance in light clutter with varying levels of preprocessing. Basic preprocessing is only the first step of inversion and normalization, as detailed in Section 6.2. Similarly, Figure 6.15 investigates the scenario of heavy clutter.

The filter deals well with a uniform level of fine clutter, but poorly with abrupt, straight-edged clutter. This is quite understandable, because the main features of these squares which are useful for the MINACE filter are the four edges and the distances between them. Also, full preprocessing generally gives better peaks for the stamps, because the stamps are closer to the ideal. Although the edge effects in the filter's output are normally ignored, in these examples, and especially with the basic preprocessing, they illustrate the filter's sensitivity to straight edges. The last observation is on the bottom square in Figure 6.15, which is one of the squares in the whole testing set with the lowest filter peaks. Although the square itself is fairly clear, the nearby clutter shapes have a highly detrimental effect.

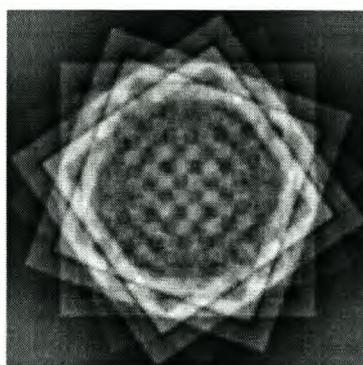


Figure 6.13 Normalized space-domain representation of the chosen MINACE filter, trained with some size variation and a c_{em} -value of 0.02.

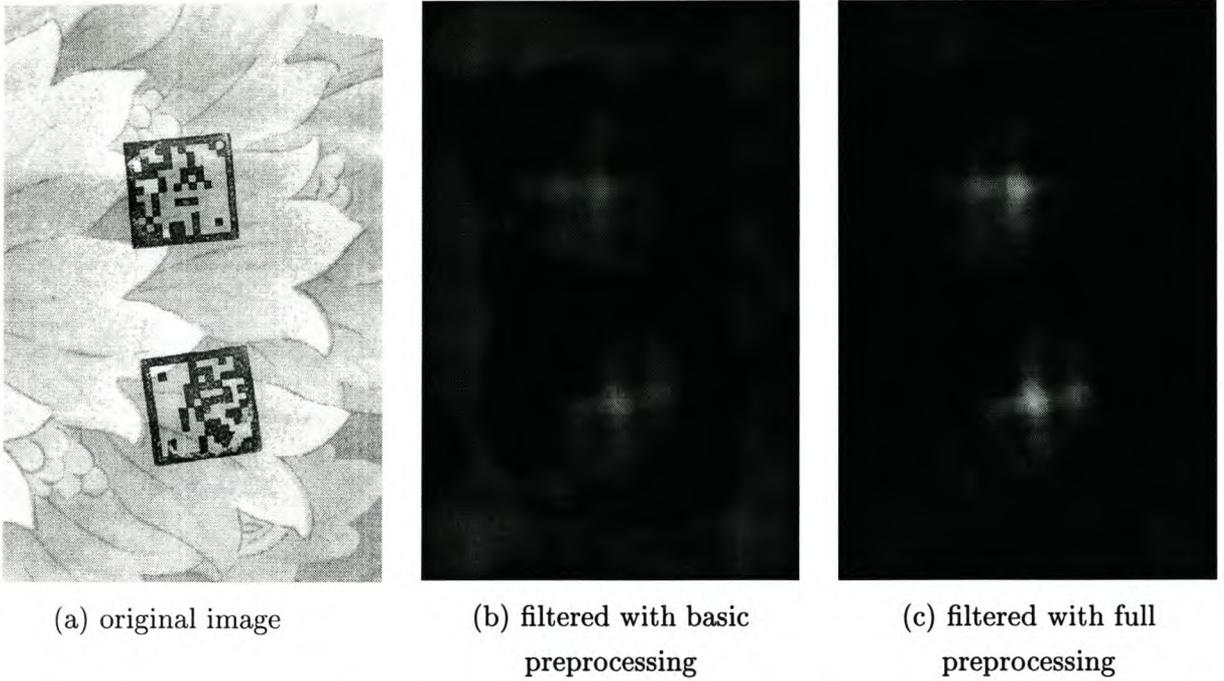


Figure 6.14 Applying the chosen MINACE filter in light clutter, with varying amounts of preprocessing.

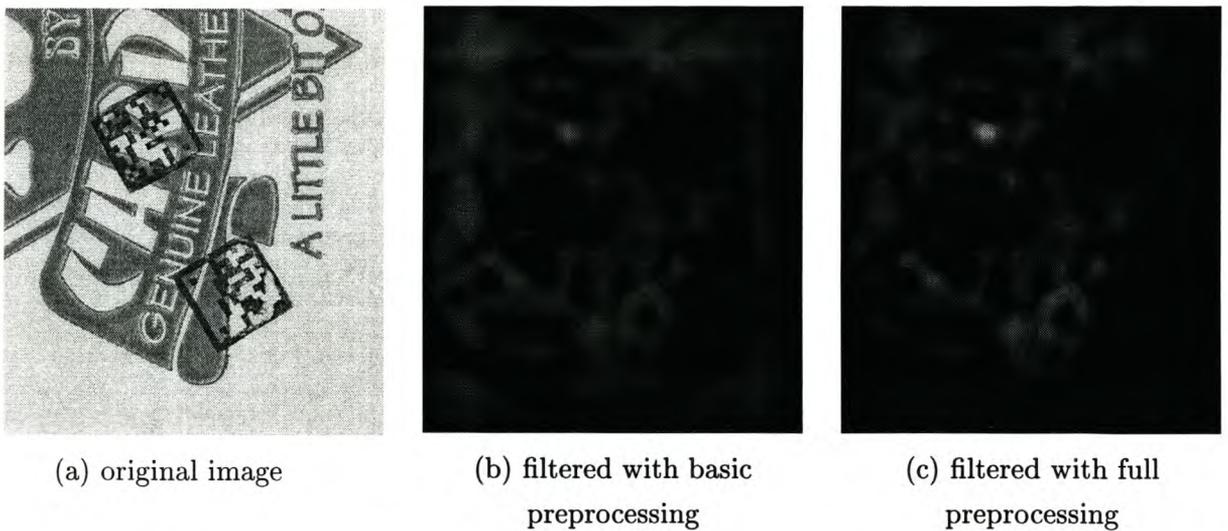


Figure 6.15 Applying the chosen MINACE filter in heavy clutter, with varying amounts of preprocessing.

6.3.4 Training for size variations

The choice of size variation discussed in Section 6.1.4 is done in order to relax the criteria for placement of the camera. The effect of size variation is investigated here, and the discussion uses the most difficult example of a stamp so far (the one with heavy clutter used in Figure 6.15).

A MINACE filter trained with no square size variation (i.e. trained with the exact size of the squares) is demonstrated in Figure 6.16a. This filter is trained with the final c_{em} -value of 0.02, and a filter size of 150 (the smallest size necessary). The most noticeable difference in the filtered image is the smaller size of all the peaks, therefore giving better location accuracy.

A filter trained with a higher level of square size variation produces an output image as shown in Figure 6.16b. In this case, the square size in the training set varies linearly between 80 and 120. A filter size of 172 is needed to fit a square, 120 pixels wide, diagonally. It is clear from the output that this filter is not a good choice. Apparently, as the filter's object size variance increases, it becomes more unsure of where to place the peak. A corner is a very significant feature in this application, because every training image contains four corners of a square. Also, the training process shows that the training images of the two extreme square sizes tend to be dominant. The structure of four peaks, which is visible on the squares in the filter output, is formed because the size of the squares in the image is in between the two extreme training sizes. Therefore, there are four places inside the square where the filter emphasizes both the nearby corner (the smallest square in the training set) as well as the furthest corner (the largest square in the training set). The only conclusion is that, in this application, the MINACE filter is not effective as a size-invariant detector.

Although the initial choice of square size variation may result in slightly poorer filtered results, it makes the filter a little less dependent on possible distancing problems and therefore more practical. As long as the two squares of the stamp are amongst the eventual ROIs, the postprocessing will be quite capable of finding them.

6.4 Segmentation

Although adjacent peaks are not expected to be a big problem in this case study, the watershed-based segmentation technique is compared to the faster global thresholding. The image used for the comparison is the output image of the chosen MINACE filter in heavy clutter, shown in Figure 6.15. The initial stages of the watershed technique is shown in Figure 6.17. The preparatory steps are histogram inversion and smoothing (7×7 mean filter). The smoothing is to prevent the watershed algorithm from being too sensitive to noisy parts of the filtered output.

Figure 6.18 compares the output of the two segmentation techniques for a threshold value of 0.3. The low threshold value is necessary to detect both stamps. Note that the filtered image is not smoothed before global thresholding, because the absolute peak values are more important and region formation is less sensitive to noise than with the watershed technique.

Although the two squares in a stamp are not close enough to necessitate awareness of region adjacency in the segmentation stage, heavy clutter introduces other peaks which makes the situation less certain. The region clustering effect of global thresholding can be seen on the lower square in Figure 6.18a. It is fortunate that the highest peak in that thresholded region is the correct one. Further comparisons will be made in Section 6.6 when the whole testing set is evaluated.

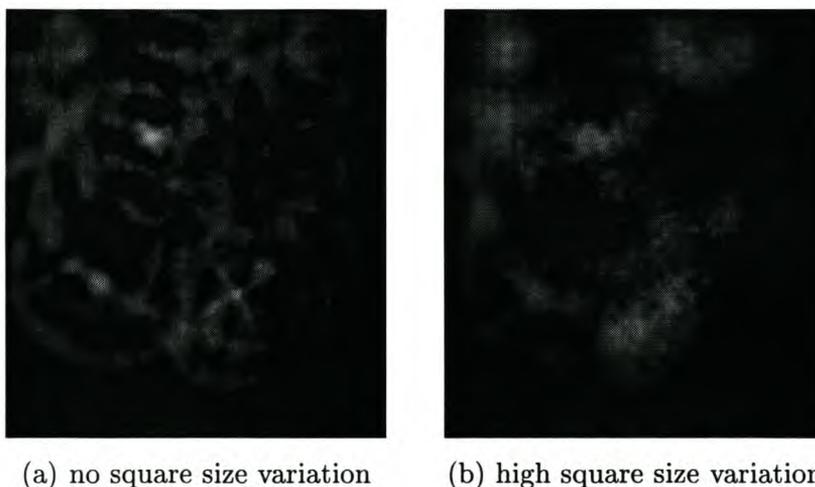


Figure 6.16 Performance of MINACE filter trained with different levels of square size variation.

6.5 Parameter extraction and ROI verification stage

This stage uses a number of specific techniques to identify regions containing squares. The techniques exploit the natural characteristics of a square, i.e. the four, 90° apart, rising edges surrounding its center (as seen when examining the grayscale intensity at the outside of the outline cells of a preprocessed square). The first step is to determine and refine basic

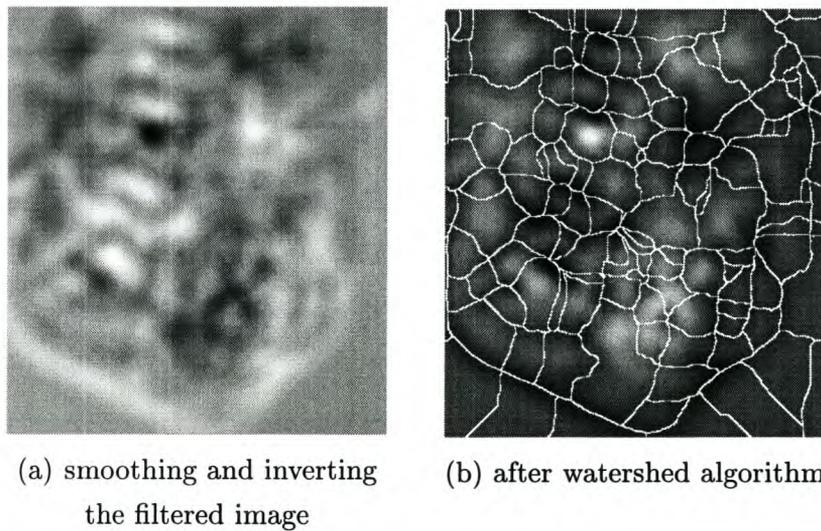


Figure 6.17 Initial stages of watershed-based segmentation. The MINACE filter output is of a stamp in heavy clutter.

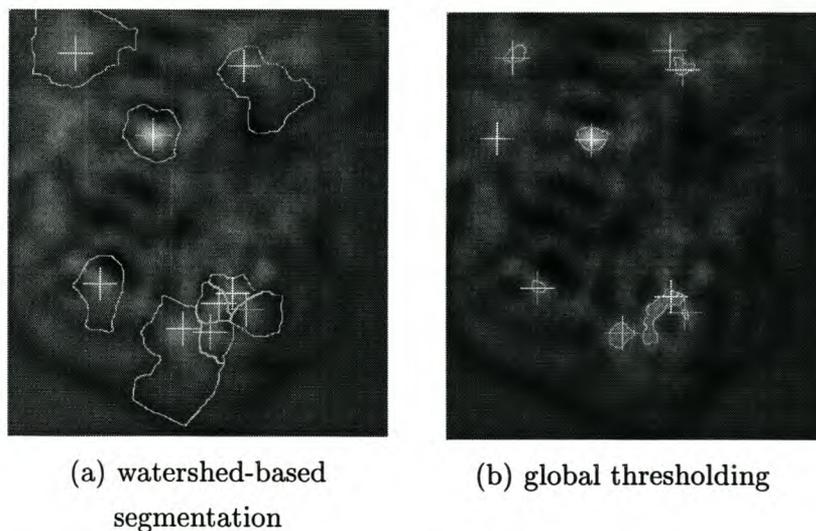


Figure 6.18 Comparison of the watershed-based segmentation technique and global thresholding, for a threshold value of 0.3.

square parameters (side distances and rotation) within each ROI. A squareness estimate is used to narrow the range of eligible regions, and the remaining ROIs are searched to find two similarly sized squares separated by the desired distance. Finally, if a stamp is found, the data in the squares are read.

The technique used to search for the edges rely strongly on the principles of the Hough transform [32]. A Hough transform takes the shape of an angle vs. radius matrix, where the matrix cells represent the intensity average calculated over the corresponding line in the input image. This line averaging concept is illustrated in Figure 6.19. A variation of the Hough transform is used to search for edges. This variation is a differential search, and can tell the difference between a rising edge and a falling edge (since prominent falling edges can be found just on the inside of a square's border, depending on the data inside the square). This search is a function of angle and radius, and can be plotted as a matrix (or image) when a regular step size is used. These Hough outputs are displayed as grayscale intensities.

The rotation of each of the 2 squares in the stamp is easier to extract than it is to read the 2 datamatrices. Reading a datamatrix requires that the square's rotation, center coordinate and side distances be known very accurately. This stage is designed to handle this, and also to be a robust enough measurement to deal with the severe clutter of the data set created in Section 6.1.3. The design of this stage should not require alterations

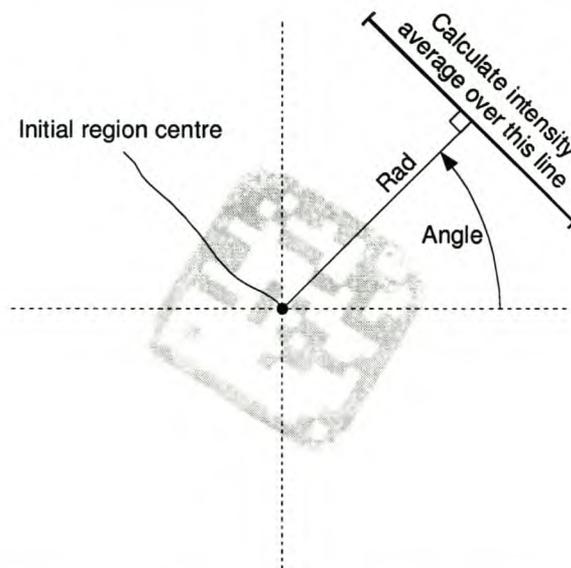


Figure 6.19 The perpendicular line averaging concept, central to the Hough transform.

for any reasonable application that uses this 2D code layout. The parameter extraction and ROI verification stage is as follows:

1. Requirements

- the preprocessed source image
- the output of the MINACE and smoothing filter
- the output of the segmentation algorithm

2. Initialize the angular and radial resolution parameters

These parameters determine the resolution with which the differential Hough matrix is calculated (see step (4a)). The resolution is experimentally determined so that square edges can be detected reliably. The parameters are:

- $Angsteps = 120$
The number of angular steps in 360° .
- $Radmax = 67, Radmin = 37$
The radial distances where the search for edges in steps of one pixel begin and end. MINACE peaks are well centered, and the filter is trained for squares where the side distance should be around 52. These two parameter values therefore provide an adequate margin of error.
- $LineAvLength = 100$
This is the length of the line, perpendicular to the vector described by the angle and radius from the region center, over which an intensity average is calculated.

The first two parameters are used only for the initial Hough edge search.

3. Locate center of each ROI

The center of a ROI is simply the pixel inside with the highest value in the MINACE filter output.

4. Determine and refine squareness measurements for each ROI

For each ROI, do:

(a) Calculate the differential Hough matrix

This differential search is done by taking the difference between two nearby perpendicular line averages (subtracting the furthest one from the closest one).

A 3×3 smoothing filter is applied to the output matrix of the Hough search prior to use. The corresponding differential Hough matrix for the stamp in the background of Figure 6.19 is shown in Figure 6.20. The angular resolution is 3° , and the radial resolution is one pixel.

(b) **Find rough ROI rotation**

This step uses a stamp's squareness to do a robust search for the rotation. The assumption is that a square's differential Hough matrix will have four 90° -separated directions (columns) containing very high scores in the search for rising edges. Figure 6.20 shows clearly that this assumption has merit. Note the high negative peaks on the inside of the square's border, where the falling edges tend to be. If a differential search is not used, this would be a problem. To extract the outside edges automatically, the differential Hough matrix is reduced to a vector by taking the maximum over every angular direction (column). If the ROI is a square, this vector contains exactly four periods of a periodic signal. The offset of the peak in the signal is extracted by dividing the vector into four equal parts and adding them together. This final detection vector of the matrix from Figure 6.20 is shown in Figure 6.21. The index containing the maximum value in this final vector indicates the rotation of the square. This rotation value is only as accurate as the rotational resolution chosen in step (2), but a greater resolution will increase the calculation time.

(c) **Estimate the four side distances using the differential Hough matrix**

The estimated rotation from the previous step indicates the four columns, or angles, in the differential Hough matrix containing the square's edges. These four columns are now searched from the furthest radius index to the closest,

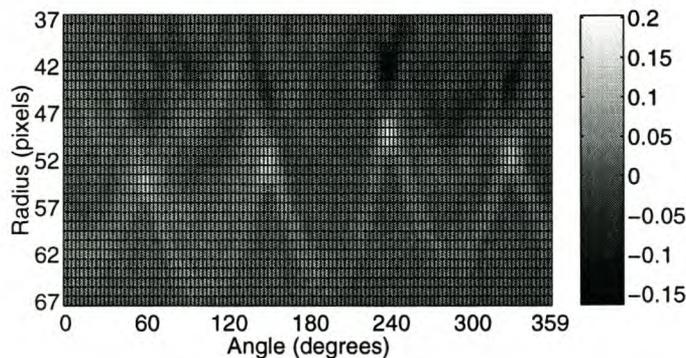


Figure 6.20 Example of a differential Hough matrix, with polar co-ordinates from the center of the ROI.

for the largest peak value.

(d) **Refining ROI rotation estimate**

Another differential Hough matrix is calculated, this time for the radial and angular region around the highest peak (best detected side distance) in the previous Hough matrix. This matrix covers a radial distance of six pixels in steps of 0.1, and an angular distance of 7.2° in steps of 0.1° . The angular distance includes a 20% overlap with regards to the angular resolution of the first Hough matrix. The highest peak in this new matrix, after applying a 3×3 smoothing filter, improves both the rotation estimate and the side distance estimate of this side. The square's Hough matrix for this step is shown in Figure 6.22.

(e) **Refine all edge distances**

Using the final rotation estimate from the previous step, three radial differential Hough curves are calculated to refine the remaining side distances. The

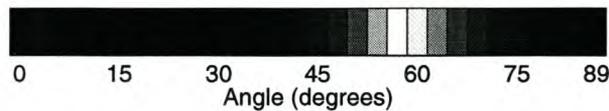


Figure 6.21 The search vector calculated from the differential Hough matrix, used to search for the base ROI rotation ($0^\circ - 90^\circ$).

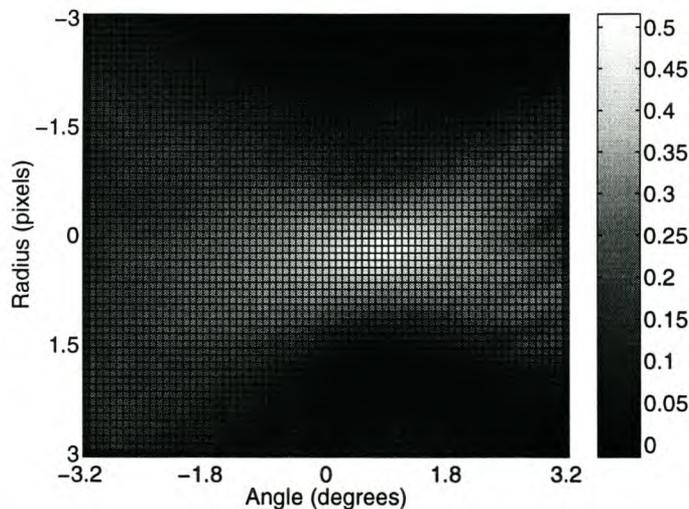


Figure 6.22 Differential Hough matrix for rotation refinement of the example square, relative to the polar co-ordinate of the best detected side.

searching distance is 10 pixels in both directions from the estimate in steps of 0.1, enough to cover more than the width of a datamatrix cell. The highest peak in each of the curves is used for the new estimate. The curves for this step in the detection of the example square is shown in Figure 6.23. All four curves are shown, for comparison with the already refined side distance from the previous step.

5. Narrowing down the regions

Regions are eliminated according to the following criteria:

(a) Four edges

A square must have four edges, and the four final differential Hough values at the detected side distances are inspected. If any edge has a value below 0.05 (a good edge generally has a value around 0.5), the region is discarded. This threshold is important for detection sensitivity. Although some poorly stamped squares may be discarded with this choice of threshold, clutter resistance is improved.

(b) Squareness

Squareness is measured by the following equation:

$$\text{squareness} = 1 - \frac{|\text{length}_{hor} - \text{length}_{ver}|}{\text{length}_{hor} + \text{length}_{ver}}. \quad (6.2)$$

If a region's squareness is below 0.97, it is discarded.

(c) Duplicates

If the centers of two regions are within 60 pixels of one another, the region with

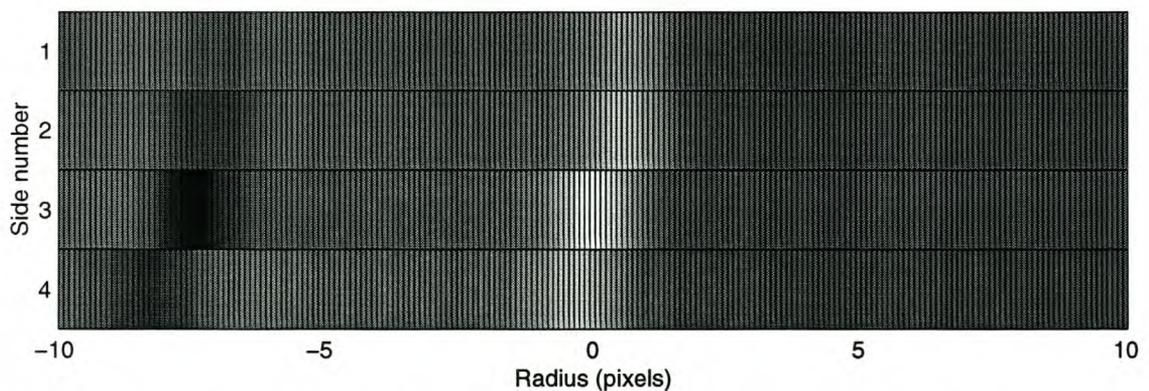


Figure 6.23 Radial differential Hough curves for side distance refinement of the example square, relative to the initial estimate.

the weakest edges are discarded. This edge-ratio is calculated for each region as the product of the four final differential Hough values at the detected side distances.

6. Processing region pairs

For every combination of two of the surviving regions (regions i and j), the following are calculated:

(a) Sidelength ratio

The average sidelength A is measured for every region, and the sidelength ratio is calculated as $1 - \frac{|A_i - A_j|}{A_i + A_j}$. This must be greater than 0.97 to continue the final criteria.

(b) Distance ratio

The distance D_{ij} between two regions should be twice the true sidelength of the squares, and the distance ratio is therefore calculated as $1 - \frac{|A_i + A_j - D_{ij}|}{A_i + A_j + D_{ij}}$. This must be greater than 0.97 to continue.

(c) Final choice

If the algorithm manages to reach this stage, the two regions are very likely not a false alarm. If this point is reached more than once though, the region pair with the best distance ratio is taken as the final choice for the barcode stamp.

If a stamp is found in the input image, the data matrices are read. This is done using the two center co-ordinates, the two rotation values, the sidelength values of the squares, and trigonometry. Each cell in the datamatrix is read as the average value of a 3×3 , $+$ -shaped grouping of pixels at what is calculated as the center of the cell. Afterwards, the raw grayscale output is thresholded to achieve a binary result. This threshold value is determined using the K-means clustering algorithm [33] on the sample space of a data-matrix's grayscale values.

The K-means clustering algorithm clusters the $N = 144$ data points x_n into $K = 2$ disjoint subsets S_k , in order to minimize the sum-of-squares error function

$$J = \sum_{k=1}^K \sum_{n \in S_k} |x_n - \mu_k|^2, \quad (6.3)$$

where μ_k is the centroid of the data points in S_k . The algorithm only achieves an estimate of the minimum, but is useful for its simplicity. The implementation of the algorithm proceeds as follows:

1. Initialize μ_k to 0.5 and 1, respectively

2. repeat:

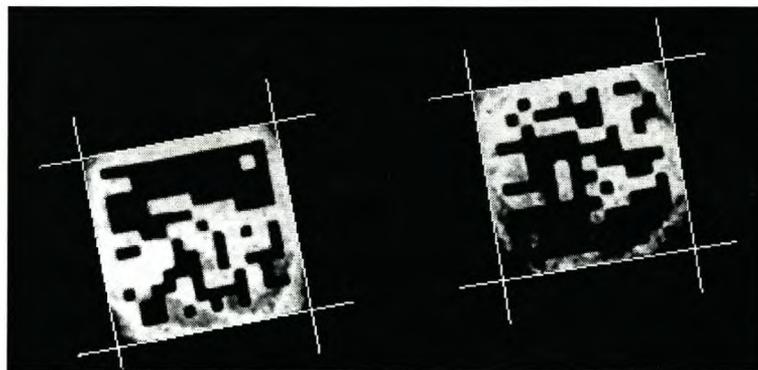
(a) Divide data points into subsets S_k

(b) Calculate new centroids μ_k

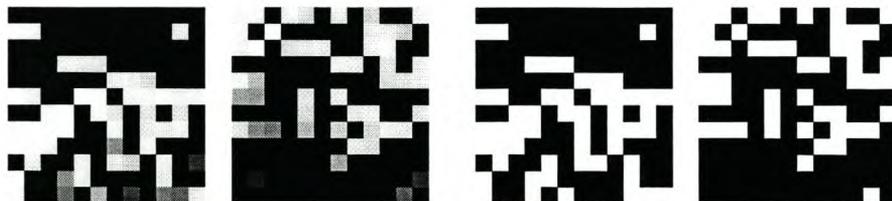
until total difference between new and old centroids is less than 0.001

The initialization of $\mu_0 = 0.5$ is a better estimate of a cluttered background than $\mu_0 = 0$. If this initialization causes one of the subsets to contain zero data points anywhere in the algorithm, it is restarted with the initialization values for μ_k as the minimum and maximum of the data points. When the algorithm is finished, the value between the two centroids is used to binarize the grayscale square.

Figure 6.24 demonstrates the data reading routine. The poor quality of the stamp in this figure results in two misread bits in both datamatrices, even though the background is perfectly clear. The bit error rates are more thoroughly investigated in Section 6.6 on the full testing set.



(a) detected stamp



(b) raw grayscale output

(c) binarized datamatrices

Figure 6.24 Reading the datamatrices from a sample stamp.

If the implementation of the project is approved, the rest of the design detail can be finalized. This include: data size (and final size of the datamatrices), the encoder/decoder to use (any of the commercial ones), final handling specifications for the stamping device (true expected clutter), unforeseen limitations of the digital camera, etc. Only then can one proceed with the data extraction and decoding steps.

A full evaluation of the performance and bit error rates of the detection and parameter estimation algorithms on the various testing sets is discussed next.

6.6 Detector evaluation

The detector is complex, and this evaluation will focus more on the achievable end result than on all the possible variations and the effect of the trade-offs. The best detection results are achieved when watershed-based thresholding at a peak value of 0.3 is used on the filtered output. The testing set images are used as they are, for testing efficiency, but this weakens the effect of the histogram adjustments in the preprocessing slightly. The evaluation starts by discussing the squares that are not detected by this algorithm.

Figure 6.25 shows a number of squares that, after passing any of the thresholding stages, are rejected by the verification stage. These squares all have one poor edge, and some of them only narrowly misses the detection criteria. Trying to make more of these pass the detection criteria weakens the detector against clutter, and the data reading stage is more likely to have bit errors even with the better of these examples.

When global thresholding is used on the filtered output instead of watershed-based thresholding, at the same value of 0.3, three more squares are not detected. These squares are all from the heavily cluttered images, and are shown next to their filter outputs in Figure 6.26. All three squares are victim of the region clustering effect of global thresholding. Global thresholding can avoid this by using a higher threshold value, but some squares

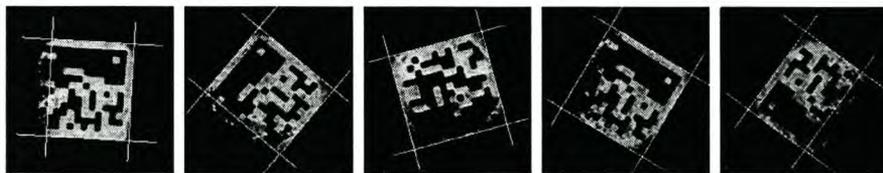


Figure 6.25 Squares with poor edges are rejected by the verification stage.

have very low MINACE peaks and will then be lost instead.

Finally, the information represented by the squares are read. Compiling statistics on the accuracy of the rotational variable is only of limited use. If the difference in rotation between the two squares of each of the 30 stamps in the three source images testing ink degradation (where the square rotation is not changed) is examined, the numbers show a slight variation. The mean value of this relative rotation is unimportant, because it only indicates the human error that occurred in aligning the squares manually. The standard deviation, however, is about 0.55° , while the finest resolution used in the measuring stage is 0.1° . Although it is not trivial to isolate the various factors that contribute to this variation, it is much less than the 30° between two rotational positions, and therefore poses no problem. The process of reading the data from the squares is much more delicate.

The bit error rates for the five subsets of the testing set are compared in Table 6.2. This is for the case of watershed-based detection and a threshold of 0.3. Some surprises show up in this table. There seems to be very little difference in the bit error rates between carton, clean and lightly cluttered backgrounds. Ink degradation clearly does affect the bit error rate, while a variation in stamping pressure does not. Serious clutter, of course, results in stamps which are unreadable with the simple binarization technique employed here. The high maximum bit error rate for row 1 in the lightly cluttered subset is simply from the left square shown in Figure 6.3b. Figure 6.27 shows some of the squares with bit error rates between 10 and 15.

The execution speed of the detector is investigated in Table 6.3, where the time taken for

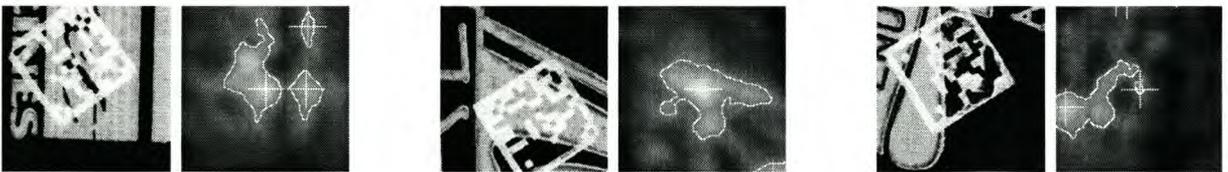


Figure 6.26 Some squares in heavy clutter are not detected by global thresholding.

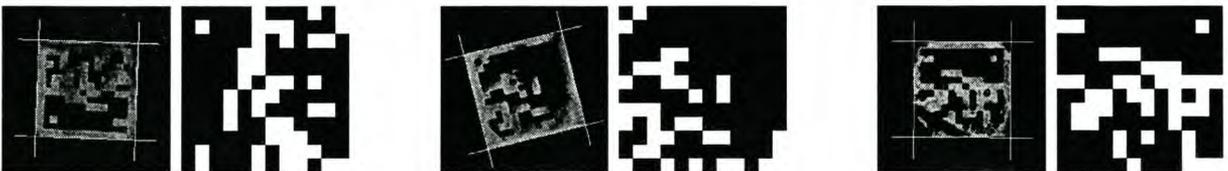


Figure 6.27 Some squares which have between 10 and 15 bit errors.

Table 6.2 Bit error rates for the testing set images.

subset identifier	image reference	no. of detected squares	total bit errors	maximum square bit errors	median bit errors per square
clean	Figure 6.4a	14	49	11	2
	Figure 6.4b	22	57	8	2.5
light clutter	Figure 6.5a	16	47	28	1
	Figure 6.5b	10	24	5	2.5
heavy clutter	Figure 6.6a	12	274	65	12
	Figure 6.6b	16	397	69	5.5
stamping pressure	Figure 6.7a	18	25	9	0
	Figure 6.7b	20	22	6	0
degradation	Figure 6.8a	20	70	12	2
	Figure 6.8b	20	86	15	4
	Figure 6.8c	20	89	10	4

the various stages are shown for both segmentation techniques. The image used for this investigation is a 800×800 cutout of the center of the heavily cluttered testing image in Figure 6.6b. Accepting the loss of one square, the threshold value is chosen as 0.5 to allow both techniques to detect three of the four stamps. If the threshold value is 0.3, the watershed technique detects all the squares with a total execution time of 20.5 seconds. That increase in time is entirely due to the postprocessing and the increased number of regions (from 11 to 43).

Table 6.3 Comparing the execution times of the various stages of the detector when using either global thresholding or watershed-based segmentation on a 800×800 heavily cluttered image, using a threshold value of 0.5 (refer to Appendix A for implementation detail).

detection stage (global thresholding)	execution time (seconds)	detection stage (watershed technique)
preprocessing	1.8	preprocessing
MINACE filter	4.3	MINACE and smoothing filter
global thresholding	0.6	watershed technique
postprocessing (13 regions)	4.2	postprocessing (11 regions)
total	10.9	total

6.7 Conclusion

The prototype stamped code reader is successfully completed, and performs well even under poor conditions.

In this case study, the object detection approach is augmented by a customized parameter extraction algorithm to process the ROIs. This algorithm exploits the known regular features of the stamps to locate and describe each square with subpixel accuracy.

As soon as a specification for an application using these stamps is finalized, a decoder can be added which processes the data correctly. The various executional blocks of the detector can then be adjusted to suit the needs of the specific application. These needs can include anything such as required resolution, execution speed and expected clutter.

Chapter 7

Conclusion

The object detection approach of this thesis is shown to be effective, especially when combined with the proper postprocessing. The approach, in summary, is to combine the robust MINACE correlation filter with a suitable segmentation algorithm, in order to apply customized postprocessing (even time-consuming classification techniques) on a selected set of ROIs.

The MINACE filter has numerous design parameters, and although it is important to make good choices, none of them are found to be too difficult. Even the energy minimization weight can be determined experimentally if the rest of the design is sound. It is important to have a carefully designed training set which does not expect too much from the filter. Whether both rotation-invariance and size-invariance can be achieved, depends on the objects.

The techniques considered for the segmentation of correlation outputs are global and adaptive thresholding, and watershed-based thresholding. Compared to the speed of global thresholding and the flexibility and accuracy of watershed-based thresholding, the variants of adaptive thresholding are found to be unsuited for this purpose. Using watershed-based thresholding in highly cluttered environments can benefit either detection rate or false alarm numbers, depending on how the correlation outputs of the objects and the clutter tend to cluster.

The first case study of diamond detection is challenging, since uncut diamonds are feature-poor and their environment is highly cluttered. The proposed approach ends up with a significant number of false alarms, even after regional postprocessing and using watershed-based thresholding with full cluster elimination. Nevertheless, the MINACE filter can be

used on its own as an effective enhancement filter to indicate potential hotspots for a trained operator.

In the second case study, the problem of reading a dynamic 2D barcode stamp amounts to the detection and parameter extraction of two stamped squares, the components of a stamp. The ROI-output of the proposed approach is processed with a custom detection routine, based on the Hough transform. The MINACE filter performs well, although observations show that the filter is not capable of being a size-invariant detector for the stamped squares. Within that constraint, the system is capable of detecting squares within clutter accurately. Using watershed-based thresholding as opposed to global thresholding can achieve a slightly better detection rate in heavy clutter. The level of bit errors achieved in no clutter and in light clutter requires error correction, but is acceptable for implementation.

This approach is recommended for serious consideration in a wide possible range of industrial object detection problems.

Bibliography

- [1] C. F. Hester and D. Casasent, “**Multivariant technique for multiclass pattern recognition,**” *Applied Optics*, vol. 19, no. 11, pp. 1758–1761, 1980.
- [2] Gopalan Ravichandran and David Casasent, “**Minimum noise and correlation energy optical correlation filter,**” *Applied Optics*, vol. 31, no. 11, pp. 1823–1833, 1992.
- [3] David Casasent and Rajesh Shenoy, “**Detection and classification in SAR using MINACE correlation filters,**” *Proceedings of SPIE*, vol. 2487, pp. 211–224, 1995.
- [4] Simon Haykin, **Neural Networks: A Comprehensive Foundation.** Prentice Hall, second ed., 1999.
- [5] Manuel G. Penedro, María J. Carreira, Antonio Mosquera and Diego Cabello, “**Computer-aided diagnosis: A neural-network-based approach to lung nodule detection,**” *IEEE Transactions on Medical Imaging*, vol. 17, no. 6, pp. 872–880, 1998.
- [6] J. Daugman, “**Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters,**” *Journal of the Optical Society of America A*, vol. 2, no. 7, pp. 1160–1169, 1985.
- [7] David P. Casasent, “**Gabor wavelet filters and fusion for distortion-invariant multi-class object detection,**” *Proceedings of SPIE*, vol. 2491, pp. 430–440, 1995.
- [8] David M. Weber and David P. Casasent, “**Quadratic gabor correlation filters for object detection,**” *Proceedings of SPIE*, vol. 3078, pp. 708–719, 1997.
- [9] A. Iyer and D. Casasent, “**New basis function distortion invariant detection filters,**” *Photonics for Processors, Neural Networks, and Memories II*, vol. 2297, pp. 4–9, 1994.

-
- [10] B. V. K. Vijaya Kumar, “**Tutorial survey of composite filter designs for optical correlators**,” *Applied Optics*, vol. 31, no. 23, pp. 4773–4798, 1992.
- [11] Gilbert Strang, **Introduction to Linear Algebra**. Wellesley-Cambridge Press, 1993.
- [12] J. D. Brasher, C. F. Hester and D. W. Lawson, “**Multi-stage, higher-order SDF filters**,” *Proceedings of SPIE*, vol. 1297, pp. 103–109, 1990.
- [13] Simon Haykin, **Adaptive Filter Theory**. Prentice Hall, third ed., 1996.
- [14] Neil Muller, **Facial recognition, eigenfaces and synthetic discriminant functions**. PhD thesis, University of Stellenbosch, 2000.
- [15] B. V. K. Vijaya Kumar, “**Minimum variance synthetic discriminant functions**,” *Journal of the Optical Society of America A*, vol. 3, no. 10, pp. 1579–1584, 1986.
- [16] A. Mahalanobis, B.V.K. Vijaya Kumar, and D. Casasent, “**Minimum average correlation energy filter**,” *Applied Optics*, vol. 26, no. 17, pp. 3633–3640, 1986.
- [17] Gregory House and David Casasent, “**Multi-class 3D distortion-invariant object detection in clutter**,” *Proceedings of SPIE*, vol. 2237, pp. 92–101, 1994.
- [18] David Casasent and Rajesh Shenoy, “**Synthetic aperture radar detection and clutter rejection MINACE filters**,” *Pattern Recognition*, vol. 30, no. 1, pp. 151–161, 1996.
- [19] B. V. K. Vijaya Kumar, J. D. Brasher, C. F. Hester, G. Srinivasan and S. Bolapragada, “**Role of the constraint values in synthetic discriminant function filter design**,” *Proceedings of SPIE*, vol. 1959, pp. 23–31, 1993.
- [20] David Casasent, Yoon-Yim Choo, and Gregory House, “**Advanced and orthogonal MINACE filter sets: Initial detection results**,” *Proceedings of SPIE*, vol. 2237, pp. 2–13, 1994.
- [21] Gregory House and David Casasent, “**Advanced model-based distortion-invariant filters allowing peak variations**,” *Proceedings of SPIE*, vol. 2490, pp. 53–63, 1995.
- [22] Anil K. Jain, **Fundamentals of Digital Image Processing**. Prentice Hall, 1989.

- [23] Jun Ohya, Akio Shio, and Shigeru Akamatsu, “**Recognizing characters in scene images**,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 214–220, 1994.
- [24] Øivind Due Trier and Torfinn Taxt, “**Evaluation of binarization methods for document images**,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 312–315, 1995.
- [25] Torfinn Taxt, Patrick J. Flynn, and Anil K. Jain, “**Segmentation of document images**,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 12, pp. 1322–1329, 1989.
- [26] W. Niblack, **An Introduction to Digital Image Processing**. Prentice Hall, 1986.
- [27] H. Digabel and C. Lantuéjoul, “**Iterative algorithms**,” in *Proceedings of the 2nd European Symposium on the Quantitative Analysis of Microstructures in Material Science, Biology and Medicine* (J. L. Chermant, ed.), pp. 85–99, Stuttgart: Riederer Verlag, 1978.
- [28] Luc Vincent and Pierre Soille, “**Watersheds in digital spaces: an efficient algorithm based on immersion simulations**,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.
- [29] Danielle Argiro and Charlie Gage, “**Chapter 1 - Writing Programs / VIFF Format**,” in *Khoros Manual vol. 2 - Khoros Programmer’s Manual*, University of New Mexico, 1992.
<http://www.tnt.uni-hannover.de/js/soft/imgproc/khoros/khoros1/manual.html>
(Nov 20, 2002).
- [30] Athanasios Papoulis and S. Unnikrishna Pillai, **Probability, Random Variables and Stochastic Processes**. McGraw-Hill, fourth ed., 2001.
- [31] EDIFICE Bar Code Group, “**Brochure: Automatic Data Capture (for) 2D Code Applications**,”
<http://www.edifice.org/2dbrochure.pdf>
(Nov 20, 2002).
- [32] R. C. Gonzales and R. E. Woods, **Digital Image Processing**. Addison Wesley, 1992.

-
- [33] J. MacQueen, “**Some methods for classification and analysis of multivariate observations,**” *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.
- [34] John G. Proakis and Dimitris G. Manolakis, **Digital Signal Processing: Principles, Algorithms and Applications**. Prentice Hall, third ed., 1996.

Appendix A

Software Development

A.1 Vital statistics

- **Testing setup**

Redhat Linux 9 with GNU libc libraries version 2.3.2 (release 27.9), GNU C++ compiler version 3.2.2 (release 5), and Matlab with the Image Processing Toolbox (used only for MINACE filter training)

- **Additional C++ libraries**

- **VXL version 1.0-beta2-fix1** (<http://vxl.sourceforge.net/>)

A multi-platform collection of C++ libraries designed for computer vision research. CMake version 1.4.3 or higher is required (<http://www.cmake.org>). VXL is copyrighted by TargetJr Consortium (represented by GE Corporate Research and Development), but free to use. The following parts of this library is used: matrix, vector and image classes, read and write support of the PNG image format, eigenanalysis, median filtering.

- **FFTW version 2.1.3** (<http://www.fftw.org/>)

The “Fastest Fourier Transform in the West” is a comprehensive collection of fast C routines for computing the discrete Fourier transform in one or more dimensions. It is fully portable and adapts to machine architecture for performance. FFTW is free software under the GNU General Public Licence (<http://www.gnu.org/copyleft/gpl.html>). The in-place transforms of this library are used to conserve memory. Also, the image sizes in this work are not

fixed and this library must be used with the “ESTIMATE”-flag, which means that it uses a reasonable but possibly suboptimal computation plan. To compensate for this, all images are zero-padded to a size which can be exploited by the algorithm. The diamond detection case study zero-pads to a factor of 256, and the stamped code case study zero-pads to a factor of 128.

- **Preferred compiler optimization flags**

(with the exception of the FFTW library, where the default is used)

`-O3 -march=pentium2 -mmmx -ffast-math -fschedule-insns2 -funroll-loops
-fomit-frame-pointer -fsingle-precision-constant`

- **Machine used to benchmark execution times**

300 MHz Pentium 2

A.2 Overview of new software

The code generated for the two case studies amounts to more than 4000 lines, and revolves around a central, functional library of utility routines. The library consists of a number of main components:

- **Co-ordinate classes**

A rectangular and a polar co-ordinate class pair is written for the i-j axis with a basic inter-class arithmetic capability. Very useful for any trigonometry, and developed originally for the Hough transformations in the stamped code reader.

- **File handling routines**

This set of mostly template routines include a lot of interfacing with the VXL library, to bridge the VXL subdivisions of image file handling and matrix class. These routines include read and write support for PNG files and space-delimited text files (storage of floating point filters).

- **Basic image manipulation**

This set of routines include histogram utilities such as normalization and clipping.

- **Image filtering**

Support is included for filtering an image with either one or two filters in the Fourier

domain, using the FFTW library and including all the necessary software for handling zero-padding and edge-effects. Wrappers are included to support a basic smoothing filter, and an acceptably fast 3×3 median filter is hardcoded.

- **Regional processing**

The routine for watershed algorithm is supported by a host of subroutines which handles its thresholding, histogram width adjustment, cluster elimination et al. Apart from the routines for the other styles of thresholding such as local and global thresholding, there are also a number of utilities used for region viewing purposes.

The diamond detector is assisted with a custom preprocessing function, and has utilities for generating statistics with the truth data. This project also includes the unique principle component and contour variance postprocessing routines. A separate routine handles the extraction of the true and false class data sets from the source images.

The stamp code reader sports a large region verification, parameter extraction and statistics generation routine, with various assisting subroutines. These subroutines handle the line averaging concept from the Hough transform. This project also has its own preprocessing function.

The Matlab routine used to train MINACE filters is a quick replacement for an ageing and unsupported C++ program inherited through Dr. Dave Weber from Rajesh Shenoy of Carnegie Mellon University. This program is not written for portability, relies partly on discontinued C++ matrix libraries, and includes support for various undocumented extensions to MINACE filter technology.