THE SELECTION AND EVALUATION OF GREY-LEVEL THRESHOLDS APPLIED TO DIGITAL IMAGES

Thesis

Submitted in Fulfilment of the

Requirements for the Degree of

MASTER OF SCIENCE

of Rhodes University

by

ANTON DAVID BRINK

October 1987

CONTENTS

ACKNOWLEDGEMENTS

I wish to thank my supervisor, Prof. G. de Jager, for his advice and guidance, and for initially interesting me in this project. I also thank Prof. E. E. Baart for his help during Prof. de Jager's absence.

I am greatly indebted to Justin Jonas for many hints, suggestions and tips concerning various aspects of some of the equipment and programs used in this project. I am also grateful to the twenty people who participated in the subjective threshold evaluation technique.

Special thanks go to Estelle Brink and Lorraine Latré for proof-reading the manuscript and for their encouragement and support throughout this project. Further thanks go to Rodney Baxter for his comments, suggestions and friendship.

Finally, I am indebted to the CSIR for their financial assistance during the last two years, and to Miss Heather Kew and Miss Margaret Shepherd for typing this thesis.

Anton Brink

LIST OF FIGURES AND TABLES

ABSTRACT

Many applications of image processing require the initial segmentation of the image by means of grey-level thresholding. In this thesis, the problems of automatic threshold selection and evaluation are addressed in order to find a universally applicable thresholding method. Three previously proposed threshold selection techniques are investigated, and two new methods are introduced. The results of applying these methods to several different images are evaluated using two threshold evaluation techniques, one subjective and one quantitative. It is found that no threshold selection technique is universally acceptable, as different methods work best with different images and applications.

CHAPTER ONE

INTRODUCTION

The field of digital image processing encompasses a wide variety of processing techniques. These can be roughly grouped into four general classes, those of **digitization and coding, enhancement and restoration, reconstruction** and **image analysis.** Image digitization is used to convert pictures from continuous to discrete form. The resulting digital images can then be coded so as to conserve storage space or transmission channel capacity. Enhancement and restoration techniques can be used to improve images that have been degraded by noise or blurring. Finally, image analysis involves the segmentation of images to facilitate the extraction, identification or measurement of image properties or parts. Current and prospective applications of digital image processing include, among others, unsupervised factory automation, improvements in medical X-ray and acoustic imaging, enhancement and interpretation of LANDSAT photographs as well as improvements in the quality of domestic television pictures.

The origin of the present project stems from a requirement that a local manufacturer of refrigerators had, viz, some rapid means to check that the dimensions of refrigerator panels remained within specifications. The cross-sectional dimensions of a channel in the side panels were particularly critical. These measurements had to be performed within 20 seconds and hence the use of a computer-based imaging system was suggested. A video camera interfaced to a microcomputer was used to obtain an edge view of the channel. This image is shown in Fig. 1.1, together with the dimensions to be measured.

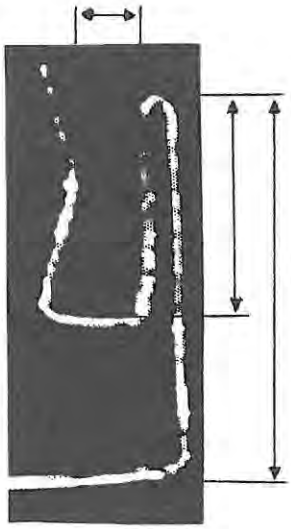


Figure 1.1 An edge view of the channel in the refrigerator panel. The black marks indicate the dimensions to be measured.

Due to difficulties in lighting and slight distortions on the panel, the edges of the panel are indistinct. This could clearly lead to errors in measurement. Ideally, all parts of the image belonging to the panel edge would be a uniform white, while the rest of the image should be uniformly black. This problem thus suggests the use of the third class of image processing techniques mentioned, that of segmentation.

Image segmentation involves the partitioning of an image into classes, for example "objects" and "background". This classification is based on one or more of the properties of the parts of the image, such as texture, shape or brightness. Since the object in this image is generally brighter than the background, the segmentation can be based on brightness. This form of segmentation is called grey-level thresholding. When a threshold is applied to an image, all parts of the image brighter than that threshold level are redefined as white,

while all parts darker than the threshold level become black. The selection of a suitable threshold should be done automatically, as speed is a consideration.

Many threshold selection techniques have been proposed. The aim of this thesis is to find or devise the best, or most reliable, method of threshold selection. This further implies that some form of threshold evaluation would be required. This also had to be devised.

In Chapter Two the capture of images, as well as the equipment used will be described. Some basic definitions will also be introduced. Chapter Three deals with threshold selection techniques. Three previously proposed methods are tested and two new techniques are described. The evaluation of thresholds and thresholding methods is dealt with in Chapter Four. A quantitative evaluation method using computer-generated test images is described, as well as a method involving subjective evaluations by 20 people. In Chapter Five the results of applying the thresholds selected to six different images are presented, together with the subjective evaluations. The methods were also applied to four computer-generated images and evaluated quantitatively. The results obtained are discussed in Chapter Six.

CHAPTER TWO

IMAGE CAPTURE

In this chapter the apparatus and methods used to capture and digitize images are described. First, however, it is necessary to introduce some fundamental definitions and explanations of terms which are used throughout the remainder of this project.

A digital image is a two-dimensional array of picture elements, or "pixels". Each pixel has a binary value corresponding to the light intensity at that point of the image. This value is called the "grey-level" of the pixel. The number of discrete grey-levels from black to white is fixed and determined by the number of digits, or "bits", that make up the binary values. This can be termed the "brightness resolution" of the image. Normally, black is represented by zero and white by the highest value that can be made up by the limited number of bits. For example, if an image has a resolution of 4 bits/pixel, then black = 0000 (0 decimal) and white = 1111 (15 decimal), yielding a total of 16 discrete grey-levels.

Six digital images were used in this project. They appear in Chapter Five. A Pulnix TM-34K CCD video camera was used to capture the images. This was interfaced to an IBM PC microcomputer via a PC-EYE Video Capture System, which is produced by Chorus Data Systems. The IBM PC was also equipped with a Tecmar Graphics Master high resolution graphics display adapter, which can display pictures of up to 640 x 400 pixels with 4 bits/pixel, or 16 grey-levels. The PC-EYE system can capture an image with a maximum resolution of 640 x 512 pixels

with 6 bits/pixel (64 grey-levels). Images with such resolution can be stored on a magnetic disk, but cannot be displayed with the Tecmar adapter.

The video camera produces an interlaced video signal which forms 30 frames per second. This signal is digitized at a rate greater than that at which the IBM PC can transfer the data to memory. Acquisition must thus be delayed and so occurs during several frames of the video signal. Every **t**-th line of the video signal is digitized, where **t** is an integer called the "throttle count". The interval between acquisitions is used to write the data to memory. Any acquisition occurring during this interval could lead to the data being over-written and lost. According to the PC-EYE User's Guide, the minimum value of the throttle count can be determined by

$$t = a + i$$

where **i** = 3 for a non-interlaced video signal

i = 5 for an interlaced video signal

a is the truncated quotient of

(Horizontal pixel count)/[(No. pixels per byte) x 50]

For the maximum resolution of 640 x 512 pixels with 6 bits/pixel, these formulae yield a minimum throttle count value of 14.

The software provided with the PC-EYE system includes a set of machine language subroutines which perform various tasks associated with the capture, storage and display of the images. The individual routines and their applications are described in the PC-EYE Technical Reference Manual. Software interfaces are provided to enable the subroutines to be called from an applications program written in BASIC.

A program, VAR6BIT.BAS, was written for the capture of images with a grey-level resolution of 6 bits/pixel. The program allows the user to set the frame size before acquisition. A throttle count of 14 was used to enable the capture of images up to 640 pixels wide. The IBM Advanced BASIC interpreter, BASICA.COM, was used to run the program. After entering the desired frame size, the user can capture and display a new image, transfer an image to a file called "JUNK.PIC" on a formatted disk in drive "B", or terminate the program by pressing <P>, <S> or <Esc>, respectively. A copy of the program listing is included in Appendix A.

The images were sent via a standard RS232C serial communications line to a Digital Equipment Corporation VAX-11/730 (VAX) minicomputer, where all further processing took place. A locally written version of Control Data Corporation's RMF system was used to do this. This version enables a microcomputer to function as a terminal for the VAX, making it possible for image files to be transferred from the IBM PC to the VAX. As there is no system user's manual available for this version of RMF, a short Standard Operating Procedure has been written (Appendix B).

The VAX was equipped with the Starlink Software Collection. This is a collection of astronomical programs and subroutines from various sources. It includes a library of subroutines for the display and handling of image files, which can be linked to any programs requiring these facilities.

Images on the VAX are stored as Bulk Data Frame files, which have ".BDF" appended to the file name. Before the PC-EYE images, appended by ".PIC", could be displayed or processed by the VAX, they had to be converted to the correct format. The main difference between the IBM PC/PC-EYE capture format and the VAX/Starlink display format lies in their respective pixel coordinate numbering conventions. PC-EYE numbers coordinates according to the standard video scanning sequence. Since the scan begins at the top left hand corner of the image, this is chosen as the origin, with coordinate values increasing downwards and to the right. The image display terminal used on the VAX numbers the coordinates according to the usual mathematical format, with the origin at the bottom left hand corner and values increasing upwards and to the right. The two conventions are depicted in Fig. 2.1.

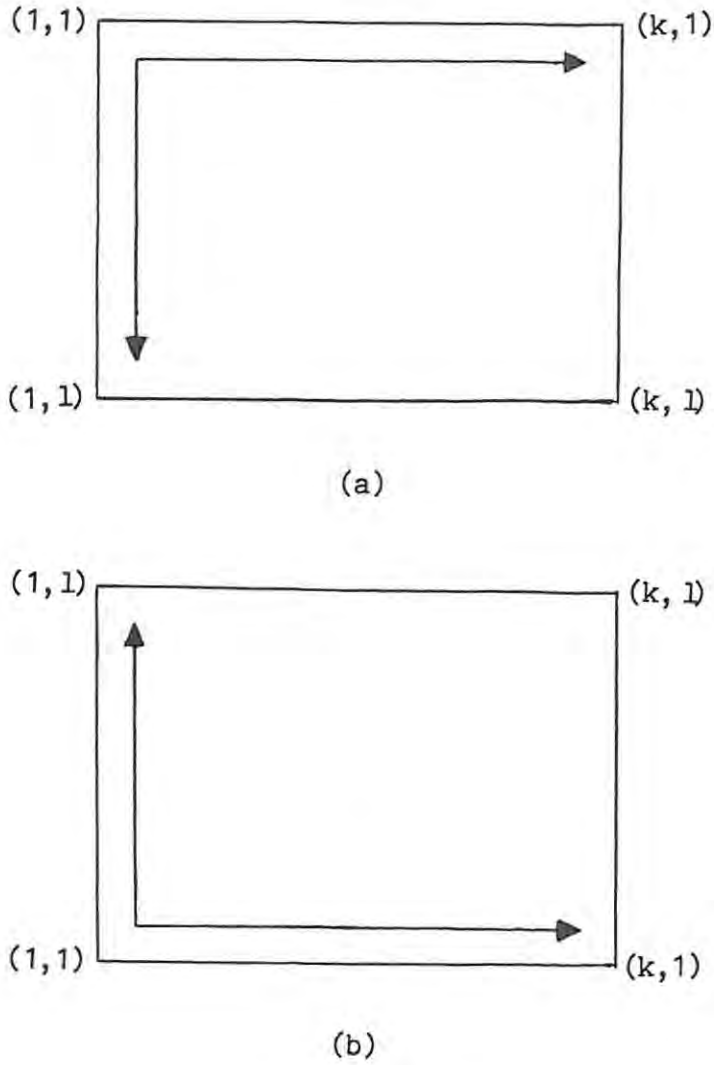


Figure 2.1 Coordinate numbering conventions. (a) PC-EYE "video scanning" format. (b) VAX display "mathematical" format.

A program, PCEYE2.FOR, to perform this conversion was written. All programs written and used on the VAX were coded in FORTRAN and are included in Appendix A.

The graphics display terminal used on the VAX has a maximum grey-level resolution of 8 bits/pixel, enabling 256 discrete grey-levels to be displayed. In order to make full use of this resolution, four images of each subject were captured with the IBM PC/PC-EYE system. These nearly-identical images each had a grey-level resolution of 6 bits/pixel. They were transferred to the VAX and, after conversion to

the correct format, were added together to form a single image with 8 bits/pixel. A program called ADDIM.FOR was used to perform the addition. Due to the noise distribution and lighting being slightly different in each of the four 64-level images, a good approximation to a single 256-level image is obtained.

All images were displayed on a Tektronix M4115B graphics terminal which has a maximum resolution of 1280 x 1024 pixels. A locally written command, TDISP.FOR, was used to display images using "lookup tables" specified by the user. A lookup table is a 3 x 256 array which is used to specify the intensities, on a scale of 0 to 255, of each of the red, green and blue primary components which make up the display colour of each of the 256 grey-levels (Fig. 2.2). Like images, lookup tables are stored as Bulk Data Frame files.

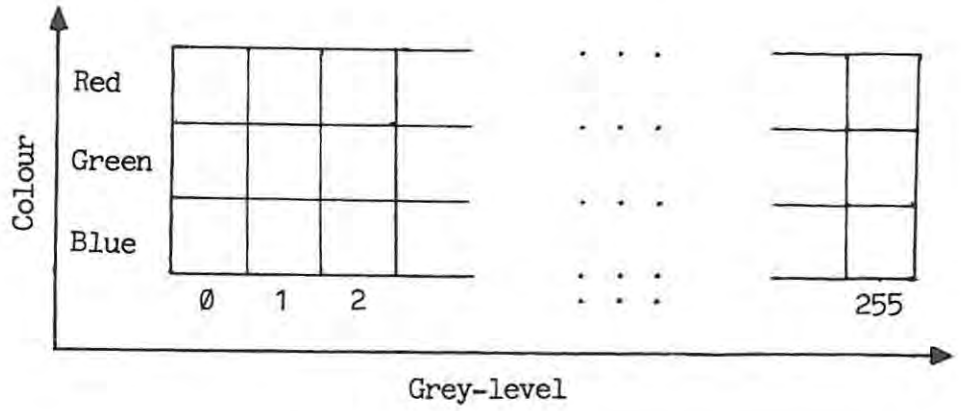


Figure 2.2 Graphic representation of the layout of a lookup table.

Images can be displayed in either false-colour or a grey-scale format. When a false-colour display is used, the grey-levels of the image are

assigned different colours by the lookup table. These colours bear no relation to the actual colours of the subject. All colours can be formed by varying the relative intensities of the three primary colours. To obtain a grey-scale display, the relative intensities must be the same for all three primary colours, from Ø (black) to 255 (white).

A standard grey-scale lookup table was used for image display in this project. This assigned uniformly increasing intensities from black to white to the image grey-level values Ø to 255, respectively. A program, GREYSCL.FOR, was written and used to create such grey-scale lookup tables.

The images captured by the system described in this chapter could be used in various applications such as measurement and counting. As mentioned in the introduction, the first problem encountered in the processing of an image is that of identifying the parts of the image. If the parts of the image differ in brightness, grey-level thresholding can be used to achieve this. Methods of threshold selection are investigated in the following chapter.

CHAPTER THREE

THRESHOLD SELECTION

In this chapter techniques for the unsupervised selection of thresholds for images, such as those captured by the system described in Chapter Two, are investigated. The chapter is divided into three parts. Section 3.1 contains a general discussion of the basic principle of threshold selection. Three interesting techniques are investigated in Section 3.2, and two new methods are introduced and described in Section 3.3.

Initially, references on the subject of thresholding were found by having an on-line search of the INSPEC database performed. The key words "picture thresholding", "threshold selection" and "entropic threshold", linked by logical "OR" operators, yielded several interesting articles. The titles and abstracts of the articles found were listed and printed, and those of interest were ordered through the Rhodes University Library. Further reference materials were found by consulting the lists of references contained in these publications and in text books on general digital image processing.

3.1 General Discussion

Many methods of threshold selection have been proposed. The simplest of these base their decision directly on the shape of the grey-level histogram [Gonzalez and Wintz (1977), Ballard and Brown (1982), Castleman (1979), Horn (1986), Pratt (1978), Kohler (1981)]. In the ideal case for thresholding, the image consists of strongly contrasting, evenly illuminated object and background regions in

roughly equal proportions. If a histogram of such an image is drawn, showing the number of pixels, or fraction of the total image, at each grey-level, it would have two well-defined peaks of roughly equal height separated by a sharp valley. Such a histogram is said to be "bimodal". In this case, the two peaks correspond to the respective object and background regions of the image, and the valley to the transitional region between them. It would thus be reasonable to select the grey-level corresponding to the valley bottom as the optimum threshold (Fig. 3.1).

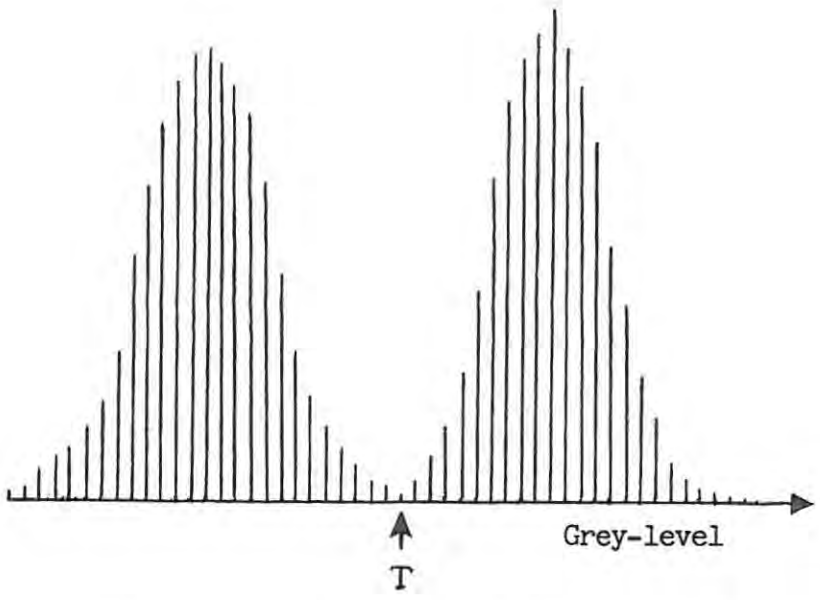


Figure 3.1 Selection of the grey-level corresponding to the valley in a bimodal histogram as the optimum threshold.

Often, however, the histogram peaks differ enormously in height. The valley may be so wide that selection of a threshold becomes ambiguous, or it may not exist at all, with one peak forming a shoulder on the other. Furthermore, the histogram may not be bimodal at all.

In order to overcome such difficulties, some techniques [Weszka, Nagel and Rosenfeld (1974), Rosenfeld and Davis (1978), Peleg (1978), Weszka and Rosenfeld (1979), Kirby and Rosenfeld (1979)] modify the histogram by sharpening the valley or converting it into a peak. Many methods make use of other properties of either the histogram or the image itself on which the choice of threshold can be based. Some of these develop a "criterion function" whose value depends on where the threshold is selected. The optimum threshold is that one at which the criterion function reaches its maximum or minimum value, depending on the form of the function. Comprehensive surveys of thresholding techniques have been presented by Weszka (1978), Kohler (1981) and Kittler, Illingworth and Föglein (1985).

3.2 Three Interesting Methods

The three methods investigated and implemented on the VAX were selected largely because of their distinctive and non-trivial approaches to the problem of threshold selection. The first of the methods, presented below, develops a criterion function based on the entropy of the histogram. The threshold selected is that one which maximises this function. The second method assumes that the histogram is bimodal. The histogram is then approximated by the sum of two Gaussian distributions, and the error involved in this approximation is evaluated as a function of the threshold value. The optimum

threshold is then that one which minimises this error. The third method used chooses the threshold in such a way that the first three moments of the original image are preserved in the thresholded bilevel image.

3.2.1 The Entropy Method

The entropy concept used in image processing is borrowed from information theory [Gonzalez and Wintz (1977), Pratt (1978), Rosenfeld and Kak (1976), Andrews and Hunt (1977), Hall (1979)]. Two algorithms for threshold selection based on the entropy of the histogram were originally proposed by Pun [Pun (1980), Pun (1981)]. In the first of these [Pun (1980)] a few errors were made in algebraic manipulations.

These errors were corrected and a new method, based on the same principles, proposed by Kapur, Sahoo and Wong (1985). Using this method, thresholds are applied to the image by iteration. For each threshold, two probability distributions are defined, one for those grey-levels above the threshold and one for those grey-levels less than or equal to the threshold. The entropy associated with each of these distributions is calculated, and the sum of these entropies is used as a criterion function. The threshold corresponding to the maximum value of this function is the optimum one.

The grey-level histogram can be viewed as a probability distribution. If f_0, f_1, ..., f_n are the observed frequencies corresponding to the grey-levels, $g = 0, 1, \ldots, n$, then the grey-level probabilities can be defined as

$$p_g = f_g / N; \qquad N = \sum_{g=\emptyset}^{n} f_g; \qquad g = \emptyset, 1, \ldots, n \tag{3.1}$$

where **N** is the total number of pixels in the picture and there are (**n** + 1) grey-levels. The entropy of the histogram is defined as

$$H_n = - \sum_{g=\emptyset}^{n} p_g \ln p_g \tag{3.2}$$

Two probability distributions are derived from the grey-level probability distribution, one for grey-levels $\emptyset$ to T and the other for levels (T + 1) to **n**, where T is a candidate threshold such that $\emptyset \leq T < n$. The two distributions are

$$A : p_\emptyset / P_\emptyset (T), p_1 / P_\emptyset (T), \ldots, p_T / P_\emptyset (T) \tag{3.3.1}$$

$$B : p_{T+1} / P_1 (T), p_{T+2} / P_1 (T), \ldots, p_n / P_1 (T) \tag{3.3.2}$$

where $P_\emptyset (T)$ and $P_1 (T)$ are the class probabilities, given by

$$P_\emptyset (T) = \sum_{g=\emptyset}^{T} p_g; \quad P_1 (T) = 1 - P_\emptyset (T) \tag{3.4}$$

The entropies associated with each distribution are

$$H_A(T) = -\sum_{g=\emptyset}^{T} [p_g / P_\emptyset(T)] \ln [p_g / P_\emptyset(T)]$$

$$= \ln P_\emptyset(T) + H_\emptyset(T) / P_\emptyset(T) \qquad (3.5.1)$$

$$H_B(T) = -\sum_{g=T+1}^{n} [p_g / P_1(T)] \ln [p_g / P_1(T)]$$

$$= \ln P_1(T) + H_1(T) / P_1(T) \qquad (3.5.2)$$

where $H_\emptyset(T)$ and $H_1(T)$ are the class entropies, given by

$$H_\emptyset(T) = -\sum_{g=\emptyset}^{T} p_g \ln p_g \quad ; \quad H_1(T) = H_n - H_\emptyset(T) \qquad (3.6)$$

The sum of the entropies $H_A(T)$ and $H_B(T)$, $\Psi(T)$, was used as the criterion for threshold selection. From equations (3.5.1) and (3.5.2),

$$\Psi(T) = \ln P_\emptyset(T) P_1(T) + H_\emptyset(T)/P_\emptyset(T) + H_1(T)/P_1(T) \qquad (3.7)$$

If $\Psi(T)$ is maximised, the maximum distinction between the object and background distributions is obtained. The value of T which maximises $\Psi(T)$ is selected as the optimum threshold.

A program, LEVELENT.FOR, was written and used to implement this method on the VAX. Plots of the histograms of the images were obtained with

two programs, HIST.FOR, which determined the grey-level frequencies, and DRAW.FOR, which used these to plot the histograms. For each image on which the entropy method was used, the variation of $\Psi(T)$ with changing T was plotted, using a program called DRAWENT.FOR.

3.2.2 The Minimum Error Method

A "Minimum Error" method has been proposed by Kittler and Illingworth (1986). This method assumes that the grey-level histogram is roughly bimodal. It is further assumed that the two peaks are approximately Gaussian in shape, and hence that the distribution can be approximated by the sum of two true Gaussian distributions. The threshold is used to divide the original distribution into two parts, each of which is then modelled by a Gaussian distribution. The threshold which produces the best fit, and hence the least error, is then chosen as the optimum threshold.

The grey-level histogram is again treated as a probability distribution. It is assumed that the two peaks of the bimodal histogram can be approximated by Gaussian distributions with means μ_0 and μ_1, standard deviations σ_0 and σ_1 and **a priori** probabilities P_0 and P_1, respectively. Thus, ideally, the grey-level probabilities are given by

$$p_g = \sum_{k=0}^{1} P_k \; p_{gk} \qquad (3.8)$$

where

$$p_{gk} = \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left[- \frac{(g - \mu_k)^2}{2\,\sigma_k^2} \right] \quad ; k = \emptyset, 1 \qquad (3.9)$$

The "minimum error threshold" is that grey-level, τ, for which the grey-levels, $g = \emptyset, 1, \ldots, n$, satisfy

$$P_\emptyset\, p_{g\emptyset} > P_1\, p_{g1} \quad \text{if} \quad g \leq \tau \qquad (3.10.1)$$

$$P_\emptyset\, p_{g\emptyset} < P_1\, p_{g1} \quad \text{if} \quad g > \tau \qquad (3.10.2)$$

Before this threshold level can be determined, the parameters $\mu_\emptyset$, μ_1, $\sigma_\emptyset$, σ_1, $P_\emptyset$ and P_1 must be known. These can be estimated from the grey-level histogram, as follows:

$$\mu_\emptyset\,(T) = \Big[\sum_{g=\emptyset}^{T} p_g\, g \Big] / P_\emptyset\,(T) \qquad (3.11.1)$$

$$\mu_1\,(T) = \Big[\sum_{g=T+1}^{n} p_g\, g \Big] / P_1\,(T) \qquad (3.11.2)$$

$$\sigma_\emptyset\,(T) = \Big[\sum_{g=\emptyset}^{T} (g - \mu_\emptyset\,(T))^2\, p_g \Big] / P_\emptyset\,(T) \qquad (3.12.1)$$

$$\sigma_1\,(T) = \Big[\sum_{g=T+1}^{n} (g - \mu_1\,(T))^2\, p_g \Big] / P_1\,(T) \qquad (3.12.2)$$

where $P_{\emptyset}(T)$ and $P_1(T)$ are the class probabilities given by (3.4), p_g are the grey-level probabilities from (3.1) and T is a candidate threshold. The Gaussian models of (3.9) now become

$$p_{gk}(T) = \frac{1}{\sqrt{2\pi}\ \sigma_k(T)} \exp\left[- \frac{(g - \mu_k(T))^2}{2\,\sigma_k^2(T)} \right] ; \quad k = \emptyset, 1 \qquad (3.13)$$

The conditional probability, $e_g(T)$, of grey-level g being replaced in the image by a correct binary value is

$$e_g(T) = P_k(T)\, p_{gk}(T) / p_g \quad ; \quad g = \emptyset, 1, \ldots, n \qquad (3.14)$$

where $k = \emptyset$ if $g \leq T$ and $k = 1$ if $g > T$.

Since p_g is independent of both k and T, the authors chose to ignore the denominator in the analysis. The true effect that this has on the behaviour of $e_g(T)$ with varying T may be of interest. This has not been investigated here. Taking the natural logarithm of the numerator in (3.14), adding ln 2π and multiplying by -2 yields

$$\varepsilon_g(T) = [(g - \mu_k(T))/\sigma_k(T)]^2 + 2\ln \sigma_k(T) - 2\ln P_k(T) \qquad (3.15)$$

where $k = \emptyset$ if $g \leq T$ and $k = 1$ if $g > T$. $\varepsilon_g(T)$ is a measure of classification error. The average classification error for the whole

image can then be characterised by the criterion function

$$J(T) = \sum_{g=\emptyset}^{n} p_g \, \varepsilon_g(T) \qquad (3.16)$$

Substituting from (3.15), (3.4), (3.11) and (3.12) into J (T) yields

$$\begin{aligned} J(T) = 1 &+ 2\,[P_\emptyset\,(T)\,\ln\sigma_\emptyset\,(T) + P_1\,(T)\,\ln\sigma_1\,(T)] \\ &- 2\,[P_\emptyset\,(T)\,\ln P_\emptyset\,(T) + P_1\,(T)\,\ln P_1\,(T)] \end{aligned} \qquad (3.17)$$

which is easier to compute. As the threshold, T, is varied, the Gaussian models change. The better the fit between the data and the models, the smaller the classification error becomes. The threshold yielding the lowest value of J (T) will give the best fit and this is then the minimum error threshold, τ .

The minimum value of J (T), and hence the threshold, was found by iteration, using the program LEVELMIN.FOR. DRAWMIN.FOR was used to plot graphs depicting the behaviour of J (T) as T is varied.

3.2.3 The Moment-Preserving Method

Tsai (1985) describes an unusual approach to thresholding whereby the first three moments of the original image are preserved in the thresholded bilevel image. Using this approach the correct threshold

is obtained without the need for iteration. A representative grey-level is also obtained for each of the two thresholded classes. The r-th moment, m_r, of the image can be obtained from the grey-level probability histogram by

$$m_r = \sum_{g=\emptyset}^{n} p_g \, g^r \tag{3.18}$$

where $g = \emptyset, 1, \ldots, n$ are the image grey-levels and p_g is given by (3.1). Let the bilevel image consist of pixels with the two grey-levels $z_\emptyset$ and z_1, where $z_\emptyset < z_1$, and let $P_\emptyset$ and P_1, $P_\emptyset + P_1 = 1$, denote the fractions of the below-threshold and above-threshold pixels, respectively. Then the first three moments of the bilevel image are given by

$$m'_r = \sum_{k=\emptyset}^{1} P_k \, (z_k)^r \quad ; \quad r = 1, 2, 3 \tag{3.19}$$

For these moments to be preserved implies that

$$m'_r = m_r \quad ; \quad r = 1, 2, 3 \tag{3.20}$$

From (3.19) and (3.20) can be obtained four equalities, as follows

$$P_0 z_0^0 + P_1 z_1^0 = m_0 = 1 \qquad (3.21.1)$$

$$P_0 z_0^1 + P_1 z_1^1 = m_1 \qquad (3.21.2)$$

$$P_0 z_0^2 + P_1 z_1^2 = m_2 \qquad (3.21.3)$$

$$P_0 z_0^3 + P_1 z_1^3 = m_3 \qquad (3.21.4)$$

where m_r, $r = 1, 2, 3$, are given by (3.18). To find the optimum threshold, T, these equations must be solved to yield P_0 and P_1. T is then chosen to satisfy

$$P_0 = \sum_{g=0}^{T} p_g = P_0 (T) \qquad (3.22)$$

that is, it is the P_0-tile of the histogram. In practice, there may be no discrete grey-level which is exactly the P_0-tile of the histogram. In such a case, the grey-level closest to the P_0-tile is chosen to be the threshold.

In order to solve the equations (3.21) for P_0 and P_1, the values of z_0 and z_1 are required. The process of calculating these can be summarised by the following equations.

$$z_0 = \tfrac{1}{2}\left[-c_1 - \sqrt{c_1^2 - 4c_0}\right] \qquad (3.23.1)$$

$$z_1 = \tfrac{1}{2}\left[-c_1 + \sqrt{c_1^2 - 4c_0}\right] \qquad (3.23.2)$$

where

$$c_0 = (1/c_d)\begin{vmatrix} -m_2 & m_1 \\ -m_3 & m_2 \end{vmatrix} \qquad (3.24.1)$$

$$c_1 = (1/c_d)\begin{vmatrix} m_0 & -m_2 \\ m_1 & -m_3 \end{vmatrix} \qquad (3.24.2)$$

$$c_d = \begin{vmatrix} m_0 & m_1 \\ m_1 & m_2 \end{vmatrix} \qquad (3.24.3)$$

Then

$$P_0 = (1/P_d)\begin{vmatrix} 1 & 1 \\ m_1 & z_1 \end{vmatrix} \qquad (3.25.1)$$

$$P_1 = 1 - P_0 \qquad (3.25.2)$$

where

$$P_d = \begin{vmatrix} 1 & 1 \\ z_0 & z_1 \end{vmatrix} \qquad (3.25.3)$$

This method was implemented using a program called LEVELMOM.FOR. The results of applying this method, the Entropy method and the Minimum Error method to the six images used, are presented in Chapter Five.

3.3 Two New Methods

Two new techniques of threshold selection have been formulated and are presented here. Both methods involve a comparison of the bilevel resultant image with the original image. The first of these methods involves evaluating the sum of the magnitudes of the differences in grey-level between corresponding pixels of the bilevel image and the original image. The optimum threshold is that one which minimises this sum. The second method uses a statistical measure of correlation between the original image and the bilevel result. The correlation varies with threshold value, and the optimum threshold maximises this correlation.

3.3.1 The Minimum Difference Method

A direct approach image thresholding involving the evaluation of the sum of the magnitudes of the differences in grey-level of corresponding pixels in the original and bilevel images is introduced here. Thresholds are applied iteratively to the image. For each iteration, the below- and above-threshold mean grey-levels are chosen to be the "black" and "white" levels of the binary image, respectively.

For threshold T, the sum of the differences, X (T), can be expressed

$$X(T) = \sum_{g=\emptyset}^{T} |\mu_{\emptyset}(T) - g| p_g + \sum_{g=T+1}^{n} |\mu_1(T) - g| p_g \quad ; \quad \emptyset \leq T < n \quad (3.26)$$

where p_g are the grey-level probabilities (3.1) and $\mu_{\emptyset}(T)$ and $\mu_1(T)$ are the below- and above-threshold means given by (3.11.1) and (3.11.2). The optimum threshold was that level at which X (T) was minimised.

In approach, this method is very similar to those used by Otsu [Otsu (1978), Otsu (1979)], especially the "Least Squares Criterion" which differs in that the sum of the squares of differences, rather than the sum of the magnitudes of differences, is used.

LEVELAB5.FOR was used to implement this method on the VAX. Since X (T) varies with T, plots were produced depicting this variation.

3.3.2 The Maximum Correlation Method

Another thresholding technique involving direct comparison of the bilevel result with the original image is described here. In this case, the correlation between the original image and the bilevel image is evaluated for different thresholds. The optimum threshold maximises this correlation.

The coefficient of correlation [Freund (1979), Oppenheim and Schafer (1975)], ρ_{xy}, for two sets of data, $X = \{x_1, x_2, \ldots, x_s\}$ and $Y = \{y_1, y_2, \ldots, y_s\}$, is given by

$$\rho_{xy} = \frac{E_{xy} - E_x E_y}{\sqrt{V_x V_y}} \tag{3.27}$$

where E_x and E_y are the expected values, or mathematical expectations, of the sets of data, respectively, and E_{xy} the expected value of their product. V_x and V_y are the variances of the two sets of data.

The expected values of X, Y and their product are given by

$$E_x = \sum_{i=1}^{s} x_i \, p(x_i) \tag{3.28.1}$$

$$E_y = \sum_{i=1}^{s} y_i \, p(y_i) \tag{3.28.2}$$

$$E_{xy} = \sum_{i=1}^{s} x_i \, y_i \, p(x_i \, y_i) \tag{3.28.3}$$

where x_i and y_i are data values, $p(x_i)$ and $p(y_i)$ their associated probabilities and $p(x_i\ y_i)$ the probability associated with their product. The variances V_x and V_y are given by

$$V_x = E_{xx} - [E_x]^2 \tag{3.29.1}$$

$$V_y = E_{yy} - [E_y]^2 \tag{3.29.2}$$

where

$$E_{xx} = \sum_{i=1}^{s} x_i^2\, p(x_i) \tag{3.30.1}$$

$$E_{yy} = \sum_{i=1}^{s} y_i^2\, p(y_i) \tag{3.30.2}$$

When $\rho_{xy} = 0$, there is no correlation between X and Y. If $\rho_{xy} = -1$, $Y = -cX$, which is an inverse correlation, and if $\rho_{xy} = 1$, the correlation is absolute and $Y = cX$, where c is some arbitrary constant.

Let X represent the grey-levels of the individual pixels in the original image and Y represent those of the bilevel image. If an image is made up of k x l pixels, each pixel constitutes a fraction 1/N of that image, where $N = kl$. This approach is equivalent to using the histogram as a probability distribution. The expected values of equation (3.28) now become

$$E_x = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij}\,(1/N) \tag{3.31.1}$$

$$E_y(T) = \sum_{i=1}^{k} \sum_{j=1}^{l} y_{ij}\,(1/N) \quad ; \quad 0 \le T < n \tag{3.31.2}$$

$$E_{xy}(T) = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij}\, y_{ij}\,(1/N) \quad ; \quad 0 \le T < n \tag{3.31.3}$$

where x_{ij} is the grey-level of pixel (i, j) in the original image, y_{ij} is the value ("black" or "white") of pixel (i, j) in the bilevel image and n is the highest grey-level. The variances are now given by

$$V_x = E_{xx} - [E_x]^2 \qquad (3.32.1)$$

$$V_y(T) = E_{yy}(T) - [E_y(T)]^2 \qquad (3.32.2)$$

where

$$E_{xx} = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij}^2 \, (1/N)$$

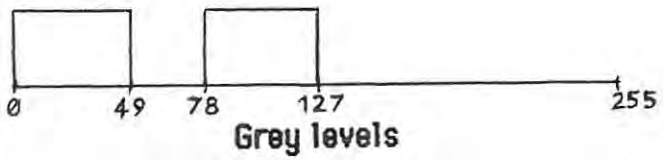
$$E_{yy}(T) = \sum_{i=1}^{k} \sum_{j=1}^{l} y_{ij}^2 \, (1/N) \quad ; \quad 0 \leq T < n$$

Only E_x and V_x are independent of the threshold because they are obtained from the original image. The correlation criterion becomes

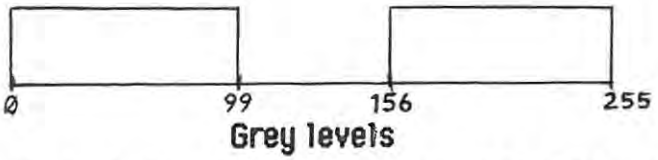
$$\rho_{xy}(T) = \frac{E_{xy}(T) - E_x E_y(T)}{\sqrt{V_x V_y(T)}} \quad ; \quad 0 \leq T < n \qquad (3.33)$$

The optimum threshold was selected by iteration as that threshold, T, which maximised $\rho_{xy}(T)$ for each of the test images. The "black" and "white" levels of the bilevel image were represented by the below- and above-threshold means ($\mu_0(T)$ and $\mu_1(T)$ respectively) given by (3.11.1) and (3.11.2). This method was also tested using various other levels and was found to be independent of their values. LEVELAB6.FOR was used to implement this method using the faster computation formulae given in Appendix C. Graphs showing how the correlation varies with changing T were plotted using PLOTAB6.FOR.

The two methods introduced here are unaffected by grey-level expansion in that they will select the same relative threshold on the expanded histogram as on the original. This results in the same final bilevel image. Consider the following histogram:



Here both techniques select a grey-level threshold T=49. If a grey-level expansion is carried out on this histogram, yielding



then both techniques select a threshold T=99, which yields the same resultant image.

It is interesting to note that in each of these cases the threshold selected occurs at the last non-zero probability before the histogram "valley". This is a valid threshold since all grey-levels less than or equal to T are mapped to "black" on the bilevel image, while all grey-levels greater than T are mapped to "white". The decision point thus occurs between T and T + 1.

It is also useful to note that, since the valley is flat, any threshold chosen within it will yield the same result. The selection techniques select the first threshold that will yield this result.

After the thresholds for the images were selected by the five methods described in this chapter, the resultant bilevel images were created using a program called TWOTONE.FOR. As input, the program requires the original image file name, a file name for the result, the threshold level and the two grey-level values for the result. The bilevel images are presented in Chapter Five.

In this chapter, threshold selection methods have been described. These methods generally chose different values for the optimum threshold, even when applied to the same image. Clearly, some technique whereby the resulting thresholds, or the techniques that chose them, can be evaluated is required. This problem is addressed in the next chapter.

CHAPTER FOUR

THRESHOLD EVALUATION

In Chapter Three, five methods of threshold selection are described. These techniques all differ in approach to the problem of threshold selection and generally choose different values for the optimum threshold, even when applied to the same image. Clearly, some criterion for selecting one threshold rather than another should be introduced.

The evaluation of thresholds and threshold selection techniques has so far been largely subjective. Relatively few viable quantitative evaluation techniques have been proposed. Some authors have addressed the problem indirectly, for example, the "Minimum Error" thresholding method of Kittler and Illingworth (1986) described in Chapter 3, Section 3.2.2. Otsu's [Otsu (1978), Otsu (1979)] methods for threshold selection first established a criterion for evaluating "goodness" of threshold, which is then optimised to find the "best" threshold. The Minimum Difference and Maximum Correlation thresholding methods described in Chapter 3, Section 3.3. adopt a similar approach.

Weszka and Rosenfeld (1978) have proposed two threshold evaluation techniques which are also used for threshold selection. These involve the use of a discrepancy or error measure, and a busyness or roughness criterion, respectively. Using these criteria, the "best" thresholds for given images are defined and hence selected.

These methods all attempt to define the "correct" or "ideal" bilevel image, and compare the results yielded by other thresholding methods with this. Since the definition of this ideal image is either subjective or the result of some particular threshold selection technique, this approach will still lead to a subjective or, at least, biased evaluation.

To develop a truly quantitative threshold evaluation technique it would be necessary to start off with an "ideal" bilevel image and then to model a physical system that, by degrading the image, turns it into a grey-level image. The processes involved in producing such a grey-level image are blurring, due to the point-spread function of a circular aperture, such as a camera's, and the addition of noise. The results of applying thresholding techniques to such a grey-level image can be evaluated by comparison with the original undegraded bilevel image, and a quantitative error can be calculated.

The test image of Fig. 4.1 was generated on the VAX by TESTBOX.FOR.

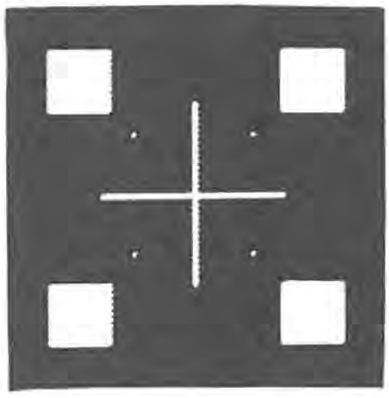


Figure 4.1 Computer-generated bilevel test image, TESTPIC.

To prevent under- and overflow when adding noise to this image, the contrast was reduced by using two intermediate grey-levels (63 and 191) instead of black (Ø) and white (255). Contrast is usually also reduced in the type of image capture system that is being modelled here.

The image was blurred by convolving it with the point spread function, or impulse response, of a circular aperture [Castleman (1979)]. If a circular aperture of diameter a is placed a distance d_i from the image plane, the point spread function in the image plane, h (r), is given by

$$h(r) = \left[2 \frac{J_1[\pi(r/r_\emptyset)]}{\pi(r/r_\emptyset)} \right]^2 \qquad (4.1)$$

$$r_\emptyset = \lambda(d_i/a) \quad ; \quad r = \sqrt{x^2 + y^2} \qquad (4.2)$$

where x and y are the spatial cartesian coordinates, J_1 is a first-order Bessel function and λ is the mean wavelength of the incident light.

Since the function under consideration has circular symmetry, it is more convenient to use Hankel Transforms rather than Fourier Transforms. The Hankel transform, F (q), of an arbitrary function f (r) is given by Bracewell (1978) as

$$F(q) = 2\pi \int_\emptyset^\infty f(r)\, J_\emptyset(2\pi qr)\, r\, dr$$

and the reverse-transform as

$$f(r) = 2\pi \int_{\emptyset}^{\infty} F(q)\ J_{\emptyset}(2\pi qr)\ q\ dq$$

where $q = \sqrt{u^2 + v^2}$ is the radial polar coordinate in the transform domain, u and v the cartesian coordinates in the transform domain and $J_{\emptyset}$ is a zero-order Bessel function. The Hankel transform is essentially a circularly symmetric form of the two-dimensional Fourier Transform.

The Hankel transform of the point spread function is the optical transfer function, $H(q)$, given by

$$H(q) = (1/\pi)\ \Pi(q/2f_c)\ [\beta - \sin\beta] \tag{4.3}$$

$$f_c = a/(\lambda d_i) \tag{4.4}$$

$$\beta = \cos^{-1}(q/f_c) \tag{4.5}$$

where Π is a "pillbox" function [Bracewell (1978)] of diameter $2f_c$ and f_c is a spatial cutoff frequency. $h(r)$ and $H(q)$ are shown in Fig. 4.2.

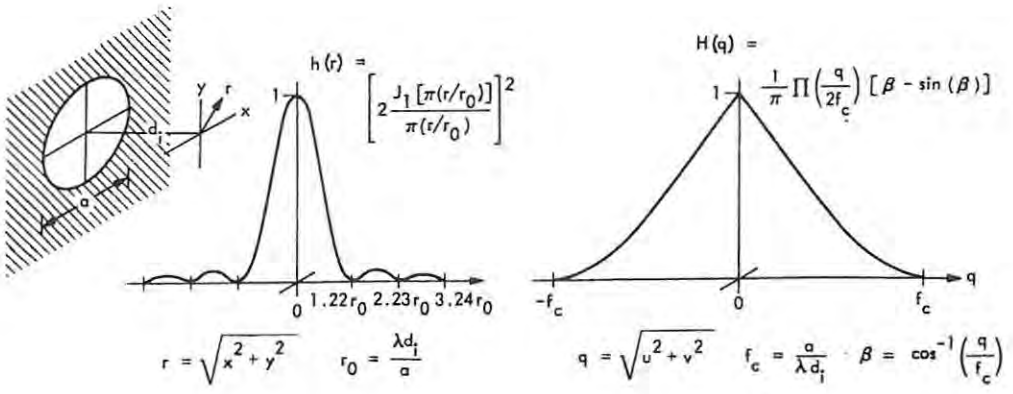


Figure 4.2 The point spread function and optical transfer function of a circular aperture. The figure is reproduced from Castleman (1979).

A locally written program, FOURFILT.FOR, was used to find the two-dimensional Fourier Transform of an input image, multiply the resulting spectrum and the optical transfer function together and reverse-transform the result to obtain the smoothed image. This operation is equivalent to convolving the image with the point spread function in the spatial domain.

FOURFILT was originally written for use with astronomical maps of the sky, and thus requires the user to specify the interval between successive pixels, both horizontally and vertically, by means of the respective scale factors, DX and DY, expressed in terms of spatial wavelengths per degree of scan. This is meaningless in the present application, and both were simply set to unity, which conveniently resulted in a spectrum of unit width and cutoff frequency $f_c = \frac{1}{2}$ cycles/pixel. FOURFILT also allows the creation of up to three output files for the image transform, the filtered image transform and the

spatial domain filtered image, respectively. Only the final filtered image was required here.

As well as the image file to be filtered, FOURFILT requires a "filter profile" specifying the shape of an isotropic filter from the origin to the cutoff frequency. The "filter" used in this case is the optical transfer function (4.3) of a circular aperture. CHINHAT.FOR was written to create filter profiles according to the equation (4.3), with a cutoff frequency specified by the user. Two profiles, CONV2 and CONV4, with $f_c = \frac{1}{2}$ cycles/pixel and $f_c = \frac{1}{4}$ cycles/pixel, respectively, were used to produce two smoothed versions of the test picture. Since edges in an image correspond to high spatial frequencies, it follows that the filter with $f_c = \frac{1}{4}$ cycles/pixel will smooth the image to a greater extent. The filter profiles appear in Fig. 4.3.

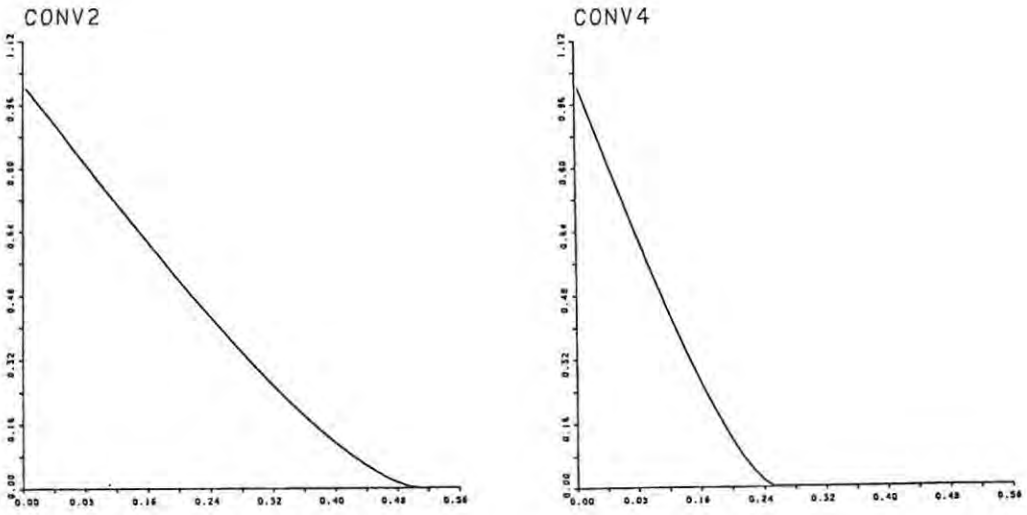


Figure 4.3 Optical transfer function profiles. (a) $f_c = \frac{1}{2}$ cycles/pixel. (b) $f_c = \frac{1}{4}$ cycles/pixel.

The resulting image pixels had real values which fell between the integer grey-levels. ROUNDOFF.FOR is a simple program written to round off these values to the nearest integers.

Random Gaussian noise was generated by converting uniformly distributed random numbers, produced by the FORTRAN function "RAN", to random numbers with a Gaussian probability distribution. A method described by Zelen and Severo (1964) was used to do this.

If U_1 and U_2 are two random numbers with a uniform probability distribution between 0 and 1, then

$$X_1 = \sqrt{-2 \ln U_1} \cos 2\pi U_2 \qquad (4.6.1)$$

$$X_2 = \sqrt{-2 \ln U_1} \sin 2\pi U_2 \qquad (4.6.2)$$

where X_1 and X_2 are two random numbers with a Gaussian probability distribution with zero mean and unit standard deviation.

NOISE.FOR uses this method to create Gaussian noise image files. The image size and the required standard deviation of the noise must be specified by the user. For an image of dimensions k x l, N = kl uniformly distributed random numbers are generated. These are used in pairs in equations (4.6.1) and (4.6.2) to create N normally distributed random numbers. To change the standard deviation of the distribution, all the normally distributed random numbers are

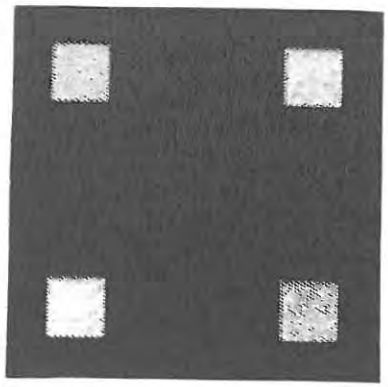
multiplied by the desired value. The numbers are then rounded off to the nearest integers, as the image grey-levels are integers.

Two image files, with the same dimensions as the test image, containing Gaussian noise with standard deviations of 2 and 4, respectively, were created. These were added in turn to each of the two smoothed images, using ADDIM.FOR, to obtain the four degraded images of Fig. 4.4.

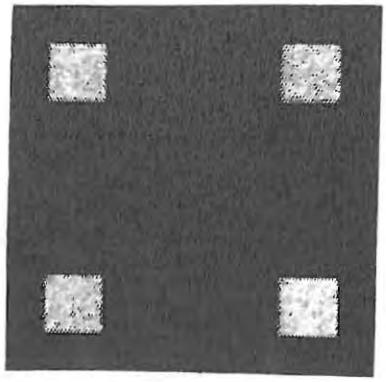
The effect of doubling the standard deviation of the noise added to a signal, as is done here, is to cause a drop in the signal-to-noise ratio of approximately 6 dB. To show this, the signal-to-noise ratio for each of the degraded images has been calculated (Fig. 4.4). If the grey-level of pixel (i, j) in the "clean" image is represented by F(i, j) and that of the corresponding pixel in the noisy image by $\hat{F}$ (i, j), then the signal-to-noise ratio [Pratt (1978)] is given by

$$SNR = -10 \log_{10} \left(\frac{\sum_{i=1}^{k} \sum_{j=1}^{l} [F(i,j) - \hat{F}(i,j)]^2}{\sum_{i=1}^{k} \sum_{j=1}^{l} [F(i,j)]^2} \right)$$

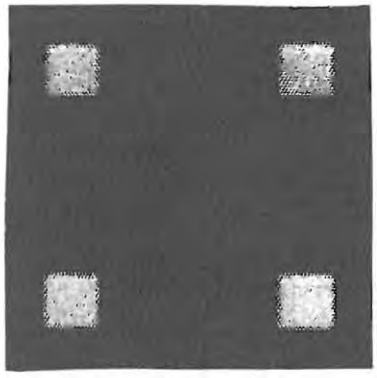
SNR.FOR used this formula to calculate the signal-to-noise ratio for each of the degraded images.



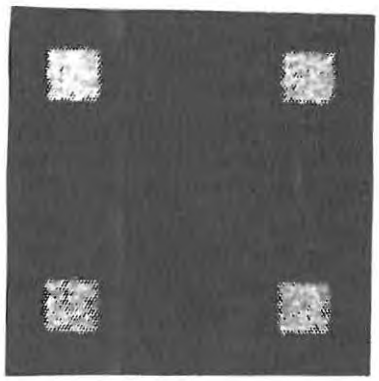
(a)



(b)



(c)



(d)

Figure 4.4 Degraded versions of the test image, TESTPIC.

(a) WPIC22 : $f_c = \frac{1}{2}$ cycles/pixel, $\sigma = 2$: SNR = 32.3464 dB

(b) WPIC24 : $f_c = \frac{1}{2}$ cycles/pixel, $\sigma = 4$: SNR = 26.4239 dB

(c) WPIC42 : $f_c = \frac{1}{4}$ cycles/pixel, $\sigma = 2$: SNR = 32.2272 dB

(d) WPIC44 : $f_c = \frac{1}{4}$ cycles/pixel, $\sigma = 4$: SNR = 26.3046 dB

The five thresholding methods described in Chapter Three were applied to each of these degraded images in turn. The results appear in Chapter Five. To evaluate the thresholds chosen, the correlation, ρ_{xy}, between each thresholded bilevel image and the original, undegraded bilevel image, TESTPIC, was calculated. The correlation is given by

$$\rho_{xy} = \frac{E_{xy} - E_x E_y}{\sqrt{V_x V_y}} \qquad (4.7)$$

$$E_x = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij} \, (1/N)$$

$$E_y = \sum_{i=1}^{k} \sum_{j=1}^{l} y_{ij} \, (1/N)$$

$$E_{xy} = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij} \, y_{ij} \, (1/N)$$

where E_x is the expected value of the original test picture, E_y is the expected value of the thresholded image, E_{xy} is the expected value of their product and $N = kl$ is the total number of pixels in each $k \times l$ pixel image. x_{ij} is the value (black or white) of pixel (i, j) in the test picture and y_{ij} is the value of pixel (i, j) in the thresholded image. V_x and V_y are the respective variances of the two images, given by

$$V_x = E_{xx} - [E_x]^2$$

$$V_y = E_{yy} - [E_y]^2$$

where

$$E_{xx} = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij}^2 \, (1/N)$$

$$E_{yy} = \sum_{i=1}^{k} \sum_{j=1}^{l} y_{ij}^2 \, (1/N)$$

XCORPIC.FOR calculated the correlation for each thresholded result. These results are presented in Chapter Five along with the thresholded images for ease of comparison.

An alternative method of threshold evaluation when an "ideal" result is not available would of necessity be more subjective. This is the case when attempting to evaluate the results of thresholding images captured by the system described in Chapter Two . A purely subjective approach to evaluation was adopted, whereby photographs of the original images, together with the bilevel results, were presented to twenty randomly chosen people. They were asked to rank the results for each image on a scale of 0 to 4, with 4 being the "best" and 0 the "worst". The mean of the 20 values thus assigned to each result was used to evaluate the thresholds. The choice of "best" or "worst" results was based on how recognizable the resulting image was, i.e. how closely it approximated the original. Although strictly such an approach is only valid with conventional images such as GIRL, as opposed to images with scientific or industrial applications, it was used for all the images in order to compare evaluations.

In Chapter Three, five threshold selection techniques were described. The results of applying these techniques to both "real" and computer-

generated images were evaluated using the two methods described in this chapter. The results obtained with the techniques described in these two chapters are presented in the following chapter.

CHAPTER FIVE

RESULTS

Five images were captured using the system described in Chapter Two. A sixth image, a standard portrait of a girl, was obtained from the Massachusetts Institute of Technology, U.S.A. These images were thresholded using the five threshold selection techniques of Chapter Three. The resulting bilevel images and the originals are presented in Section 5.1. The subjective evaluation technique described in Chapter Four was applied to these images and the results shown in Table 5.1. The thresholding techniques were also applied to the computer-generated and degraded images described in Chapter Four. These images and the bilevel results are presented in Section 5.2, together with the tabulated results of the quantitative evaluation technique of Chapter Four.

5.1 Thresholds Applied to Captured Images

The following pages contain the results of thresholding the images captured with the system described in Chapter Two, using the threshold selection techniques of Chapter Three. The image grey-level histograms and, where applicable, the graphs of the decision-criterion functions associated with some of the thresholding methods are also included.

In Table 5.1 the results of applying the subjective evaluation technique of Chapter Four to the bilevel images are presented, and the mean and standard deviation of the results for each threshold selection technique are calculated.

Figure 5.1 (**Overleaf**) Thresholding methods applied to GIRL, a standard portrait of a girl obtained from M.I.T. It has a resolution of 476 x 476 pixels with 256 grey-levels.

(a) The original image

(b) The Entropy Method: threshold T = 122

(c) The Minimum Error Method : threshold T = 64

(d) The Moment-Preserving Method : threshold T = 124

(e) The Minimum Difference Method : threshold T = 110

(f) The Maximum Correlation Method : threshold T = 116

(a) (b)

(c) (d)

(e) (f)

Fig. 5.1

Figure 5.2 (Overleaf) GIRL : the grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram

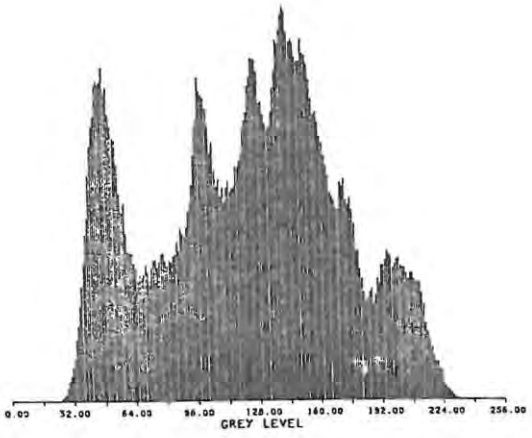
(b) The Entropy Method: the graph shows how the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, varies with changing threshold, T. It should be noted that the flat-topped part of the graph that appears to be a maximum is, in fact, negative, having been inverted by the plotting routine. The optimum threshold is selected where $\Psi(T)$ reaches its maximum value.

(c) The Minimum Error Method: a measure of average pixel classification error, J (T), derived in Chapter 3.2.2, is plotted against threshold, T. The point of minimum error yields the optimum threshold.

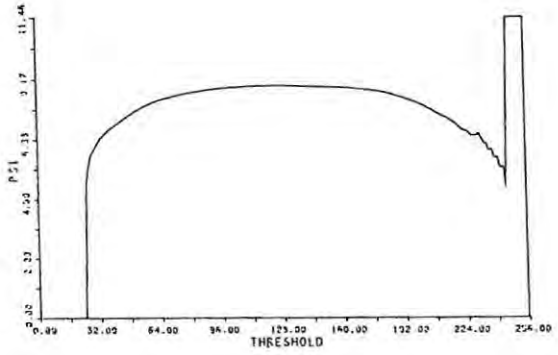
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original and bilevel images, X (T), derived in Chapter 3.3.1, is plotted against threshold, T. The level where X (T) is minimised is selected as the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, with changing threshold, T. The threshold corresponding to the maximum correlation is selected. This method is described in Chapter 3.3.2.

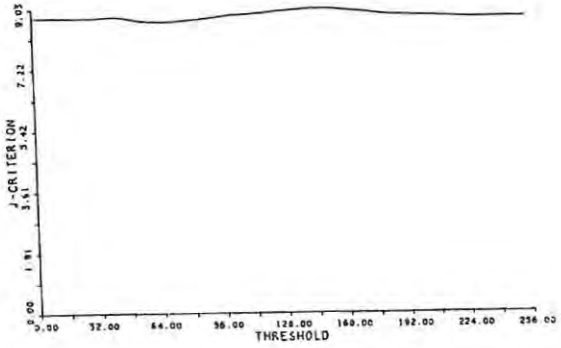
In Figure 5.2 (c), the minimum of the J(T) criterion is hard to define. This is mainly due to the Minimum Error method's dependence on the bimodality of the histogram, which, in this case, is non-existent. The graphs of Figure 5.2 (d) and (e) both show great variation with threshold, and the choice of optimum threshold in each case is clearly unambiguous. It could thus be expected that these methods would produce good results.


0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
GREY LEVEL

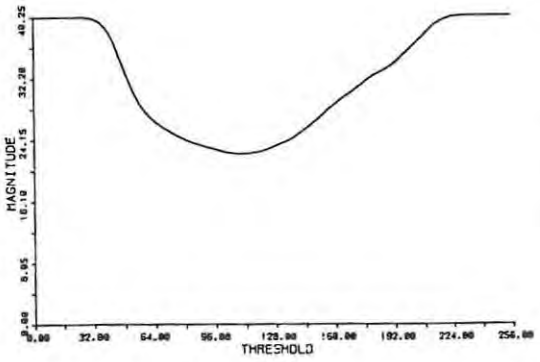
(a)


P*21
THRESHOLD

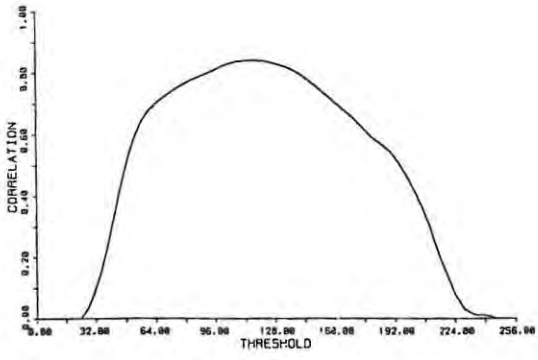
(b)


J-CRITERION
THRESHOLD

(c)


MAGNITUDE
THRESHOLD

(d)


CORRELATION
THRESHOLD

(e)

Fig. 5.2

Figure 5.3 **(Overleaf)** Thresholding methods applied to PAN8, an edge of a channel in a refrigerator panel. It has a resolution of 76 x 200 pixels with 256 grey-levels.

(a) The original image

(b) The Entropy Method: threshold T = 74

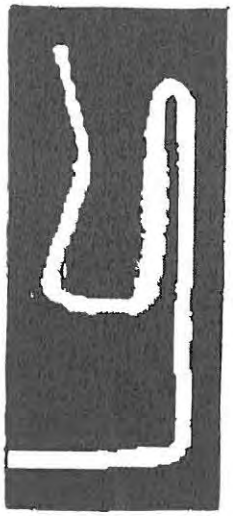
(c) The Minimum Error Method: threshold T = 24

(d) The Moment-Preserving Method: threshold T = 77

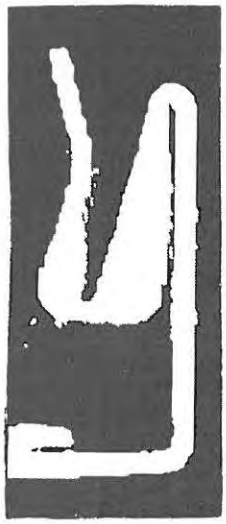
(e) The Miminum Difference Method: threshold T = 58

(f) The Maximum Correlation Method: threshold T = 73

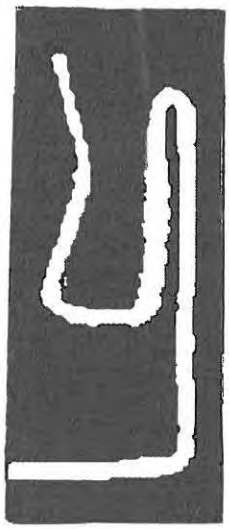
(a)



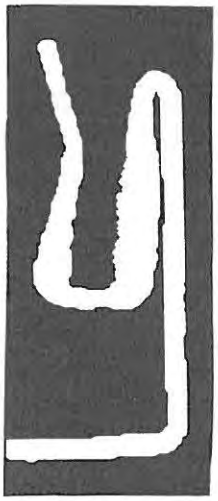
(b)



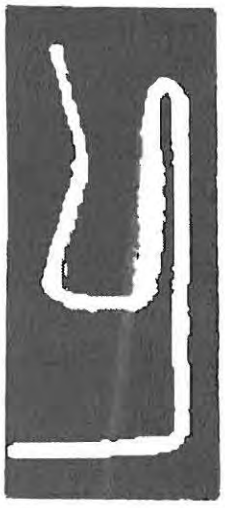
(c)



(d)



(e)



(f)

Fig. 5.3

Figure 5.4 (Overleaf) PAN8: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram.

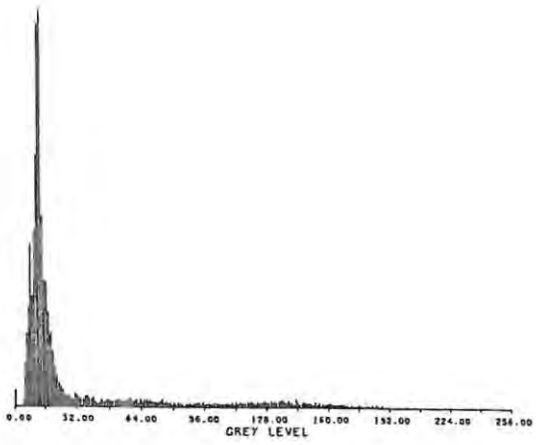
(b) The Entropy Method: a graph showing the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, with changing threshold, T. The optimum threshold is the level which maximises $\Psi(T)$.

(c) The Minimum Error Method: a measure of average classification error, J(T), derived in Chapter 3.2.2, is plotted against threshold, T. The error is minimised by the optimum threshold.

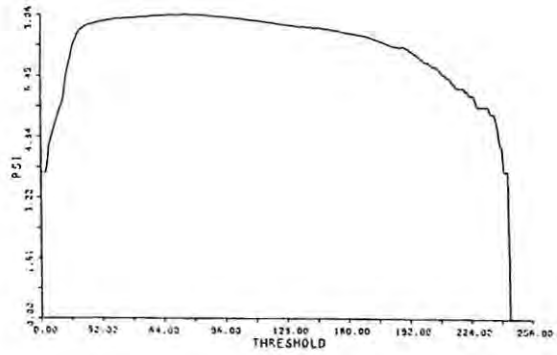
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original and bilevel images, X(T), derived in Chapter 3.3.1, is plotted against threshold, T. The level where X(T) is minimised is selected as the "best" threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, with changing threshold, T. The threshold which yields the maximum correlation between the original and bilevel images is selected. The technique is described in Chapter 3.3.2.

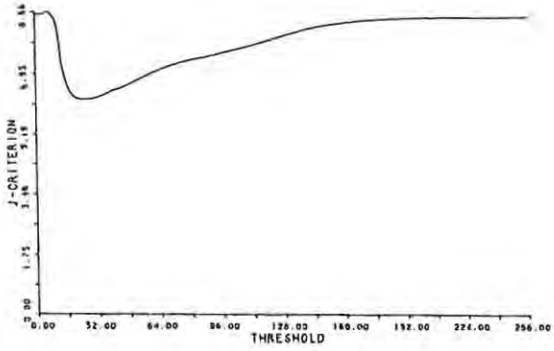
Here all the graphs display reasonable variation with changing threshold and in each case the choice of threshold is unambiguous.


0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
GREY LEVEL

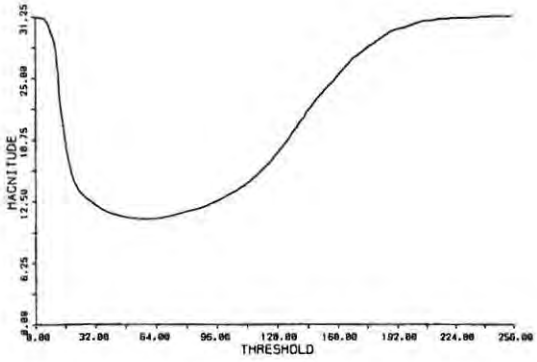
(a)


PSI
THRESHOLD

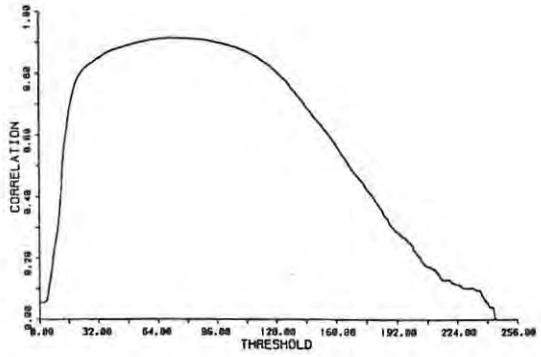
(b)


J-CRITERION
THRESHOLD

(c)


MAGNITUDE
31.25
25.00
18.75
12.50
6.25
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
THRESHOLD

(d)


CORRELATION
1.00
0.80
0.60
0.40
0.20
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
THRESHOLD

(e)

Fig. 5.4

Figure 5.5 **(Overleaf)** Thresholding methods applied to SIGN8, a sample of handwriting. The image has a resolution of 300 x 100 pixels with 256 grey-levels.

(a) The original image

(b) The Entropy Method: threshold T = 222

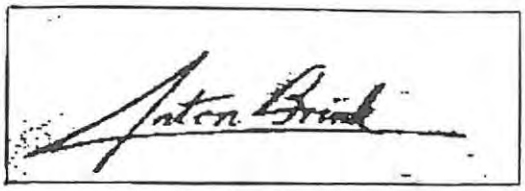
(c) The Minimum Error Method : threshold T = 221

(d) The Moment-Preserving Method: threshold T = 223

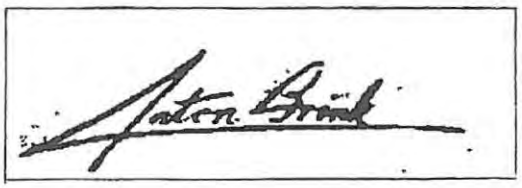
(e) The Minimum Difference Method: threshold T = 220

(f) The Maximum Correlation Method: threshold T = 219

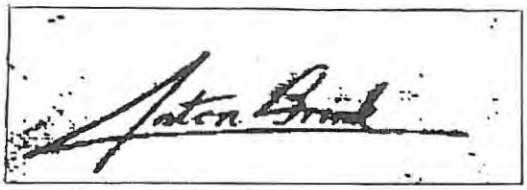
(a)



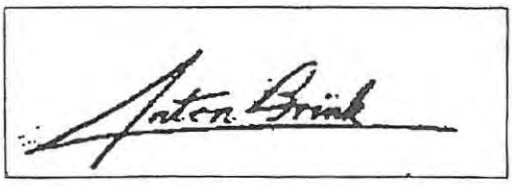
(b)



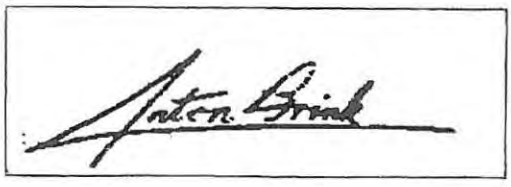
(c)



(d)



(e)



(f)

Fig. 5.5

Figure 5.6 **(Overleaf)** SIGN8: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram.

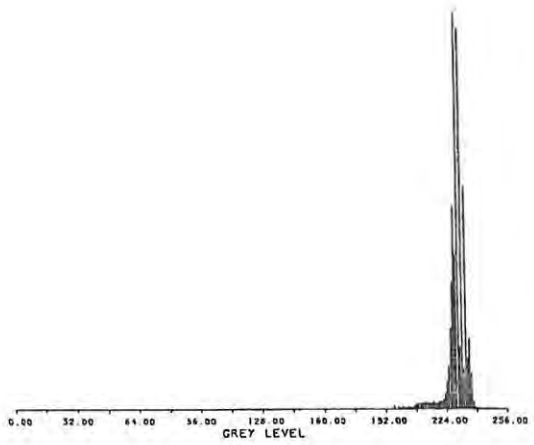
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, with changing threshold, T. The optimum threshold is the level that maximises $\Psi(T)$.

(c) The Minimum Error Method: a measure of average pixel classification error, $J(T)$, derived in Chapter 3.2.2, is plotted against threshold, T. The error is minimised by the optimum threshold.

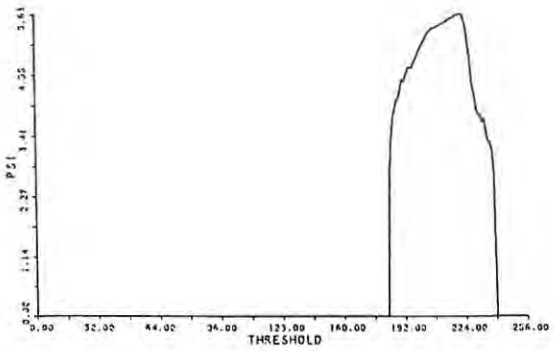
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original and bilevel images, $X(T)$, derived in Chapter 3.3.1, varies with changing threshold, T. The level where $X(T)$ is minimised is chosen as the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, with changing threshold, T. The threshold which yields the maximum correlation between the original and bilevel images is selected. The technique is described in Chapter 3.3.2.

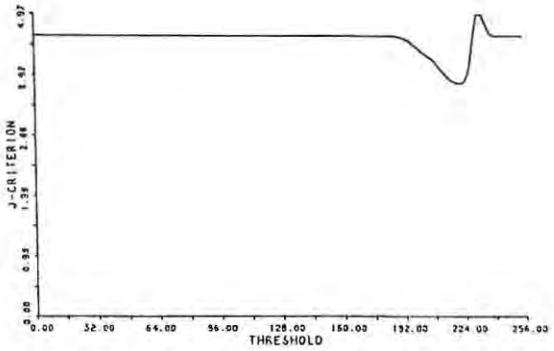
Although the histogram (Figure 5.6(a)) in this case displays little contrast, it would have been a simple matter to increase this. Such "grey-level expansion" would have had no effect on the relative thresholds selected by the various methods.


GREY LEVEL

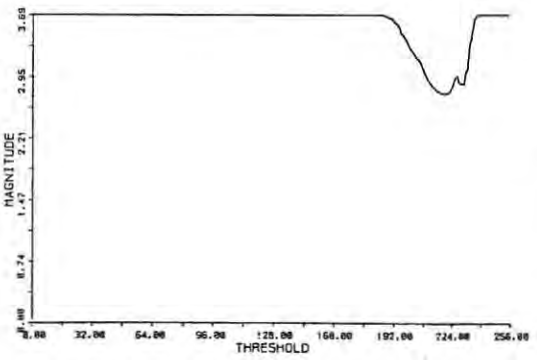
(a)


PSI
THRESHOLD

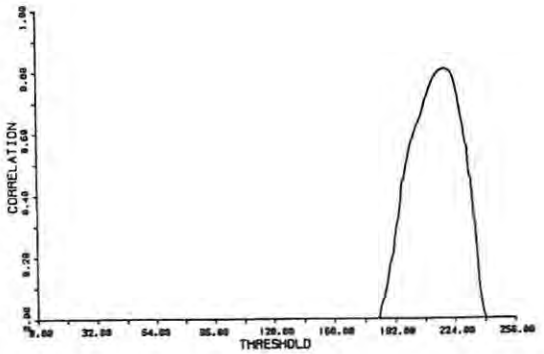
(b)


J-CRITERION
THRESHOLD

(c)


MAGNITUDE
THRESHOLD

(d)


CORRELATION
THRESHOLD

(e)

Fig. 5.6

Figure 5.7 **(Overleaf)** Thresholding methods applied to TYPE8, a section of a printed page. The image has a resolution of 128 x 128 pixels with 256 grey-levels.

(a) The original image

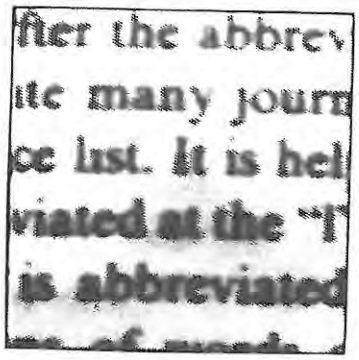
(b) The Entropy Method: threshold T = 194

(c) The Minimum Error Method: threshold T = 216

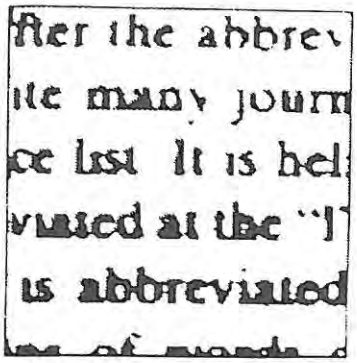
(d) The Moment-Preserving Method: threshold T = 202

(e) The Minimum Difference Method: threshold T = 206

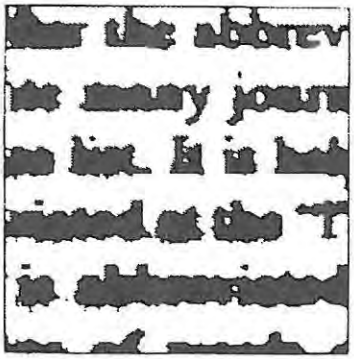
(f) The Maximum Correlation Method: threshold T = 203



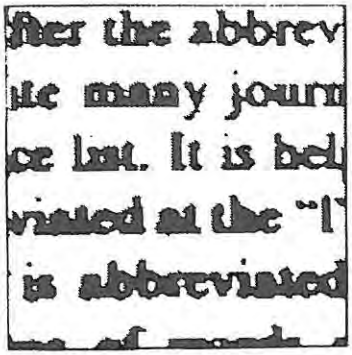
(a)



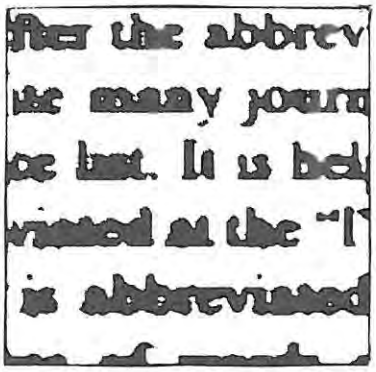
(b)



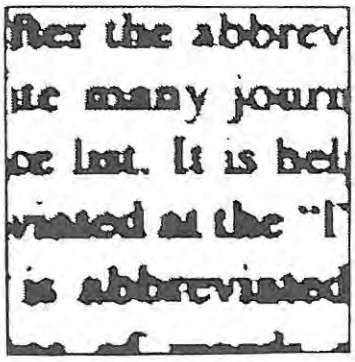
(c)



(d)



(e)



(f)

Fig. 5.7

Figure 5.8 **(Overleaf)** TYPE8: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram.

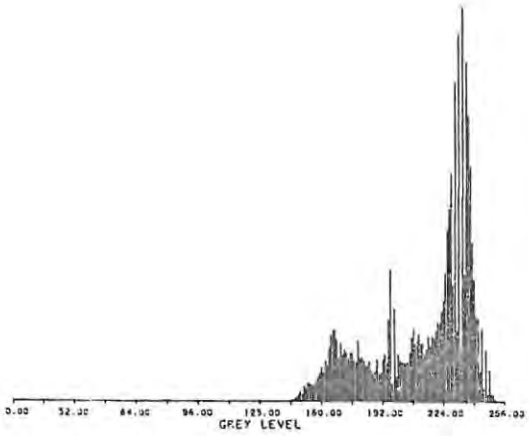
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, against threshold, T. The optimum threshold maximises $\Psi(T)$.

(c) The Minimum Error Method: the average pixel classification error, J(T), derived in Chapter 3.2.2, is plotted against threshold, T. The error is minimised by the optimum threshold.

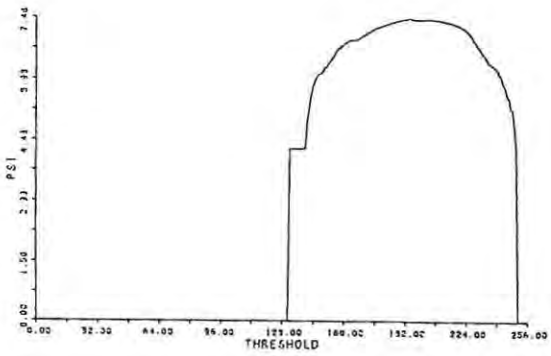
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original and bilevel images, X(T), derived in Chapter 3.3.1, is plotted against threshold, T. The level where X(T) is minimised is chosen as the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, described in Chapter 3.3.2, with changing threshold, T. The threshold that yields the maximum correlation between the original and bilevel images is selected.

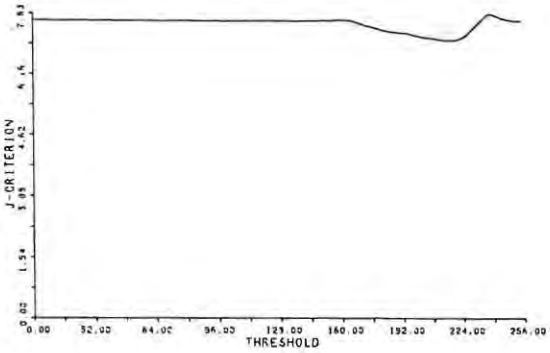
All techniques appear to work well here, with clear minima and maxima. This may be due to the relative clarity of the peaks in the histogram of Figure 5.8(a). However, inspection of the images of Figure 5.7 indicates that the methods do not work quite as well as expected. This is due to a "brightness gradient" caused by non-uniform illumination across the image.


GREY LEVEL

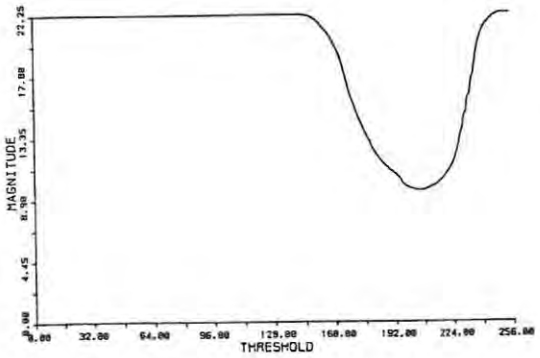
(a)


PSI
THRESHOLD

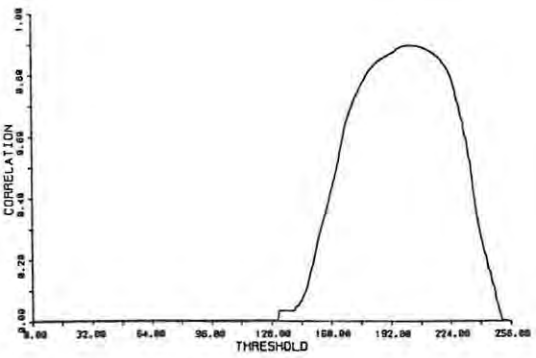
(b)


J-CRITERION
THRESHOLD

(c)


MAGNITUDE
THRESHOLD

(d)


CORRELATION
THRESHOLD

(e)

Fig. 5.8

Figure 5.9 **(Overleaf)** Thresholding methods applied to YBUG8, a picture of colonies of **Klebsiella pneumoniae** bacteria. The image has a resolution of 98 x 98 pixels with 256 grey-levels.

(a) The original image

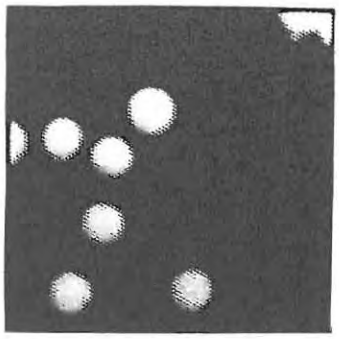
(b) The Entropy Method: threshold T = 99

(c) The Minimum Error Method: threshold T = 96

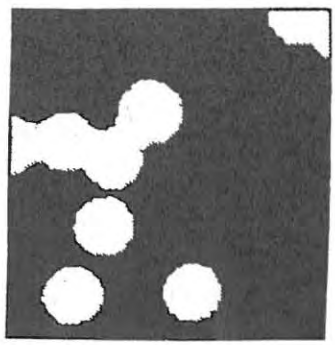
(d) The Moment-Preserving Method: threshold T = 131

(e) The Minimum Difference Method: threshold T = 132

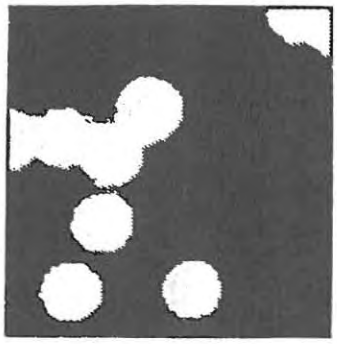
(f) The Maximum Correlation Method: threshold T = 134



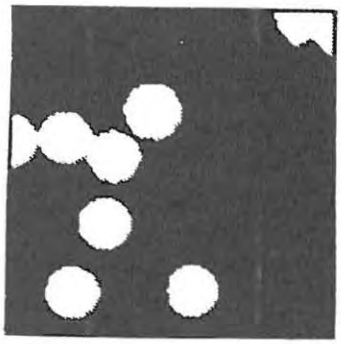
(a)



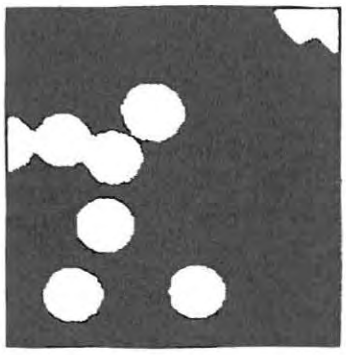
(b)



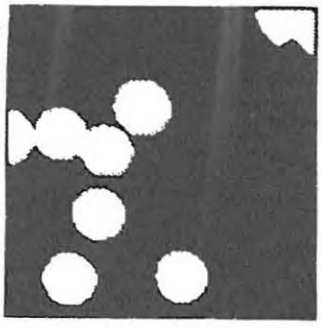
(c)



(d)



(e)



(f)

Fig. 5.9

Figure 5.10 **(Overleaf)** YBUG8: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram

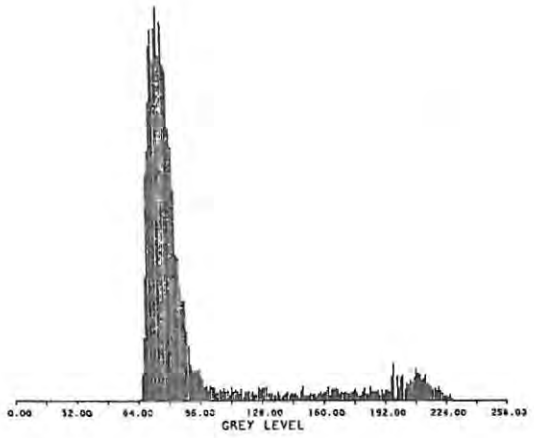
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, against threshold, T. The optimum threshold maximises $\Psi(T)$.

(c) The Minimum Error Method: the average pixel classification error, $J(T)$, derived in Chapter 3.2.2, is plotted against threshold, T. The error is minimised by the optimum threshold.

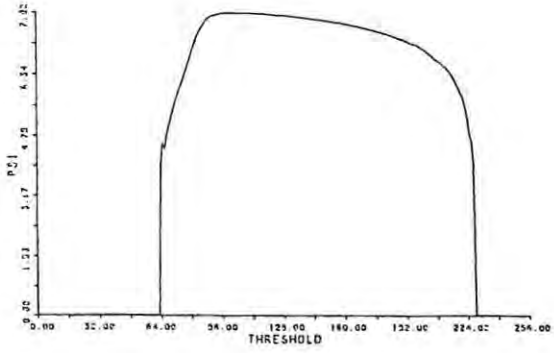
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original and bilevel images, $X(T)$, described in Chapter 3.3.1, is plotted against threshold, T. The level where $X(T)$ is minimised is the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, described in Chapter 3.3.2, with changing threshold, T. The threshold that yields the maximum correlation between the original and bilevel images is selected.

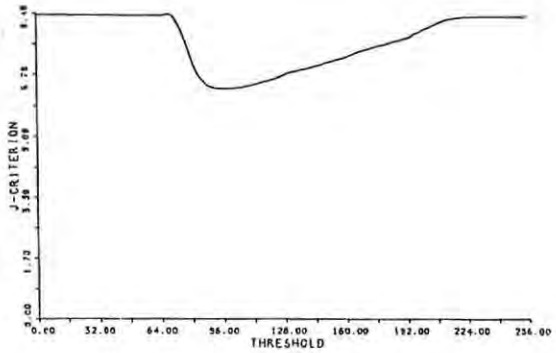
It is clear here that the criterion functions of the Entropy Method and the Minimum Error Method (Figure 5.10(b) and (c)) reached their "decision points" closer to the large peak of the histogram (Figure 5.10(a)), corresponding to the background pixels, than the small one, corresponding to the bacteria colonies. All the methods used here appear to have selected thresholds too low, as can be seen in Figure 5.9. The requirement here was to clearly seperate all the colonies from the background and from each other. None of the methods used fully satisfied this requirement.



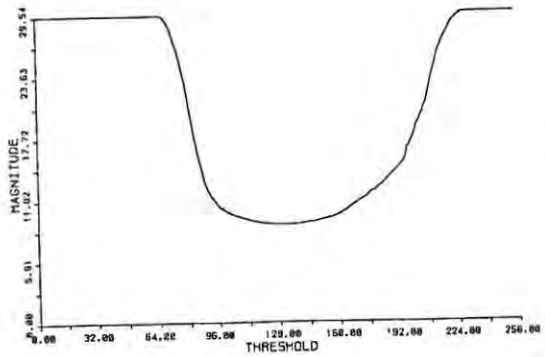
(a)



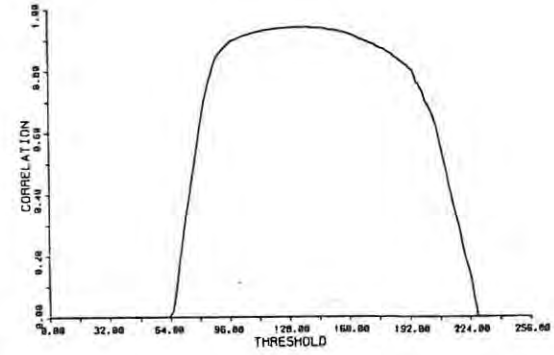
(b)



(c)



(d)



(e)

Fig. 5.10

Figure 5.11 **(Overleaf)** Thresholding methods applied to RBUG8, a picture of colonies of **Serratia marescens** bacteria. The image has a resolution of 98 x 98 pixels with 256 grey-levels.

(a) The original image

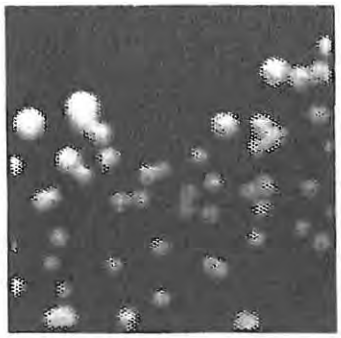
(b) The Entropy Method: threshold T = 133

(c) The Minimum Error Method: threshold T = 110

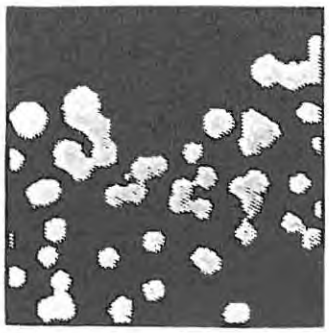
(d) The Moment-Preserving Method: threshold T = 129

(e) The Minimum Difference Method: threshold T = 124

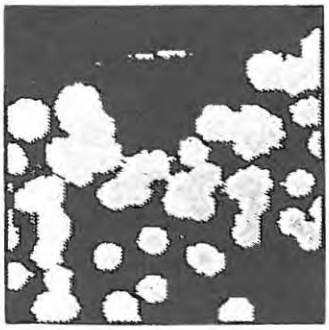
(f) The Maximum Correlation Method: threshold T = 129



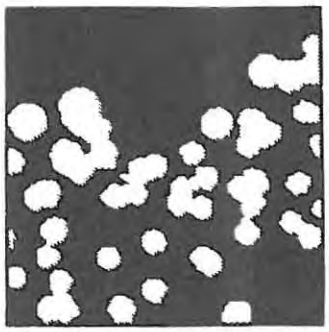
(a)



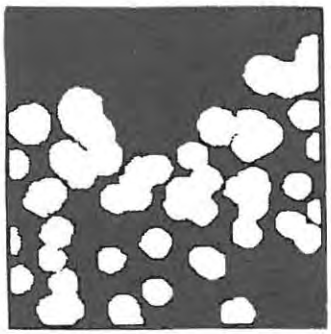
(b)



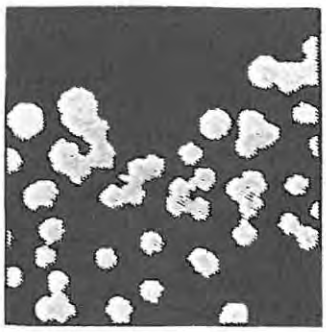
(c)



(d)



(e)



(f)

Fig. 5.11

Figure 5.12 **(Overleaf)** RBUG8: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram.

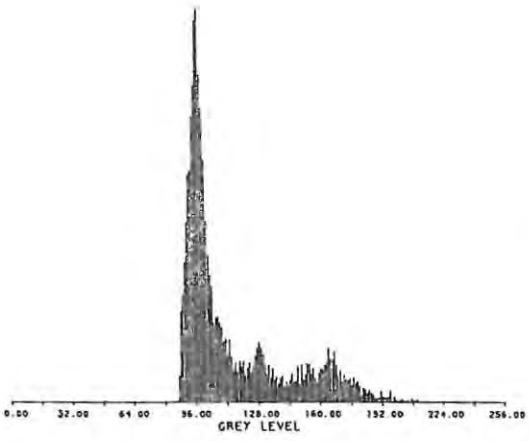
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, with changing threshold, T. The optimum threshold maximises $\Psi(T)$.

(c) The Minimum Error Method: a measure of average pixel classification error, $J(T)$, derived in Chapter 3.2.2, is plotted against threshold, T. The error is minimised by the optimum threshold.

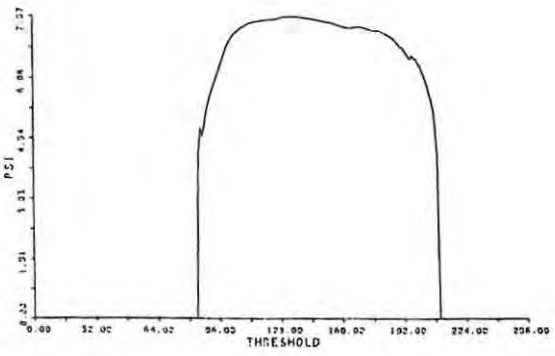
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original and bilevel images, $X(T)$, described in Chapter 3.3.1, is plotted against threshold, T. The sum is minimised by the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, defined in Chapter 3.3.2, with changing threshold, T. The threshold that yields the maximum correlation between the original and bilevel images is selected.

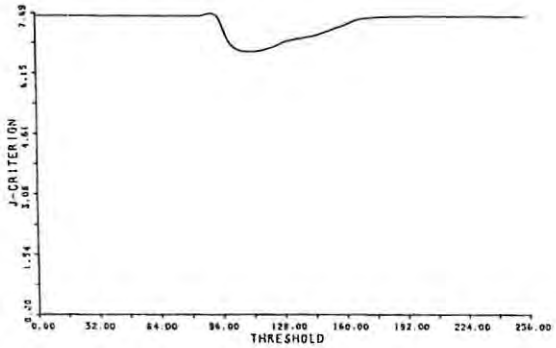
Again, as in Figure 5.9 and Figure 5.10, all methods selected thresholds that were too low to achieve complete extraction of the bacteria colonies from the background. In this case, the problem was compounded by a "brightness gradient" across the image, that is, the upper portion was better illuminated than the lower. This makes selection of a unique threshold difficult, at best, and impossible, at worst.



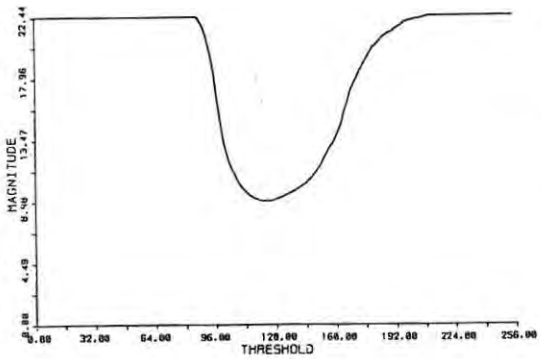
(a)



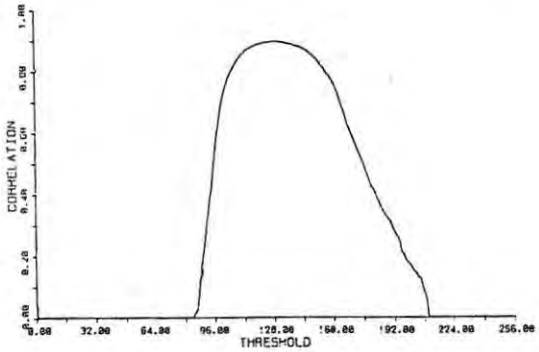
(b)



(c)



(d)



(e)

Fig. 5.12

Image	A	B	C	D	E
GIRL	3.30	0.00	3.70	1.20	1.80
PAN8	2.80	0.00	3.55	1.00	2.65
SIGN8	1.50	1.80	0.00	2.90	3.80
TYPE8	4.00	0.00	2.70	1.30	2.00
YBUG8	1.00	0.00	3.30	2.40	3.30
RBUG8	3.55	0.00	3.00	1.45	3.00
μ	2.69	0.30	2.71	1.71	2.76
σ	1.19	0.73	1.38	0.76	0.77

Table 5.1 The averaged results of applying the subjective evaluation technique of Chapter Four to the bilevel images obtained by thresholding grey-level images using the methods of Chapter Three. Column A: The Entropy Method; Column B: The Minimum Error Method; Column C: The Moment-Preserving Method; Column D: The Minimum Difference Method; Column E: The Maximum Correlation Method.

Table 5.1 contains the averaged results of the subjective evaluation technique of Chapter Four. Twenty people were asked to rate the results of thresholding each of the captured images, using the five methods of Chapter Three, on a scale of 0 to 4, where 0 was the "score" assigned to the result deemed the "worst" and 4 the "score" for the "best" result. The means and standard deviations of these results were calculated for analysis. It is clear from these that the Minimum Error method was overall the "worst", having a very low mean score. Although the Minimum Difference method has an average score, its standard deviation indicates a relatively consistent performance. The other three methods are all nearly equal in mean performance, with varying degrees of "unreliability" or standard deviation.

5.2 Thresholds Applied to Computer-Generated Images

The following pages contain the results of thresholding the test images, generated and degraded as described in Chapter Four, using the threshold selection techniques of Chapter Three. The image grey-level histograms and, where applicable, the graphs of the decision-criterion functions associated with some of the thresholding techniques are also presented.

In Table 5.2 the results of applying the quantitative correlation evaluation technique of Chapter Four to the thresholded images are presented, and the mean and standard deviation of the correlations for each threshold selection technique are calculated.

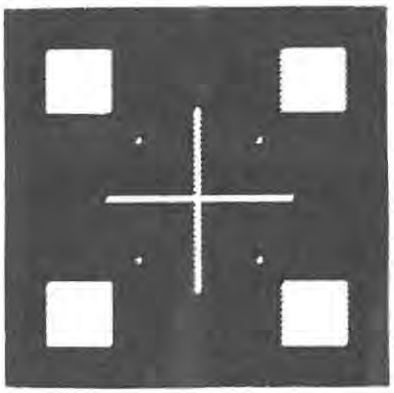


Figure 5.13 The original undegraded computer-generated bilevel test image, TESTPIC, of Chapter Four. The image has two levels, black (Ø) and white (255) and has a resolution of 128 x 128 pixels.

The quantitative evaluation method of Chapter Four involves the calculation of the correlation between the image of Figure 5.13 and each of the bilevel results obtained by thresholding the degraded versions of TESTPIC using the methods of Chapter Three. These correlations are presented in Table 5.2. The histogram of this image would consist merely of two columns, one at grey-level 0, representing the "background" pixels, and the other at grey-level 255, representing the "objects". The column at 0 would be much taller than that at 255, as the "background" makes up the major portion of the image. Reducing the contrast, as described in Chapter Four, moves these columns closer together, narrowing the valley between them. Since blurring and the addition of noise both result in the creation of intermediate grey-levels, more histogram columns will appear, spreading the two modes into roughly Gaussian distributions. This is demonstrated in the results presented on the following pages.

Figure 5.14 **(Overleaf)** Thresholding methods applied to WPIC22, a test image degraded by blurring, using an optical transfer function with $f_c = \frac{1}{2}$ cycles/pixel, and the addition of Gaussian noise with standard deviation $\sigma = 2$. The image has a resolution of 128 x 128 pixels with 256 grey-levels.

(a) The original degraded image

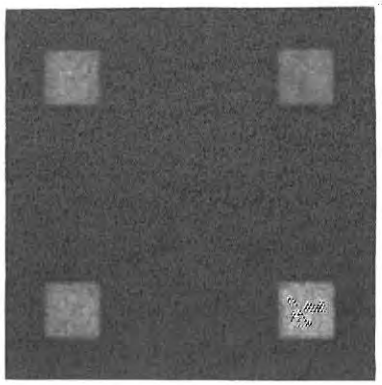
(b) The Entropy Method: threshold T = 71

(c) The Minimum Error Method: threshold T = 72

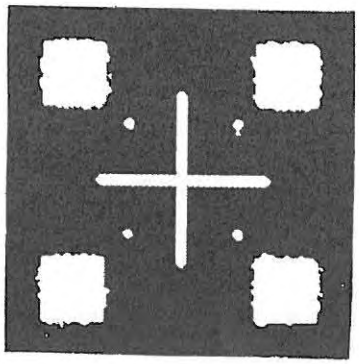
(d) The Moment-Preserving Method: threshold T = 113

(e) The Minimum Difference Method: threshold T = 122

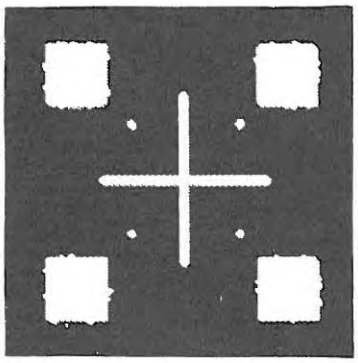
(f) The Maximum Correlation Method: threshold T = 120



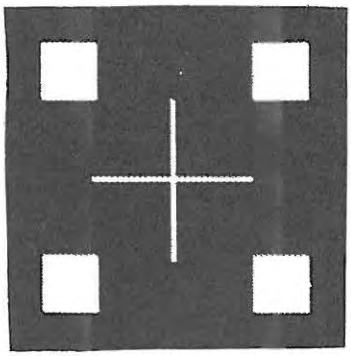
(a)



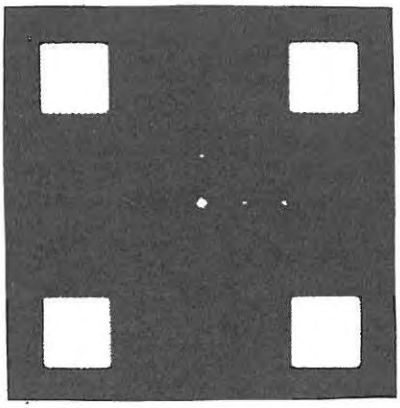
(b)



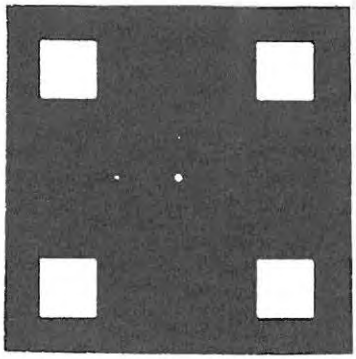
(c)



(d)



(e)



(f)

Fig. 5.14

Figure 5.15 **(Overleaf)** WPIC22: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram

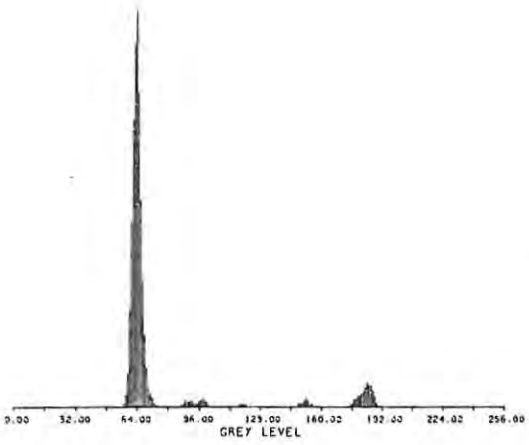
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, with changing threshold, T. The optimum threshold maximises $\Psi(T)$.

(c) The Minimum Error Method: a measure of average pixel classification error, J(T), derived in Chapter 3.2.2, is plotted against threshold, T. The error is minimised by the optimum threshold.

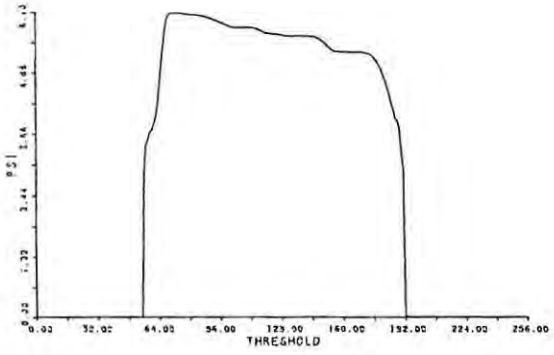
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the grey-level and bilevel images, X(T), described in Chapter 3.3.1, is plotted against threshold, T. The sum is minimised by the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, defined in Chapter 3.3.2, with changing threshold, T. The threshold that yields the maximum correlation between the grey-level and bilevel images is selected.

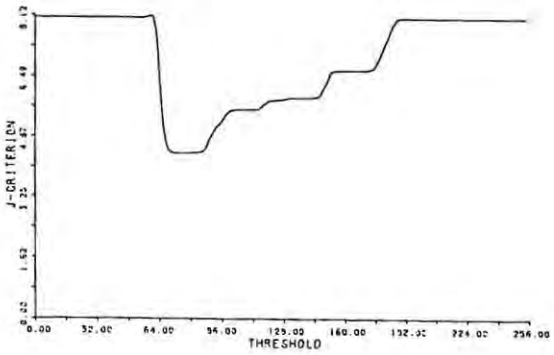
All of the criterion functions plotted in Figure 5.15 have been distorted by the broad, largely featureless valley lying between the two main modes of the histogram. These distortions take the form of broad, flat regions of minimum or maximum function value, which in turn leads to some ambiguity in threshold selection. The least affected was the Entropy Method (Figure 5.15(b)).



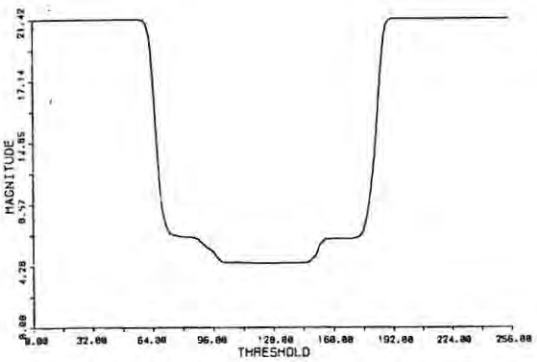
(a)



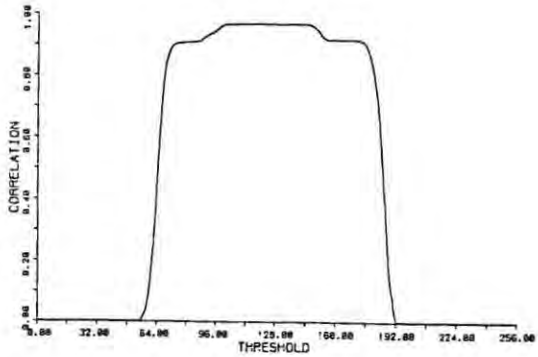
(b)



(c)



(d)



(e)

Fig. 5.15

Figure 5.16 **(Overleaf)** Thresholding methods applied to WPIC24, a test image degraded by blurring, using an optical transfer function with $f_c = \frac{1}{2}$ cycles/pixel, and the addition of Gaussian noise with standard deviation $\sigma = 4$. The image has a resolution of 128 x 128 pixels with 256 grey-levels.

(a) The original degraded image

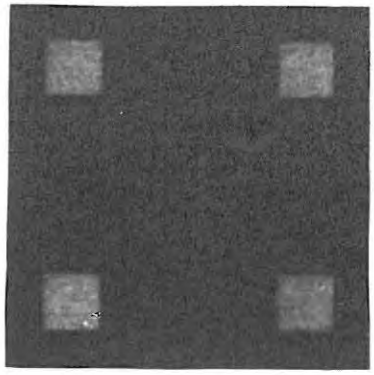
(b) The Entropy Method: threshold T = 74

(c) The Minimum Error Method: threshold T = 78

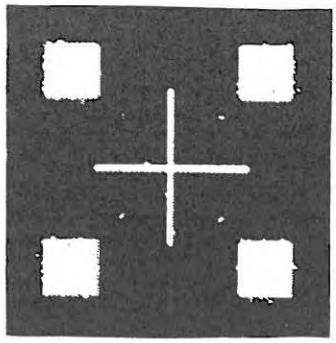
(d) The Moment-Preserving Method: threshold T = 101

(e) The Minimum Difference Method: threshold T = 125

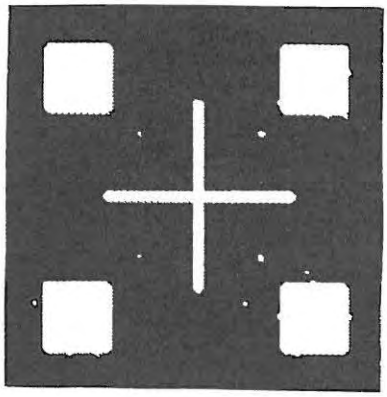
(f) The Maximum Correlation Method: threshold T = 120



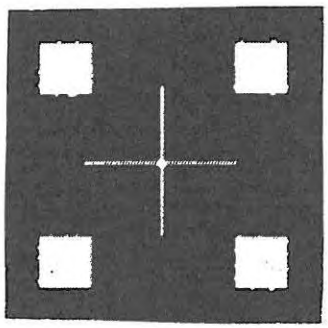
(a)



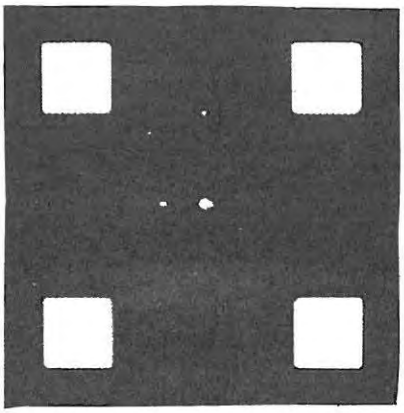
(b)



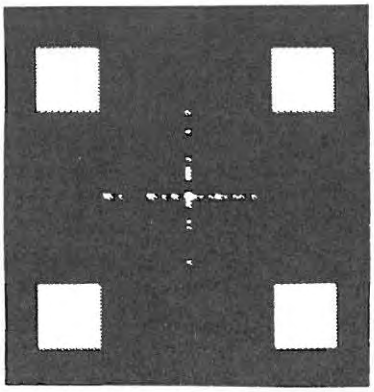
(c)



(d)



(e)



(f)

Fig. 5.16

Figure 5.17 **(Overleaf)** WPIC24: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram.

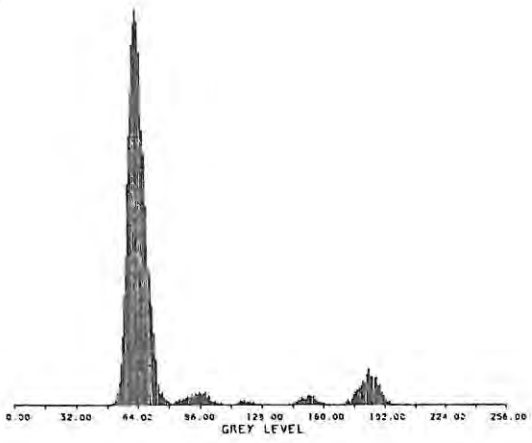
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, with changing threshold T. The optimum threshold maximises $\Psi(T)$.

(c) The Minimum Error Method: a measure of average pixel classification error, J(T), is plotted against threshold, T. The error is minimised by the optimum threshold. The method is described in Chapter 3.2.2.

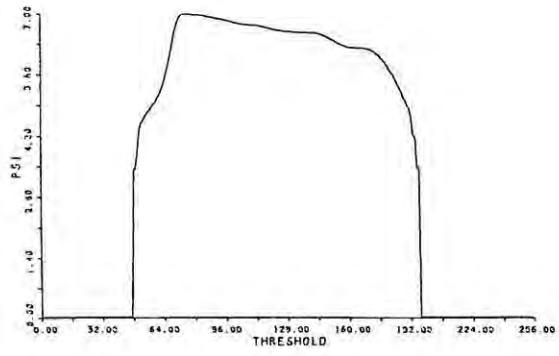
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original degraded and the bilevel images, X(T), described in Chapter 3.3.1 is plotted against threshold, T. The sum is minimised by the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, defined in Chapter 3.3.2, with changing threshold, T. The threshold that yields the maximum correlation between the grey-level and bilevel images is selected.

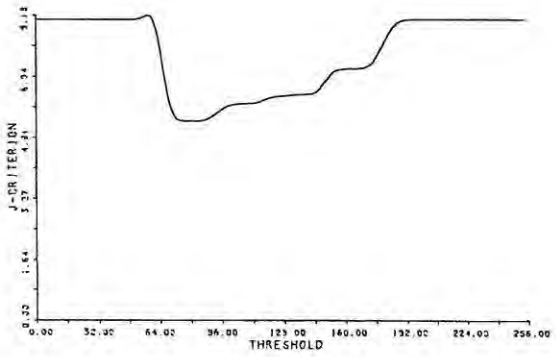
Due to the addition of more noise, the valley between the histogram peaks is now less "featureless" and, as a consequence, the selection techniques select more meaningful thresholds. The Minimum Error (Figure 5.17(c)), Minimum Difference (Figure 5.17(d)) and Maximum Correlation (FIgure 5.17(e)) methods are still ambiguous in their selection.



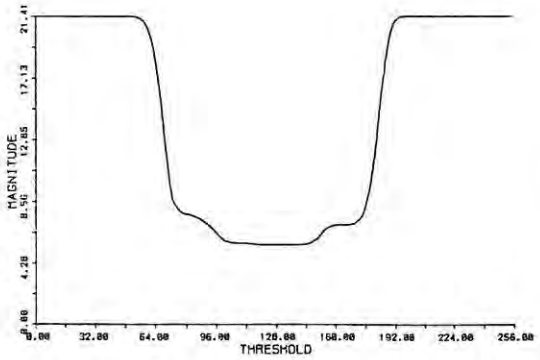
(a)



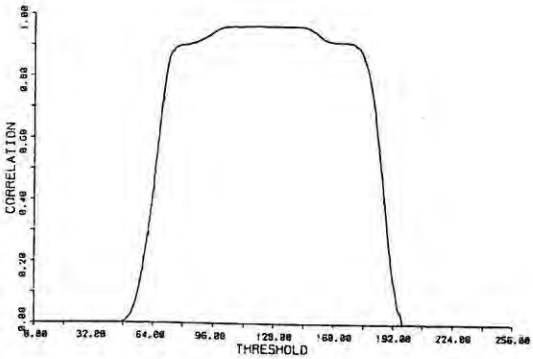
(b)



(c)



(d)



(e)

Fig. 5.17

Figure 5.18 **(Overleaf)** Thresholding methods applied to WPIC42, a test image degraded by blurring, using an optical transfer function with $f_c = \frac{1}{4}$ cycles/pixel, and the addition of Gaussian noise with standard deviation $\sigma = 2$. The image has a resolution of 128 x 128 pixels with 256 grey-levels.

(a) The original degraded image.

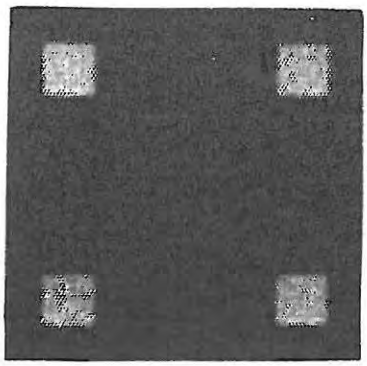
(b) The Entropy Method: threshold T = 89

(c) The Minimum Error Method: threshold T = 72

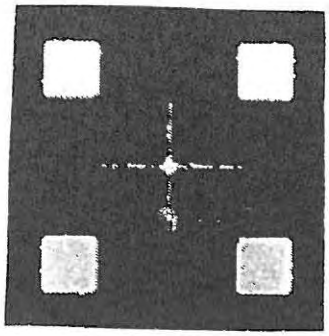
(d) The Moment-Preserving Method: threshold T = 109

(e) The Minimum Difference Method: threshold T = 113

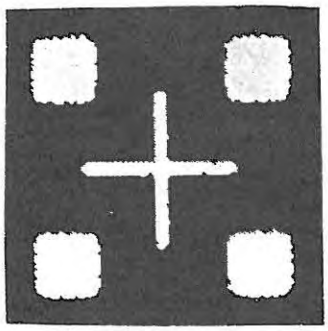
(f) The Maximum Correlation Method: threshold T = 115



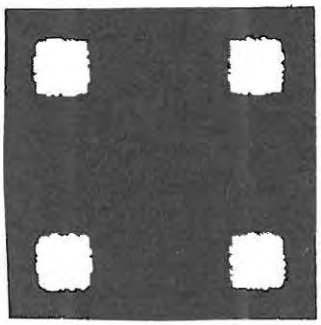
(a)



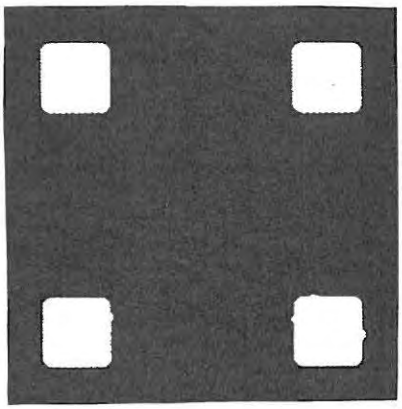
(b)



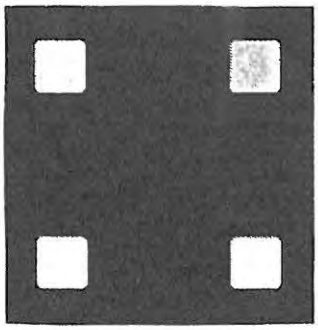
(c)



(d)



(e)



(f)

Fig. 5.18

Figure 5.19 **(Overleaf)** WPIC42: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram.

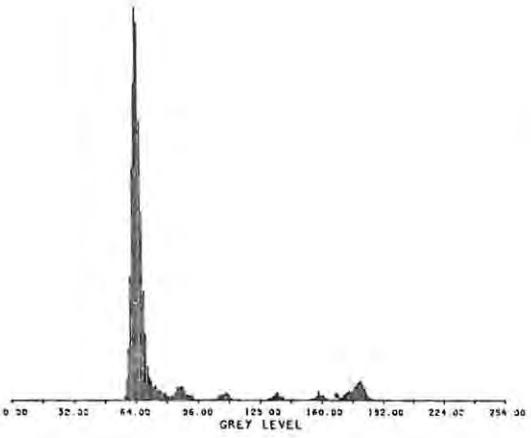
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, derived in Chapter 3.2.1, with changing theshold, T. The optimum threshold maximises $\Psi(T)$.

(c) The Minimum Error Method: a measure of average pixel classification error, J(T), derived in Chapter 3.2.2, is plotted against threshold, T. The error is minimised by the optimum threshold.

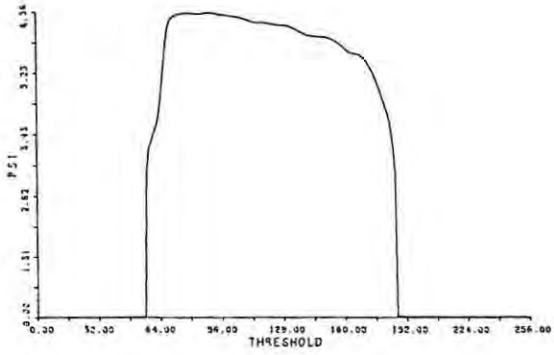
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the original degraded and bilevel images, X(T), described in Chapter 3.3.1, is plotted against threshold, T. The sum is minimised by the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, defined in Chapter 3.3.2, with changing threshold, T. The threshold that yields the maximum correlation between the grey-level and bilevel images is selected.

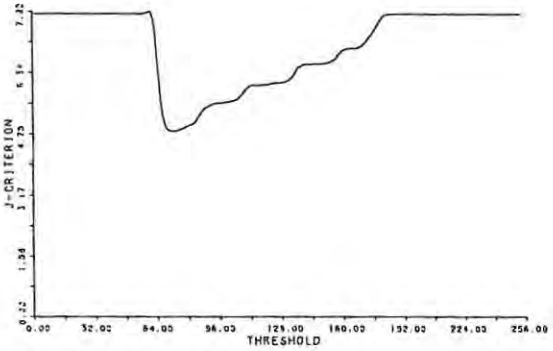
The Minimum Difference (Figure 5.19(d)) and Maximum Correlation (Figure 5.19(e)) methods remain ambiguous in their choice of threshold, though their graphs have been rounded a great deal by the further smoothing of the image.



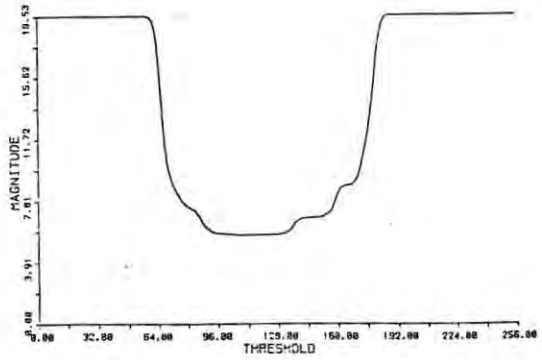
(a)



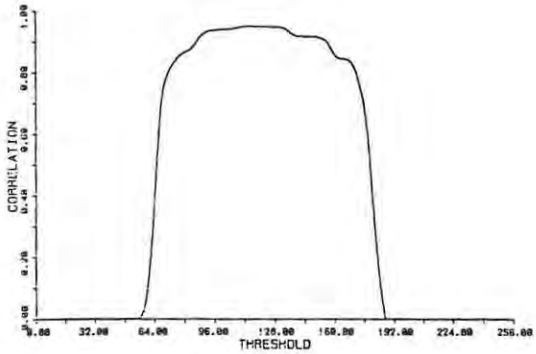
(b)



(c)



(d)



(e)

Fig. 5.19

Figure 5.20 **(Overleaf)** Thresholding methods applied to WPIC44, a test image degraded by blurring, using an optical transfer function with $f_c = \frac{1}{4}$ cycles/pixel, and the addition of Gaussian noise with standard deviation σ = 4. The image has a resolution of 128 x 128 pixels with 256 grey-levels.

(a) The original degraded image.

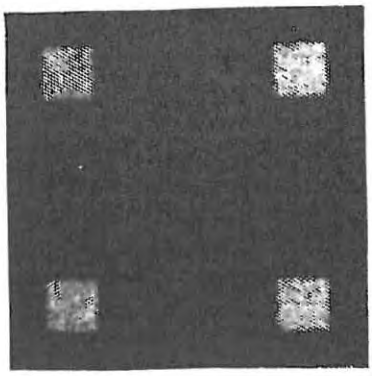
(b) The Entropy Method: threshold T = 89

(c) The Minimum Error Method: threshold T = 78

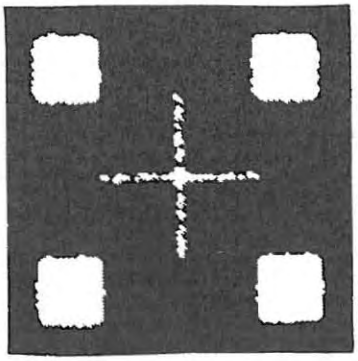
(d) The Moment-Preserving Method: threshold T = 107

(e) The Minimum Difference Method: threshold T = 117

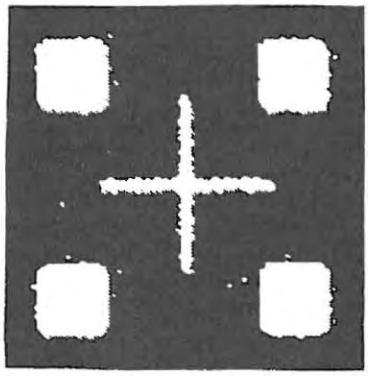
(f) The Maximum Correlation Method: threshold T = 116



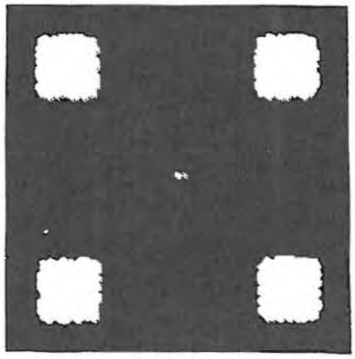
(a)



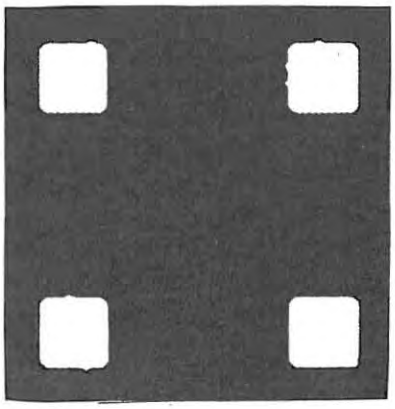
(b)



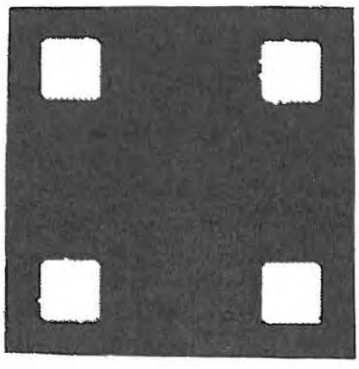
(c)



(d)



(e)



(f)

Fig. 5.20

Figure 5.21 **(Overleaf)** WPIC44: The grey-level histogram and graphs related to thresholding methods.

(a) The grey-level histogram.

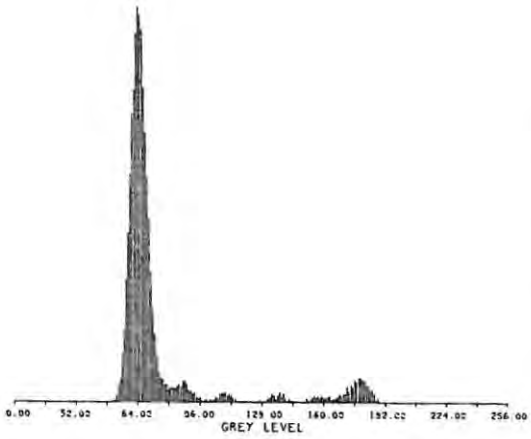
(b) The Entropy Method: the graph shows the variation of the criterion function, $\Psi(T)$, defined in Chapter 3.2.1, with changing threshold, T. The optimum threshold maximises $\Psi(T)$.

(c) The Minimum Error Method: a measure of average pixel classification error, J(T), derived in Chapter 3.2.2, is plotted against threshold, T, The error is minimised by the optimum threshold.

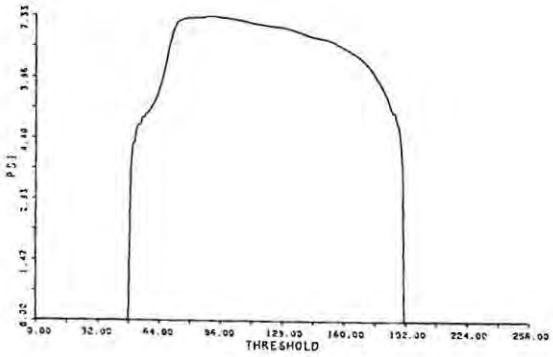
(d) The Minimum Difference Method: the sum of the magnitudes of the differences between the grey-level and bilevel images, X(T), is plotted against threshold, T. The sum is minimised by the optimum threshold.

(e) The Maximum Correlation Method: the graph shows the variation of the correlation function, $\rho_{xy}(T)$, defined in Chapter 3.3.2, with changing threshold, T. The threshold that yields the maximum correlation between the grey-level and bilevel images is selected.

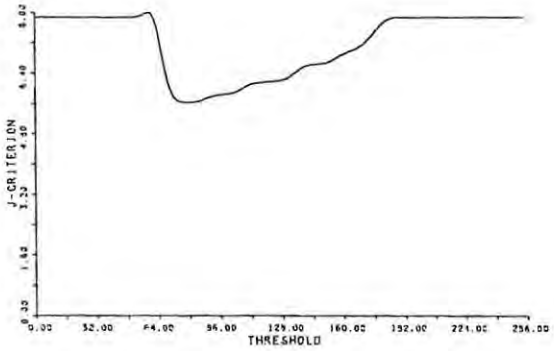
The increase in smoothing and noise applied to the image has led to a histogram (Figure 5.21(a)) with a more complete spread over the grey-levels between the peaks. This has smoothed the graphs of the criterion functions, allowing unique maxima and minima corresponding to the optimum thresholds to be found.



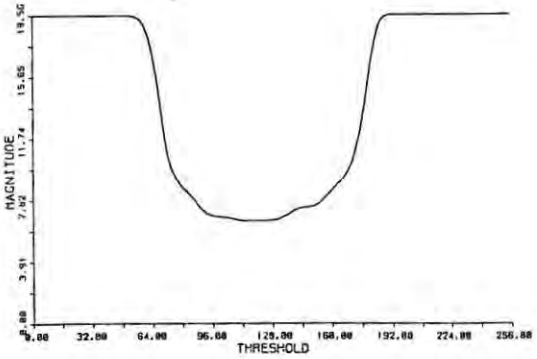
(a)



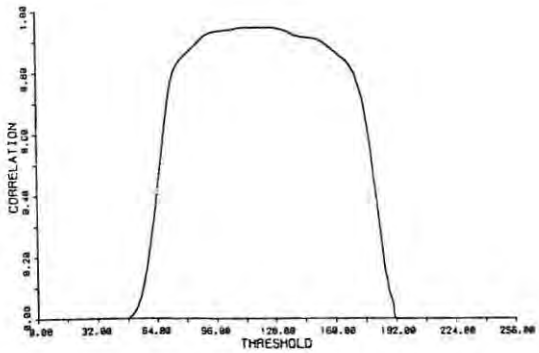
(b)



(c)



(d)



(e)

Fig. 5.21

Image	SNR (dB)	A	B	C	D	E
WPIC22	32.3464	0.8231	0.8361	0.9974	0.9626	0.9639
WPIC24	26.4239	0.8242	0.8492	0.9859	0.9619	0.9715
WPIC42	32.2272	0.8832	0.7156	0.9166	0.9526	0.9543
WPIC44	26.3046	0.8665	0.7684	0.9080	0.9522	0.9513
μ	-	0.8493	0.7923	0.9520	0.9573	0.9603
σ	-	0.0303	0.0622	0.0462	0.0057	0.0092

Table 5.2 The results of applying the quantitative evaluation method of Chapter Four to the bilevel images obtained by thresholding computer-generated grey-level images using the methods of Chapter Three. The column labelled "SNR" contains the signal-to-noise ratios, in dB's, of the grey-level images. These were calculated in Chapter Four. Column A: The Entropy Method; Column B: The Minimum Error Method; Column C: The Moment-Preserving Method; Column D: The Minimum Difference Method; Column E: The Maximum Correlation Method.

From inspection of this table, it is clear that the Moment-Preserving technique works best with a high signal-to-noise ratio, while for a low signal-to-noise ratio the Minimum Difference and Maximum Correlation methods are best. Since the Moment-Preserving Method does not depend on the histogram to the same extent that the other methods do, it was not affected in a similar way by the wide, flat histogram valley. However, it was adversely affected by the decrease in signal-to-noise ratio. The Maximum Correlation method, on the other hand, was able to make less ambiguous selections of threshold with increasing image degradation, and this improvement to some extent lessened the adverse effects of decreasing signal-to-noise ratio. These results are discussed in detail in Chapter Six.

CHAPTER SIX

DISCUSSION AND CONCLUSIONS

In the previous chapter, the results obtained with the five threshold selection techniques, described in Chapter Three, are presented. The results of evaluating the thresholded images, using the two threshold evaluation methods of Chapter Four, are also tabulated. The results obtained with the threshold selection techniques are discussed in Section 6.1, threshold evaluation techniques are discussed in Section 6.2, and Section 6.3 contains the conclusions.

6.1 Threshold Selection Techniques

All the images used in this project have approximately bimodal grey-level histograms, except GIRL (Fig. 5.2(a)), which has a multimodal distribution with six peaks, and SIGN8 (Fig. 5.6(a)), which as a unimodal distribution with a small peak, or shoulder, on one side of the mode. In all cases, except GIRL, the larger peak represents the "background" pixels and the smaller peak represents the "object" pixels. In GIRL, both "background" and "object" have a wide range of grey-levels, making threshold selection for all but artistic purposes very difficult. It is, however, a useful test image.

From inspection of the criterion function graphs in the previous chapter, it appears that the Minimum Error method selects thresholds close to the larger peak in a bimodal histogram. This leads to the inclusion of some unwanted background noise and blur in the "object" regions.

A further drawback of the Minimum Error method is its dependence on the bimodality of the histogram. In the case of GIRL, where the histogram has several modes, the criterion function has several minima and, although the threshold is selected where the "lowest" minimum occurs, the selection is not good.

The Entropy, Minimum Difference and Maximum Correlation methods generally select thresholds in the centre of the valley between the peaks of a bimodal histogram, although the Entropy method shows a slight bias towards the larger mode. Such "centred" thresholds generally result in good bilevel images. The Moment-Preserving method also appears to select a centred threshold, usually yielding a value close to that selected by the Maximum Correlation method.

For each image, the application of the five different threshold selection techniques generally yields five different "optimum" thresholds. Since there can be only one optimum threshold for a given image and application, these results were evaluated using the two methods described in Chapter Four in order to find the best threshold selection technique.

6.2 Threshold Evaluation Techniques

Inspection of the means of the evaluations presented in Table 5.1 reveals that, as expected, the four techniques that selected thresholds near the centre of the histogram valley yielded generally better results than the one that was biased towards the larger histogram peak. As shown by the standard deviations, they were also

generally more consistent in their performance.

In performing these subjective evaluations, it soon became apparent that a description of the required bilevel image, or the envisaged application, is needed in order to evaluate the results. Here the evaluations in each case were based on the similarity, or lack thereof, between the thresholded images and the original. In many cases these evaluations would no longer be valid if the envisaged application is altered.

Inspection of the results of the quantitative evaluation technique, presented in Table 5.2, reveals that, although the individual order of "goodness" has changed, the four techniques that selected "centred" thresholds are still better than the one that was biased towards the larger histogram peak. All techniques were affected to some extent by the different amounts of degradation applied to the images, the Minimum Difference and Maximum Correlation methods being the least affected. In all cases, an increase in blurring had a greater effect than an increase in the noise.

Although the quantitative evaluation method uses computer-generated images on which to base its evaluation of thresholds, it is still dependent on the application that is envisaged. In this case, the aim of the thresholding process was to restore the original, undegraded bilevel image. Again, if the application were altered, these evaluations may no longer be valid.

6.3 Conclusions

It has been noted that each threshold selection technique works best with a different type of image and/or application. There is no universal "best" method, although some, particularly the Maximum Correlation and the Moment-Preserving methods, are very consistent in their performance and generally select good, if not optimum, thresholds for most applications.

This dependence on the application is also apparent in threshold evaluation techniques. Until the application, or a general requirement, can be stated quantitatively, the subjective evaluation technique remains the more versatile, though less accurate, method.

REFERENCES

Andrews, H.C. and Hunt, B.R. (1977). Digital Image Restoration, Prentice-Hall, Englewood Cliffs, New Jersey.

Bracewell, R.N. (1978). The Fourier Transform and Its Applications, 2nd edition, McGraw-Hill, Tokyo.

Ballard, D.H. and Brown, C.M. (1982). Computer Vision, Prentice-Hall, Englewood Cliffs, New Jersey.

Castleman, K.R. (1979). Digital Image Processing, Prentice-Hall, Englewood Cliffs, New Jersey.

Freund, J.E. (1979). Modern Elementary Statistics, 5th edition, Prentice-Hall, Englewood Cliffs, New Jersey.

Gonzalez, R.C. and Wintz, P. (1977). Digital Image Processing, Addison-Wesley Publishing Co., Reading, Mass.

Hall, E.L. (1979) Computer Image Processing and Recognition, Academic Press, New York.

Horn, B.K.P. (1986). Robot Vision, The MIT Press, Cambridge, Mass.

Kapur, J.N., Sahoo, P.K. and Wong, A.K.C. (1985). "A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram", Comput. Vision, Graphics, Image Process., vol. 29, pp. 273-285.

Kirby, R.L. and Rosenfeld, A. (1979). "A Note on the Use of (Gray Level, Local Average Gray Level) Space as an Aid in Threshold Selection", IEEE Trans. Syst., Man, Cybern., vol. SMC-9, pp. 860-864.

Kittler, J. and Illingworth, J. (1986). "Minimum Error Thresholding", Pattern Recognition, vol. 19, pp. 41-47.

Kittler, J., Illingworth, J. and Föglein, J. (1985). "Threshold Selection Based on a Simple Image Statistic", Comput. Vision, Graphics, Image Process., vol. 30, pp. 125-147.

Kohler, R. (1981). "A Segmentation System Based on Thresholding", Comput. Graphics Image Process., vol. 15, pp. 319-338.

Oppenheim, A.V. and Schafer, R.W. (1975). Digital Signal Processing, Prentice-Hall, Englewood Cliffs, New Jersey.

Otsu, N. (1978). "Discriminant and Least Squares Threshold Selection", 4th Int. Joint Conf. on Pattern Recognition, Kyoto, Japan, pp. 592-596.

Otsu, N. (1979). "A Threshold Selection Method from Gray-Level Histograms", IEEE Trans. Syst., Man, Cybern., vol. SMC-9, pp. 62-66.

Peleg, S. (1978). "Iterative Histogram Modification, 2", IEEE Trans. Syst., Man, Cybern., vol. SMC-8, pp. 555-556.

Pratt, W.K. (1978). Digital Image Processing, John Wiley & Sons, New York.

Pun, T. (1980). "A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram", Signal Process., vol. 2, pp. 223-237.

Pun, T. (1981). "Entropic Thresholding : A New Approach", Comput. Graphics Image Process., vol. 16, pp. 210-239.

Rosenfeld, A. and Davis, L.S. (1978). "Iterative Histogram Modification", IEEE Trans. Syst., Man, Cybern., vol. SMC-8, pp. 300-302.

Rosenfeld, A. and Kak, A.C. (1976). Digital Picture Processing, Academic Press, New York.

Tsai, Wen-Hsiang (1985). "Moment-Preserving Thresholding : A New Approach", Comput. Vision, Graphics, Image Process., vol. 29, pp. 377-393.

Weszka, J.S. (1978). "A Survey of Threshold Selection Techniques", Comput. Graphics Image Process., vol. 7, pp. 259-265.

Weszka, J.S., Nagel, R.N. and Rosenfeld, A. (1974). "A Threshold Selection Technique", IEEE Trans. Computers, vol. C-23, pp. 1322-1326.

Weszka, J.S. and Rosenfeld, A. (1978). "Threshold Evaluation Techniques", IEEE Trans. Syst., Man, Cybern., vol. SMC-8, pp. 622-629.

Weszka, J.S. and Rosenfeld, A. (1979). "Histogram Modification for Threshold Selection", IEEE Trans. Syst., Man, Cybern., vol. SMC-9, pp. 38-52.

Zelen, M. and Severo, N.C. (1964). "Probability Functions", in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, (M. Abramowitz and I.A. Stegun, eds.), National Bureau of Standards, United States Department of Commerce.

APPENDIX A

PROGRAM LISTINGS

CONTENTS

VAR6BIT.BAS

```
10 DEF SEG = &H7000
20 BDFRAME = &H12E
30 BOFFSET = &H13F
40 BTHROTL = &H17A
50 BXDMA = &H175
60 BYDMA = &H419
70 BSETMOD = &H1A0
80 BADDR = &H1D8
90 SCALE = &H20C
100 BSETCRT = &H224
110 BSETREG = &H4A4
120 BMMIDISP = &H2DC
130 BEYEWRIT = &H29D
140 BITPEL = &HAA
150 DPBYT = &H9B
160 PUMP = &HCD
161 INPUT "FRAME WIDTH:";W%
162 INPUT "FRAME HEIGHT:";V%
163 L% = (640 - W%)/2
164 T% = (400 - V%)/2
170 BLOAD "GRINT.LOD",0
180 DEF SEG = &H71E1
190 PBASE% = &H310
200 GBASE% = &HCD0
210 DMEM% = &HA000
220 NMEM% = &H2000
225 IERR% = 0
230 CALL BADDR (PBASE%,GBASE%,DMEM%,NMEM%,IERR%)
240 IF IERR%<>0 GOTO 770
250 INTER% = 0
260 XTRO% = 0
270 IOMODE% = 0
280 MEMTYP% = 5
290 IERR% = 0
300 CALL BSETMOD (INTER%,XTRO%,IOMODE%,MEMTYP%,IERR%)
310 IF IERR%<>0 GOTO 770
320 BLACK% = 10
330 WHITE% = 30
340 IERR% = 0
350 CALL BSCALE (BLACK%,WHITE%,IERR%)
360 IF IERR%<>0 GOTO 770
370 HOFF% = 40
380 VOFF% = 12
385 IERR% = 0
390 CALL BOFFSET (HOFF%,VOFF%,IERR%)
400 IF IERR%<>0 GOTO 770
410 LEFT% = L%
420 HPEL% = H%
430 TOP% = T%
440 VPEL% = V%
445 IERR% = 0
450 CALL BDFRAME (LEFT%,HPEL%,TOP%,VPEL%,IERR%)
460 IF IERR%<>0 GOTO 770
470 TRTL% = 14
475 IERR% = 0
480 CALL BTHROTL (TRTL%,IERR%)
490 IF IERR%<>0 GOTO 770
500 IOFF% = 1
510 ICHAN% = (4-1)*16
515 IERR% = 0
520 CALL BSETREG (IOFF%,ICHAN%,IERR%)
530 IF IERR% <> 0 GOTO 770
540 DISPMOD% = &HF
550 CALL BSETCRT (DISPMOD%)
570 UNPACK% = 0
575 IERR% = 0
580 CALL BMDMA (UNPACK%,IERR%)
590 IF IERR% <> 0 GOTO 770
600 CALL BMMIDISP (UNPACK%)
610 CH$ = INKEY$
620 IF CH$ = "P" GOTO 180
630 IF CH$ = "S" GOTO 650
640 IF CH$ <> CHR$(27) GOTO 610
650 IF CH$ <> "S" GOTO 750
660 TYPE% = 3:FILE$ = "B:JUNK.PIC"
670 DEF SEG = &H7000
680 POKE BITPEL, 0
690 POKE DPBYT, 1
700 POKE PUMP, 0
710 DEF SEG = &H71E1
715 IERR% = 0
720 CALL BEYEWRIT (FILE$,TYPE%,IERR%)
730 IF IERR% <> 0 GOTO 770
740 DISPMOD% = 0
750 CALL BSETCRT (DISPMOD%)
760 STOP
770 DISPMOD% = &H0
780 CALL BSETCRT (DISPMOD%)
790 PRINT IERR%
800 STOP
```

PCEYE2.FOR

```
      PROGRAM PCEYE
C     =============
C
C     Program to convert pc-eye format data to starlink bdf.
C
C     Original written by J.L.Jonas
C     Modified by A.D.Brink,01/11/1986
C
      CHARACTER FNAME*16
      INTEGER POINTER, SIZE(2)
      INTEGER*2 IWORD
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
        CALL RDKEYC ('INPUT',,FALSE.,1,FNAME,JUNK,ISTAT)
        OPEN (UNIT=10,FORM='UNFORMATTED',RECORDTYPE='FIXED',
     *        NAME=FNAME,STATUS='OLD')
        SIZE(1) = 640
        SIZE(2) = 400
        CALL RDKEYI ('SIZE',,TRUE.,2,SIZE,JUNK,ISTAT)
        CALL WRIMAG ('OUTPUT',FMT_UB,SIZE,2,POINTER,ISTAT)
        K = 0
        DO I = 1 , 128
          READ (10) IWORD
        END DO
        CALL READ_AND_COPY (%VAL(POINTER),SIZE(1),SIZE(2))
        CALL EXIT
      END
      SUBROUTINE READ_AND_COPY (A, M, N)
C     ==================================
C
C Convert PC-EYE 6 bits/pel format to Starlink .BDF format
C with 8 bits/pel.No multiplication to convert 64 levels to
C 256 occurs in this version.
C
      BYTE A(M,N), BWORD1, BWORD2
C
        DO J = 1 , N
          K = N - J + 1
          DO I = 1 , M , 2
            READ (10) BWORD1, BWORD2
            A(I,K) = BWORD1
            A(I + 1,K) = BWORD2
          END DO
        END DO
        RETURN
      END
```

ADDIM.FOR

```
      PROGRAM ADD_IMAGES
C     ==================
C
C This program, in the absence of the Starlink program
C ADD, performs the same task, i.e.to add two input
C images together.
C
      INTEGER INPNT1,INPNT2,OUTPOINT,ASIZ(2),BSIZ(2),SIZE(2)
      INCLUDE 'STARDIR:FMTPAR.FOR'
      INCLUDE 'STARDIR:ERRPAR.FOR'
C
      CALL RDIMAG('INPUT1',FMT_SL,2,ASIZ,NDIM,INPNT1,ISTAT)
C
      CALL RDIMAG('INPUT2',FMT_SL,2,BSIZ,NDIM,INPNT2,ISTAT)
C
      DO I=1,2
        IF (ASIZ(I) .NE. BSIZ(I)) THEN
          WRITE (6,6000)
        END IF
      END DO
C
      DO I=1,2
        SIZE(I)=ASIZ(I)
      END DO
C
      CALL WRIMAG('OUTPUT',FMT_SL,SIZE,2,OUTPOINT,ISTAT)
C
      CALL ADDITION(%VAL(INPNT1),%VAL(INPNT2),SIZE,%VAL(OUTPOINT))
C
6000  FORMAT(1H ,'The images are of different size:ABORT OPERATION.')
C
      CALL EXIT
      END
C
C
      SUBROUTINE ADDITION(IN1,IN2,SZ,OUT)
C     ===================
C
      INTEGER SZ(2),IN1(SZ(1),SZ(2)),IN2(SZ(1),SZ(2))
      INTEGER OUT(SZ(1),SZ(2))
C
      DO J=1,SZ(2)
        DO I=1,SZ(1)
          OUT(I,J)=IN1(I,J)+IN2(I,J)
        END DO
      END DO
      RETURN
      END
```

TDISP.FOR

```
      PROGRAM TDISP
C     ============
C
      BYTE MIX(3,256)
      INTEGER SIZE(2), ESC, LUT_SIZE(2), LUT_P, B_POINT
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
        CALL RDIMAG ('INPUT',FMT_R,2,SIZE,NDIM,INPOINT,ISTAT)
        OFFSET = 0.0
        CALL RDKEYR ('OFFSET',.TRUE.,1,OFFSET,JUNK,ISTAT)
        SCALE = 1.0
        CALL RDKEYR ('SCALE',.TRUE.,1,SCALE,JUNK,ISTAT)
        CALL RDIMAG ('LUT',FMT_SL,2,LUT_SIZE,NDIM,LUT_P,ISTAT)
        ESC = 27
        CALL INITIALIZE
C  set machine rgb, additive, colour
        CALL SET_COLOR_MODE (4,3,1)
C  set one surface of 8 bit planes
        CALL SET_SURFACE_DEFINITIONS (1,8)
C  begin pixel operations, surface 1, 8 bits/pixel
        CALL BEGIN_PIXEL_OPERATIONS (1,11,8)
C  write pixels horizontally from bottom left
        CALL SET_PIXEL_WRITING_FACTORS (1,-1,0)
C  set viewport
        IPX1 = (1280 - SIZE(1)) / 2
        IPX2 = IPX1 + SIZE(1) - 1
        IPY1 = (1024 - SIZE(2)) / 2
        IPY2 = IPY1 + SIZE(2) - 1
        CALL SET_PIXEL_VIEWPORT (IPX1,IPY1,IPX2,IPY2)
C  set up cursor mapping
        CALL SET_WINDOW (1,1,SIZE(1),SIZE(2))
        CALL SET_VIEWPORT ((IPX1*16 + 8)/5,(IPY1*16 + 8)/5,
     *                     (IPX2*16 + 8)/5,(IPY2*16 + 8)/5)
        CALL SET_GIN_WINDOW (0,0,4095,4095)
        CALL SET_GIN_AREA (0,-1,0,0,4095,4095)
        CALL SET_GIN_GRIDDING (0,0,0)
        CALL FLUSH
C  set colour indexes for surface 1
        CALL READ_LUT (%VAL(LUT_P),MIX)
        CALL OUTPUT_BUFFER (MIX,3*256,'CM:')
C  offset, scale and output image
        ISIZE = SIZE(1)*SIZE(2)
        CALL GETDYN ('BUFFER',FMT_UB,ISIZE,B_POINT,ISTAT)
        CALL OFF_SCALE (%VAL(INPOINT),ISIZE,OFFSET,SCALE,%VAL(B_POINT))
        CALL OUTPUT_BUFFER (%VAL(B_POINT),ISIZE,'PX:0')
        CALL EXIT
      END
      SUBROUTINE OFF_SCALE (A, N, O, S, B)
C     ====================
C
      BYTE B(N), BBUF
      INTEGER IBUF
      REAL A(N)
      EQUIVALENCE (IBUF,BBUF)
C
        DO K = 1 , N
          IBUF = MIN(MAX(NINT((A(K) + O)/S),2),255)
          B(K) = BBUF
        END DO
        RETURN
      END
      SUBROUTINE OUTPUT_BUFFER (BUFFER, N, DEST)
```

```
C     ========================
C
      CHARACTER DEST*(*)
      BYTE BUFFER(N), DEVCHARS1(8)
      INTEGER*2 IOSB2(4), TW, DEVCHARS(4)
      INTEGER*4 SYS$ASSIGN, SYS$QIOW, SYS$DASSGN, LIB$GET_EF,
     *          LIB$FREE_EF, STAT, CHAN, LBUF, EFLAG, TL
      PARAMETER IO$M_NOEOF = '2000'X
      EQUIVALENCE (TL,TW), (DEVCHARS,DEVCHARS1)
      INCLUDE '($IODEF)'
      INCLUDE '($DCDEF)'
C
C     allocate channel for dma
        STAT = SYS$ASSIGN (%DESCR('XTA0:'),CHAN,,)
        IF (.NOT. STAT) CALL LIB$STOP (%VAL(STAT))
        STAT = LIB$GET_EF (EFLAG)
        IF (.NOT. STAT) CALL LIB$STOP (%VAL(STAT))
C     set up block size at both ends
        CALL SET_DMA_BLOCK_SIZE (16384)
        DEVCHARS1(1) = DC$_REALTIME
        DEVCHARS1(2) = 119
        TL = 16384
        DEVCHARS(2) = TW
        DEVCHARS1(5) = -1
        STAT = SYS$QIOW (%VAL(EFLAG),%VAL(CHAN),%VAL(IO$_SETMODE),
     *                   IOSB2,,,DEVCHARS,,,,,)
C     perform dma output section by section
        CALL COPY ('DM:1',DEST)
        IOFUNC = IO$_WRITEVBLK
        IPTR = 1
        DO WHILE (IPTR .LE. N)
          LEN = MIN (16384,N - IPTR + 1)
          IF ((IPTR + LEN - 1) .LT. N) THEN
              IOFUNC = IOR(IO$M_NOEOF,IOFUNC)
            ELSE
              IOFUNC = IAND(NOT(IO$M_NOEOF),IOFUNC)
          END IF
          STAT = SYS$QIOW (%VAL(EFLAG),%VAL(CHAN),%VAL(IOFUNC),IOSB2,,,
     *                     BUFFER(IPTR),%VAL(LEN),,,,)
          IF (.NOT. STAT) CALL LIB$STOP (%VAL(STAT))
          IPTR = IPTR + LEN
        END DO
        STAT = SYS$DASSGN (%VAL(CHAN))
        IF (.NOT. STAT) CALL LIB$STOP(%VAL(STAT))
        STAT = LIB$FREE_EF (EFLAG)
        IF (.NOT. STAT) CALL LIB$STOP(%VAL(STAT))
        RETURN
      END
      SUBROUTINE READ_LUT (LUT, MIX)
C     ==================
C
      BYTE MIX(3,256), BBUF
      INTEGER LUT(3,256), IBUF
      EQUIVALENCE (IBUF,BBUF)
C
        DO J = 1 , 256
          DO I = 1 , 3
            IBUF = LUT(I,J)
            MIX(I,257 - J) = BBUF
          END DO
        END DO
        IBUF = 255
```

```
  MIX(1,255) = BBUF
  MIX(2,255) = BBUF
  MIX(3,255) = BBUF
  RETURN
END
```

GREYSCL.FOR

```
      PROGRAM GREYTABLE
C     ================
C
C This program can be used to create .BDF lookup tables
C (LUT) for the display of images on the Tektronix 4115B
C colour graphics terminal.The tables this program is
C concerned with are grey-level tables.256 different
C shades of grey are available.Provision is made for the
C selection of the required number of grey levels to be
C used, allowing them to be spaced evenly.
C
C        Written   08/08/1986
C
      INTEGER SIZE(2),OUTPOINT,LUT(3,256),NUM
      INCLUDE 'STARDIR:ERRPAR.FOR'
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
      SIZE(1)=3
      SIZE(2)=256
      CALL WRIMAG('OUTPUT',FMT_SL,SIZE,2,OUTPOINT,ISTAT)
C
      NUM=256
      CALL RDKEYI('LEVELS',,TRUE,,1,NUM,JUNK,ISTAT)
C
      CALL WRITELUT(NUM,%VAL(OUTPOINT))
C
      CALL EXIT
C
      END
C
C
      SUBROUTINE WRITELUT(NUM,LUT)
C     ============================
C
C Writes a lookup table of the required size.
C
      INTEGER NUM,LUT(3,256),FACTOR
C
C Initialise output array
C
      DO J=1,256
        DO I=1,3
          LUT(I,J)=0
        END DO
      END DO
C
C Write table
C
      FACTOR=256/NUM
      DO J=1,NUM
        DO I=1,3
          LUT(I,J)=FACTOR*(J-1)
        END DO
      END DO
      RETURN
      END
```

HIST.FOR

```
      PROGRAM HISTDATA
C     ================
C
C This program obtains the data required to draw a grey-level
C histogram of an image by counting the number of pixels at
C each grey level.Note that the image required can have up to
C 256 grey levels (0..255).If necessary, it is easily modified
C to accomodate more.
C
C        Modified  22/08/1986
C
      INTEGER SIZE(2),INPOINT,NUM(0:255),LVL
      INCLUDE 'STARDIR:ERRPAR.FOR'
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
      CALL RDIMAG('INPUT',FMT_SL,2,SIZE,NDIM,INPOINT,ISTAT)
C
      LVL=256
      CALL RDKEYI('LEVELS',.TRUE.,1,LVL,JUNK,ISTAT)
C
      CALL FINDVAL(%VAL(INPOINT),SIZE(1),SIZE(2),LVL,NUM)
C
C Write results to files 'HISTRES.DAT', 'HISTPLT.DAT' & 'HISTLEV.DAT'
C
      OPEN (UNIT=1,NAME='HISTRES.DAT')
      OPEN (UNIT=2,NAME='HISTPLT.DAT')
      DO LEVEL=0,LVL-1
        WRITE (1,6000)LEVEL,NUM(LEVEL)
        WRITE (2,6001)NUM(LEVEL)
      END DO
      OPEN (UNIT=3,NAME='HISTLEV.DAT')
      WRITE (3,6002)LVL
C
6000  FORMAT(1H ,I4,'      ',I6)
6001  FORMAT(1H ,I6)
6002  FORMAT(1H ,I5)
C
      CALL EXIT
      END
C
C
      SUBROUTINE FINDVAL(IN,X,Y,LVL,NUM)
C     ==================
C
C This counts the number of pixels having each grey-level value
C
      INTEGER X,Y,LVL,NUM(0:255),IN(X,Y)
C
C Initialisation sequence
C
      DO LEVEL=0,255
        NUM(LEVEL)=0
      END DO
C
C Procedure starts
C
      DO J=1,Y
        DO I=1,X
          NUM(IN(I,J))=NUM(IN(I,J))+1
        END DO
      END DO
      RETURN
      END
```

DRAW.FOR

```
      PROGRAM HISTPLOT
C     ================
C
C This program uses the data files created by histogram
C data-gathering programs such as HIST.FOR to draw the
C histograms on the VT100 screen.The resulting histograms
C can be printed using the DUMPSCREEN command.Please note
C that the C-ITOH printer should be connected to the
C terminal and selected.Only one of the VT100 terminals
C can interface to the printer.In addition, a plotfile
C called 'HISTCAL.LIS' is created.This can be plotted on
C the CALCOMP plotter using the command 'PLOT HISTCAL.LIS',
C A maximum of 256 levels can be plotted.
C
C
      INTEGER NUM(0:255),MAX,LVL
      REAL NORM(0:255)
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='HISTPLT.DAT')
      DO I=0,LVL-1
        READ(2,*)NUM(I)
      END DO
C
      CALL NORMAL(NUM,MAX,LVL,NORM)
C
      CALL DRAWDAT(NORM,LVL)
C
      CALL DRAWCAL(NORM,LVL)
C
      CALL EXIT
      END
C
C
      SUBROUTINE NORMAL(NUM,MAX,LVL,NORM)
C     ===================
C
C Normalises the histogram data in order to facilitate
C automatic 'scaling' of the histogram plots.This is
C achieved by dividing all 'columns' by the highest,
C which results in a max. height of 1.This is then
C multiplied by a constant to obtain a 'decent' plot.
C
      INTEGER NUM(0:255),MAX,LVL
      REAL NORM(0:255)
C
C Initialise
C
      DO I=0,255
        NORM(I)=0.0
      END DO
C
      MAX=0
      DO I=0,LVL-1
        IF (NUM(I) .GT. MAX) THEN
          MAX=NUM(I)
        END IF
      END DO
      IF (MAX .EQ. 0) THEN
        MAX=1
      END IF
```

```
C
      DO I=0,LVL-1
        NORM(I)=(FLOAT(NUM(I)))/(FLOAT(MAX))
      END DO
      RETURN
      END
C
C
      SUBROUTINE DRAWDAT(NORM,LVL)
C     ============================
C
C Plots the grey-level histogram on the VT100 screen.
C
      INTEGER LVL
      REAL NORM(0:255),GREY,HIST,COL,INC
C
      CALL TEKPLOT
      CALL CLEAR_VT100
      CALL CLEAR_SCREEN
      COL=16.0/FLOAT(LVL)
      INC=1.0/COL
      CALL AXIS(0.0,0.0,'GREY LEVEL',-10,16.0,0.0,0.0,INC)
      DO I=0,LVL-1
        GREY=COL*FLOAT(I)
        HIST=13*NORM(I)
        CALL PLOT(GREY,0.0,3)
        CALL PLOT(GREY,HIST,2)
      END DO
      CALL VT100
      CALL ENDPLOT
      RETURN
      END
C
C
      SUBROUTINE DRAWCAL(NORM,LVL)
C     ============================
C
C Plots the grey-level histogram on the CALCOMP plotter
C
      INTEGER LVL
      REAL NORM(0:255),GREY,HIST,COL,INC
C
      CALL CALPLOT(5,'HISTCAL')
      COL=16.0/FLOAT(LVL)
      INC=1.0/COL
      CALL AXIS(0.0,0.0,'GREY LEVEL',-10,16.0,0.0,0.0,INC)
      DO I=0,LVL-1
        GREY=COL*FLOAT(I)
        HIST=13*NORM(I)
        CALL PLOT(GREY,0.0,3)
        CALL PLOT(GREY,HIST,2)
      END DO
      CALL ENDPLOT
      RETURN
      END
```

LEVELENT.FOR

```
      PROGRAM ENTROPY
C     ===============
C
C This program determines the optimum threshold for a given
C image using the entropy of the histogram.As input the data
C file HISTPLT.DAT created by the HIST.FOR program is used.
C The method is one described by J.N.Kapur, P.K.Sahoo and
C A.K.C.Wong in their paper 'A New Method for Gray-Level
C Picture Thresholding Using the Entropy of the Histogram.'
C A maximum of 256 grey levels can be used.If necessary, the
C program can be modified to accomodate more.
C
C
C          Written   04/06/1986
C          Modified  01/08/1986
C          Modified  11/09/1986
C
      INTEGER F(0:255),TOT,LEVEL,LVL
      REAL P(0:255),PS,PSX,HS,HSX,PSIMAX,PSI(0:255)
C
C Initialise arrays
C
      DO I=0,255
        F(I)=0
        P(I)=0.0
        PSI(I)=0.0
      END DO
C
C Read in data from HISTPLT.DAT and HISTLEV.DAT
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='HISTPLT.DAT')
      DO I=0,LVL-1
        READ (2,*)F(I)
      END DO
C
C Calculate the grey-level fractions
C
      TOT=0
      DO I=0,LVL-1
        TOT=TOT+F(I)
      END DO
      DO I=0,LVL-1
        P(I)=FLOAT(F(I))/FLOAT(TOT)
      END DO
C
C Commence determination of threshold
C
      PSIMAX=0.0
      DO S=0,LVL-2
        PS=0.0
        DO I=0,S
          PS=PS+P(I)
        END DO
        PSX=1.0-PS
C
        HS=0.0
        DO I=0,S
          IF (P(I) .GT. 0.0) THEN
            HS=HS-(P(I)*LOG(P(I)))
          END IF
```

```
          END DO
C
          HSX=0.0
          DO I=S+1,LVL-1
            IF (P(I) .GT. 0.0) THEN
              HSX=HSX-(P(I)*LOG(P(I)))
            END IF
          END DO
C
          IF (PS .GT. 0.0) THEN
            IF (PSX .GT. 0.0) THEN
              PSI(S)=(LOG(PS*PSX))+(HS/PS)+(HSX/PSX)
            END IF
          END IF
C
          IF (PSI(S).GT.PSIMAX) THEN
            PSIMAX=PSI(S)
            LEVEL=S
          END IF
        END DO
C
C Write the array of PSI values to ENTRES.DAT
C
        OPEN (UNIT=3,NAME='ENTRES.DAT')
        DO I=0,LVL-1
          WRITE (3,5000)PSI(I)
        END DO
C
C Print the results on the screen
C
        WRITE (6,6000)PSIMAX
        WRITE (6,6001)LEVEL
C
5000    FORMAT(1H ,F10.3)
6000    FORMAT(1H ,'The max. value of PSI is:',F10.3)
6001    FORMAT(1H ,'The optimum threshold is:',I5)
C
        CALL EXIT
        END
```

DRAWENT.FOR

```
      PROGRAM HISTPLOT
C     ================
C
C This program uses the data files created by threshold
C selection programs such as LEVELENT.FOR to draw the
C criterion functions on the VT100 screen.The resulting
C histograms can be printed using the DUMPSCREEN command.
C Note that the C-ITOH printer should be connected to the
C terminal and selected.Only one of the VT100 terminals
C can interface to the printer.In addition, a plotfile
C called 'ENTCAL.LIS' is created.This can be plotted on
C the CALCOMP plotter using the command 'PLOT ENTCAL.LIS'.
C A maximum of 256 levels can be plotted.
C
C
C          Written   17/09/1986
C
      INTEGER LVL
      REAL NUM(0:255),MAX,NORM(0:255)
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='ENTRES.DAT')
      DO I=0,LVL-1
        READ(2,*)NUM(I)
      END DO
C
      CALL FINDMAX(NUM,MAX,LVL,NORM)
C
      CALL DRAWDAT(NORM,MAX,LVL)
C
      CALL DRAWCAL(NORM,MAX,LVL)
C
      CALL EXIT
      END
C
C
      SUBROUTINE FINDMAX(NUM,MAX,LVL,NORM)
C     ====================
C
C Finds the maximum value of FSI to allow labeling of the
C vertical axis.
C
      INTEGER LVL
      REAL NUM(0:255),MAX,NORM(0:255)
C
      MAX=0.0
      DO I=0,LVL-1
        NORM(I)=ABS(NUM(I))
        IF (NORM(I) .GT. MAX) THEN
          MAX=NORM(I)
        END IF
      END DO
      IF (MAX .EQ. 0.0) THEN
        MAX=1.0
      END IF
C
      RETURN
      END
C
C
      SUBROUTINE DRAWDAT(NORM,MAX,LVL)
```

```
C     ==================
C
C Plots the graph on the VT100 screen.
C
      INTEGER LVL
      REAL NORM(0:255),MAX
      REAL CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1*MAX
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
      CALL TEKPLOT
      CALL CLEAR_VT100
      CALL CLEAR_SCREEN
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'PSI',3,10.0,90.0,0.0,YINC)
      DO I=0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=NORM(I)
      END DO
      CX(LVL+1)=0.0
      CX(LVL+2)=XINC
      CY(LVL+1)=0.0
      CY(LVL+2)=YINC
      CALL LINE(CX,CY,LVL,1,0,0)
      CALL VT100
      CALL ENDPLOT
      RETURN
      END
C
C
      SUBROUTINE DRAWCAL(NORM,MAX,LVL)
C     ================================
C
C Plots the graph on the CALCOMP plotter
C
      INTEGER LVL
      REAL NORM(0:255),MAX
      REAL CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1*MAX
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
      CALL CALPLOT(5,'ENTCAL')
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'PSI',3,10.0,90.0,0.0,YINC)
      DO I=0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=NORM(I)
      END DO
      CX(LVL+1)=0.0
      CX(LVL+2)=XINC
      CY(LVL+1)=0.0
      CY(LVL+2)=YINC
```

```
CALL LINE(CX,CY,LVL,1,0,0)
CALL ENDPLOT
RETURN
END
```

LEVELMIN.FOR

```
      PROGRAM MINERROR
C     ================
C
C This program determines the optimum threshold for a given
C image using a statistical method to minimise the error.As
C input the data file HISTFLT.DAT created by the HIST.FOR
C program is used.The method is one described by J.Kittler
C and J.Illingworth in their paper 'Minimum Error
C Thresholding'.A maximum of 256 grey levels can be used.If
C necessary, the program can be modified to accomodate more.
C
C
C              Written   15/07/1986
C              Modified  01/08/1986
C              Modified  11/09/1986
C
      INTEGER HIST(0:255),THRESH,T,G,LVL
      REAL J(0:255),JCF,BSUM1,BSUM2,M1,M2,S1,S2,ASUM1,ASUM2
      REAL PS1,PS2,PP1,PP2,A1,A2,P1,P2,HTOT,NHIST(0:255),MAXJ
C
C Initialise arrays
C
      DO I=0,255
        HIST(I)=0
        NHIST(I)=0.0
        J(I)=0.0
      END DO
C
C Read in histogram data from HISTFLT.DAT and HISTLEV.DAT
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='HISTFLT.DAT')
      DO I=0,LVL-1
        READ (2,*)HIST(I)
      END DO
C
C Normalise the histogram
C
      HTOT=0.0
      DO I=0,LVL-1
        HTOT=HTOT+FLOAT(HIST(I))
      END DO
      DO I=0,LVL-1
        NHIST(I)=(FLOAT(HIST(I)))/HTOT
      END DO
C
C Commence calculation
C
      JCF=0.0
      DO T=0,LVL-2
        ASUM1=0.0
        ASUM2=0.0
        BSUM1=0.0
        BSUM2=0.0
        P1=0.0
        P2=0.0
        A1=0.0
        A2=0.0
        M1=0.0
        M2=0.0
        S1=0.0
```

```
          S2=0.0
          PS1=0.0
          PS2=0.0
          PP1=0.0
          PP2=0.0
          DO G=0,T
            P1=P1+NHIST(G)
            ASUM1=ASUM1+(NHIST(G)*FLOAT(G))
          END DO
          IF (P1 .GT. 0.0) THEN
            M1=ASUM1/P1
          END IF
          DO G=0,T
            A1=(FLOAT(G)-M1)**2
            BSUM1=BSUM1+A1*NHIST(G)
          END DO
          IF (P1 .GT. 0.0) THEN
            S1=SQRT(BSUM1/P1)
          END IF
          DO G=T+1,LVL-1
            P2=P2+NHIST(G)
            ASUM2=ASUM2+(NHIST(G)*FLOAT(G))
          END DO
          IF (P2 .GT. 0.0) THEN
            M2=ASUM2/P2
          END IF
          DO G=T+1,LVL-1
            A2=(FLOAT(G)-M2)**2
            BSUM2=BSUM2+A2*NHIST(G)
          END DO
          IF (P2 .GT. 0.0) THEN
            S2=SQRT(BSUM2/P2)
          END IF
          IF (P1 .GT. 0.0) THEN
            PP1=P1*LOG(P1)
            IF (S1 .GT. 0.0) THEN
              PS1=P1*LOG(S1)
            END IF
          END IF
          IF (P2 .GT. 0.0) THEN
            PP2=P2*LOG(P2)
            IF (S2 .GT. 0.0) THEN
              PS2=P2*LOG(S2)
            END IF
          END IF
          J(T)=1.0+2*(PS1+PS2)-2*(PP1+PP2)
        END DO
        THRESH=0
C
C Find min. value of the criterion function
C
        JCF=J(0)
        DO T=1,LVL-2
          IF (J(T) .LT. JCF) THEN
            JCF=J(T)
            THRESH=T
          END IF
        END DO
C
C Print result on screen and write criterion function
C to a data file MINRES.DAT.
C
```

```
      OPEN (UNIT=2,NAME='MINRES.DAT')
      DO T=0,LVL-1
        WRITE (2,5000)J(T)
      END DO
      WRITE (6,6000)JCF
      WRITE (6,6001)THRESH
C
5000  FORMAT(1H ,F10.5)
6000  FORMAT(1H ,'The minimum of the crit. func. was:',F10.5)
6001  FORMAT(1H ,'*** OPTIMUM THRESHOLD=',I4,' ***')
C
      CALL EXIT
      END
```

DRAWMIN.FOR

```
      PROGRAM HISTPLOT
C     ================
C
C This program uses the data files created by threshold
C selection programs such as LEVELMIN.FOR to draw the
C criterion functions on the VT100 screen.The resulting
C histograms can be printed using the DUMPSCREEN command.
C Note that the C-ITOH printer should be connected to the
C terminal and selected.Only one of the VT100 terminals
C can interface to the printer.In addition, a plotfile
C called 'MINCAL.LIS' is created.This can be plotted on
C the CALCOMP plotter using the command 'PLOT MINCAL.LIS'.
C A maximum of 256 levels can be plotted.
C
C
C          Written   17/09/1986
C
      INTEGER LVL
      REAL NUM(0:255),MAX,NORM(0:255)
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='MINRES.DAT')
      DO I=0,LVL-1
        READ(2,*)NUM(I)
      END DO
C
      CALL FINDMAX(NUM,MAX,LVL,NORM)
C
      CALL DRAWDAT(NORM,MAX,LVL)
C
      CALL DRAWCAL(NORM,MAX,LVL)
C
      CALL EXIT
      END
C
C
      SUBROUTINE FINDMAX(NUM,MAX,LVL,NORM)
C     ====================
C
C Finds the maximum value of J to allow labeling of the
C vertical axis.
C
      INTEGER LVL
      REAL NUM(0:255),MAX,NORM(0:255)
C
      MAX=0.0
      DO I=0,LVL-1
        NORM(I)=ABS(NUM(I))
        IF (NORM(I) .GT. MAX) THEN
          MAX=NORM(I)
        END IF
      END DO
      IF (MAX .EQ. 0.0) THEN
        MAX=1.0
      END IF
C
      RETURN
      END
C
C
      SUBROUTINE DRAWDAT(NORM,MAX,LVL)
```

```
C     ==================
C
C Plots the graph on the VT100 screen.
C
      INTEGER LVL
      REAL NORM(0:255),MAX
      REAL CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1*MAX
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
      CALL TEKPLOT
      CALL CLEAR_VT100
      CALL CLEAR_SCREEN
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'J-CRITERION',11,10.0,90.0,0.0,YINC)
      DO I=0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=NORM(I)
      END DO
      CX(LVL+1)=0.0
      CX(LVL+2)=XINC
      CY(LVL+1)=0.0
      CY(LVL+2)=YINC
      CALL LINE(CX,CY,LVL,1,0,0)
      CALL VT100
      CALL ENDPLOT
      RETURN
      END
C
C
      SUBROUTINE DRAWCAL(NORM,MAX,LVL)
C     ================
C
C Plots the graph on the CALCOMP plotter
C
      INTEGER LVL
      REAL NORM(0:255),MAX
      REAL CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1*MAX
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
      CALL CALPLOT(5,'MINCAL')
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'J-CRITERION',11,10.0,90.0,0.0,YINC)
      DO I=0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=NORM(I)
      END DO
      CX(LVL+1)=0.0
      CX(LVL+2)=XINC
      CY(LVL+1)=0.0
      CY(LVL+2)=YINC
```

```
      CALL LINE(CX,CY,LVL,1,0,0)
      CALL ENDPLOT
      RETURN
      END
```

LEVELMOM.FOR

```
      PROGRAM MOMENT
C     ==============
C
C This program determines the optimum threshold for a given
C image while preserving the first four moments of the image.
C As input the data file HISTPLT.DAT created by the HIST.FOR
C program is used.The method is one described by Wen-Hsiang
C Tsai in his paper 'Moment-Preserving Thresholding: A New
C Approach'.Up to 256 grey levels can be used.If necessary,
C the program can be modified to accomodate more.
C
C
C             Written   05/06/1986
C             Modified  01/08/1986
C
      INTEGER F(0:255),TOT,LEVEL,LVL
      REAL P(0:255),M(0:3),C0,C1,Z0,Z1,P0,P1,PR,PROB
C
C Initialise arrays
C
      DO I=0,255
        F(I)=0
        P(I)=0.0
      END DO
C
C Read in data from HISTPLT.DAT and HISTLEV.DAT
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='HISTPLT.DAT')
      DO I=0,LVL-1
        READ (2,*)F(I)
      END DO
C
C Calculate the grey-level fractions
C
      TOT=0
      DO I=0,LVL-1
        TOT=TOT+F(I)
      END DO
      DO I=0,LVL-1
        P(I)=FLOAT(F(I))/FLOAT(TOT)
      END DO
C
C Calculate the first four moments M(0)..M(3).
C By definition, M(0)=1.0
C
      M(0)=1.0
      DO I=1,3
        M(I)=0.0
      END DO
      DO I=1,3
        DO J=0,LVL-1
          M(I)=M(I)+(F(J)*(J**I))
        END DO
      END DO
C
C Calculate the intermediate coefficients C0 & C1
C
      C0=(M(1)*M(3)-M(2)**2)/(M(0)*M(2)-M(1)**2)
      C1=(M(1)*M(2)-M(0)*M(3))/(M(0)*M(2)-M(1)**2)
C
```

```
C Calculate the two final grey-levels Z0 & Z1
C
      Z0=0.5*(-C1-SQRT(C1**2-4*C0))
      Z1=0.5*(-C1+SQRT(C1**2-4*C0))
C
C Calculate the two final grey-level fractions P0 & P1
C
      P0=(Z1-M(1))/(Z1-Z0)
      P1=1.0-P0
C
C Commence determination of the threshold
C
      LEVEL=0
      PR=0.0
      DO J=0,LVL-1
        PR=PR+P(J)
        IF (PR .LE. P0) THEN
          LEVEL=J
          PROB=PR
        END IF
      END DO
C
C Print the results on the screen
C
      WRITE (6,6000)M(0),M(1),M(2),M(3)
      WRITE (6,6001)Z0,Z1
      WRITE (6,6002)P0
      WRITE (6,6003)PROB
      WRITE (6,6004)LEVEL
C
6000  FORMAT (1H ,'M0=',F5.3,' M1=',F10.3,' M2=',F10.3,' M3=',F10.3)
6001  FORMAT (1H ,'Z0=',F10.3,' Z1=',F10.3)
6002  FORMAT (1H ,'Required lower fraction:',F5.3)
6003  FORMAT (1H ,'Obtained lower fraction:',F5.3)
6004  FORMAT (1H ,'** OPTIMUM THRESHOLD **:',I5)
C
      CALL EXIT
      END
```

LEVELAB5.FOR

```
      PROGRAM MOD_DIFF
C     ================
C
C This program uses the 'magnitude of the difference' threshold
C selection method proposed in Chapter 3.The magnitude of
C the difference between the original image and the bilevel
C result is evaluated for all thresholds.The threshold
C selected is that one which minimises this difference.An
C output file, 'RESAB5.DAT', is produced for plotting function
C versus threshold.
C
C
C          Written   19/06/1987
C
      INTEGER HIST(0:255),LVL,THRESH
      REAL X(0:255),CRIT
C
C Read data from histogram files
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='HISTPLT.DAT')
      DO I=0,LVL-1
        READ (2,*)HIST(I)
      END DO
C
C Find the 'magnitude function', X
C
      CALL MAGNITUDE(HIST,LVL,X)
C
C Calculate the threshold
C
      CALL THRESHOLD(X,LVL,CRIT,THRESH)
C
C Write X to 'RESAB5.DAT'
C
      OPEN (UNIT=3,NAME='RESAB5.DAT')
      DO T=0,LVL-1
        WRITE (3,6000)X(T)
      END DO
C
C Write CRIT and THRESH to screen
C
      WRITE (6,6001)CRIT,THRESH
C
6000  FORMAT(1H ,F12.7)
6001  FORMAT(1H ,'Criterion =',F12.7,' Threshold =',I4)
C
      CALL EXIT
      END
C
C
      SUBROUTINE MAGNITUDE(HIST,LVL,X)
C     ================================
C
C This obtains the criterion function on which the choice
C of threshold is based.
C
      INTEGER HIST(0:255),LVL,T,G
      REAL A,B,C,SUM,MS0,MS1,MU0,MU1,P0,P1
```

```
      REAL MU,X(0:255)
C
      SUM=0.0
      DO G=0,LVL-1
        SUM=SUM+FLOAT(HIST(G))
      END DO
C
      C=0.0
      MU=0.0
      DO T=0,LVL-2
        MS0=0.0
        MS1=0.0
        MU0=0.0
        MU1=0.0
        P0=0.0
        P1=0.0
        A=0
        B=0
C
        DO G=0,T
          P0=P0+FLOAT(HIST(G))/SUM
          MS0=MS0+FLOAT(G*HIST(G))/SUM
        END DO
        IF (P0 .GT. 0.0) THEN
          MU0=MS0/P0
        END IF
        DO G=0,T
          A=A+(ABS(FLOAT(G)-MU0))*FLOAT(HIST(G))
        END DO
C
        DO G=T+1,LVL-1
          MS1=MS1+FLOAT(G*HIST(G))/SUM
        END DO
        P1=1.0-P0
        IF (P1 .GT. 0.0) THEN
          MU1=MS1/P1
        END IF
        DO G=T+1,LVL-1
          B=B+(ABS(FLOAT(G)-MU1))*FLOAT(HIST(G))
        END DO
        X(T)=(A+B)/SUM
      END DO
      DO G=0,LVL-1
        MU=MU+FLOAT(G*HIST(G))/SUM
      END DO
      DO G=0,LVL-1
        C=C+(ABS(FLOAT(G)-MU))*FLOAT(HIST(G))
      END DO
      X(LVL-1)=C/SUM
      RETURN
      END
C
C
      SUBROUTINE THRESHOLD(X,LVL,CRIT,THRESH)
C     ======================
C
C This determines the optimum threshold, obtained by
C minimising the magnitude of the difference between the
C binary image and the original.
C
```

```
      INTEGER LVL,THRESH,T
      REAL X(0:255),CRIT
C
      THRESH=0
      CRIT=X(0)
      DO T=1,LVL-1
        IF (X(T) .LT. CRIT) THEN
          CRIT=X(T)
          THRESH=T
        END IF
      END DO
C
      RETURN
      END
```

PLOTAB5.FOR

```
      PROGRAM HISTPLOT
C     ================
C
C This program uses the data files created by threshold
C selection programs such as LEVELAB5.FOR to draw the
C criterion functions on the VT100 screen.The resulting
C histograms can be printed using the DUMPSCREEN command.
C Note that the C-ITOH printer should be connected to the
C terminal and selected.Only one of the VT100 terminals
C can interface to the printer.In addition, a plotfile
C called 'ABSCAL.LIS' is created.This can be plotted on
C the CALCOMP plotter using the command 'PLOT ABSCAL.LIS'.
C A maximum of 256 levels can be plotted.
C
C
C           Written   19/06/1987
C
      INTEGER LVL
      REAL NUM(0:255),MAX,NORM(0:255)
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='RESAB5.DAT')
      DO I=0,LVL-1
        READ(2,*)NUM(I)
      END DO
C
      CALL FINDMAX(NUM,MAX,LVL,NORM)
C
      CALL DRAWDAT(NORM,MAX,LVL)
C
      CALL DRAWCAL(NORM,MAX,LVL)
C
      CALL EXIT
      END
C
C
      SUBROUTINE FINDMAX(NUM,MAX,LVL,NORM)
C     ====================
C
C Finds the maximum value of the function to allow labeling of the
C vertical axis.
C
      INTEGER LVL
      REAL NUM(0:255),MAX,NORM(0:255)
C
      MAX=0.0
      DO I=0,LVL-1
        NORM(I)=ABS(NUM(I))
        IF (NORM(I) .GT. MAX) THEN
          MAX=NORM(I)
        END IF
      END DO
      IF (MAX .EQ. 0.0) THEN
        MAX=1.0
      END IF
C
      RETURN
      END
C
```

```
C
      SUBROUTINE DRAWDAT(NORM,MAX,LVL)
C     ================
C
C Plots the graph on the VT100 screen.
C
      INTEGER LVL
      REAL NORM(0:255),MAX
      REAL CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1*MAX
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
      CALL STARTPLOT(6,'TEK')
      CALL CLEAR_VT100
      CALL CLEAR_SCREEN
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'MAGNITUDE',9,10.0,90.0,0.0,YINC)
      DO I=0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=NORM(I)
      END DO
      CX(LVL+1)=0.0
      CX(LVL+2)=XINC
      CY(LVL+1)=0.0
      CY(LVL+2)=YINC
      CALL LINE(CX,CY,LVL,1,0,0)
      CALL VT100
      CALL ENDPLOT
      RETURN
      END
C
C
      SUBROUTINE DRAWCAL(NORM,MAX,LVL)
C     ================
C
C Plots the graph on the CALCOMP plotter (or the EPSON)
C
      INTEGER LVL
      REAL NORM(0:255),MAX
      REAL CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1*MAX
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
      CALL STARTPLOT(5,'ABSCAL')
      CALL PLOT(1.0,1.0,-3)
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'MAGNITUDE',9,10.0,90.0,0.0,YINC)
      DO I=0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=NORM(I)
```

```
END DO
CX(LVL+1)=0.0
CX(LVL+2)=XINC
CY(LVL+1)=0.0
CY(LVL+2)=YINC
CALL LINE(CX,CY,LVL,1,0,0)
CALL ENDPLOT
RETURN
END
```

LEVELAB6.FOR

```
      PROGRAM TEST_SIX
C     ================
C
C This program uses the Maximum Correlation threshold selection
C method proposed in Chapter 3.The correlation between the
C original image and the bilevel result is evaluated for all
C thresholds.The threshold selected is that one which maximises
C the correlation.An output file, 'RESAB6.DAT', is produced for
C plotting the correlation versus threshold.
C
C
C          Written   22/08/1987
C
      INTEGER HIST(0:255),LVL,THRESH
      REAL COR(0:255),CCOR
C
C Read data from histogram files
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='HISTPLT.DAT')
      DO I=0,LVL-1
        READ (2,*)HIST(I)
      END DO
C
C Find the correlation function, COR
C
      CALL CORRELATE(HIST,LVL,COR)
C
C Calculate the threshold
C
      CALL THRESHOLD(COR,LVL,CCOR,THRESH)
C
C Write COR to 'RESAB6.DAT'
C
      OPEN (UNIT=3,NAME='RESAB6.DAT')
      DO T=0,LVL-1
        WRITE (3,6000)COR(T)
      END DO
C
C Write CCOR and THRESH to screen
C
      WRITE (6,6001)CCOR,THRESH
C
6000  FORMAT(1H ,F12.7)
6001  FORMAT(1H ,'Max. correlation =',F12.7,' Threshold  ',I4)
C
      CALL EXIT
      END
C
C
      SUBROUTINE CORRELATE(HIST,LVL,COR)
C     ==================================
C
C This calculates the correlation function relating the
C original input picture to the thresholded result.
C
      INTEGER HIST(0:255),LVL,T,G
      REAL SUM,P(0:255),MS0,MS1,MU0,MU1,P0,P1
      REAL EX,EX2,VX,EY(0:255),EY2(0:255),VY(0:255),EXY(0:255)
```

```
      REAL COR(0:255)
C
      SUM=0.0
      DO G=0,LVL-1
        SUM=SUM+FLOAT(HIST(G))
      END DO
C
      DO G=0,LVL-1
        P(G)=FLOAT(HIST(G))/SUM
      END DO
C
      EX=0.0
      EX2=0.0
      VX=0.0
      DO T=0,LVL-1
        EY(T)=0.0
        EY2(T)=0.0
        VY(T)=0.0
        EXY(T)=0.0
        COR(T)=0.0
      END DO
C
      DO G=0,LVL-1
        EX=EX+FLOAT(G)*P(G)
        EX2=EX2+FLOAT(G**2)*P(G)
      END DO
      VX=EX2-(EX**2)
C
      DO T=0,LVL-2
        P0=0.0
        P1=0.0
        MS0=0.0
        MS1=0.0
        MU0=0.0
        MU1=0.0
        DO G=0,T
          P0=P0+P(G)
          MS0=MS0+G*P(G)
        END DO
        P1=1.0-P0
        DO G=T+1,LVL-1
          MS1=MS1+G*P(G)
        END DO
        IF (P0 .GT. 0.0) THEN
          MU0=MS0/P0
        END IF
        IF (P1 .GT. 0.0) THEN
          MU1=MS1/P1
        END IF
C
        DO G=0,T
          EY(T)=EY(T)+MU0*P(G)
          EY2(T)=EY2(T)+(MU0**2)*P(G)
          EXY(T)=EXY(T)+MU0*FLOAT(G)*P(G)
        END DO
        DO G=T+1,LVL-1
          EY(T)=EY(T)+MU1*P(G)
          EY2(T)=EY2(T)+(MU1**2)*P(G)
          EXY(T)=EXY(T)+MU1*FLOAT(G)*P(G)
        END DO
```

```
          VY(T)=EY2(T)-(EY(T)**2)
C
          IF ((VX*VY(T)) .GT. 0.0) THEN
            COR(T)=(EXY(T)-EX*EY(T))/SQRT(VX*VY(T))
          END IF
C
        END DO
        RETURN
        END
C
C
        SUBROUTINE THRESHOLD(COR,LVL,CCOR,THRESH)
C       =========================
C
C This determines the optimum threshold, obtained by
C maximising the correlation between the binary image
C and the original.
C
        INTEGER LVL,THRESH,T
        REAL COR(0:255),CCOR
C
        THRESH=0
        CCOR=COR(0)
        DO T=1,LVL-1
          IF (COR(T) .GT. CCOR) THEN
            CCOR=COR(T)
            THRESH=T
          END IF
        END DO
C
        RETURN
        END
```

PLOTAB6.FOR

```
      PROGRAM PLOT_TEST_SIX
C     ---------------------
C
C This program creates CALCOMP plotter plotfile CORRAB6.LIS
C using data read in from files 'HISTLEV.DAT' and 'RESAB6.DAT'
C created by 'HIST.FOR' and 'LEVELAB6.FOR', respectively.
C This plotfile contains a plot of the correlation, C(T), versus
C threshold level, T.This correlation criterion is defined in
C notes by A.Brink (M.Sc Book 3) and is calculated by the
C thresholding program, 'LEVELAB6.FOR'.
C
C
C           Written   22/08/1987
C
      INTEGER LVL
      REAL C(0:255)
C
C Initialise array
C
      DO I=0,255
        C(I)=0.0
      END DO
C
C Read data from files
C
      OPEN (UNIT=1,NAME='HISTLEV.DAT')
      READ (1,*)LVL
      OPEN (UNIT=2,NAME='RESAB6.DAT')
      DO I=0,LVL-1
        READ (2,*)C(I)
      END DO
C
      CALL DRAWTEK(C,LVL)
C
      CALL DRAWCAL(C,LVL)
C
      CALL EXIT
      END
C
C
      SUBROUTINE DRAWTEK(C,LVL)
C     -------------------------
C
C This plots correlation criterion, C, versus threshold level, T,
C on the VT-100 screen.
C
      INTEGER LVL
      REAL C(0:255),CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
C Plot array C on screen.
C
      CALL STARTPLOT(6,'TEK')
      CALL CLEAR_VT100
```

```
      CALL CLEAR_SCREEN
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'CORRELATION',11,10.0,90.0,0.0,YINC)
      DO I=0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=C(I)
      END DO
      CX(LVL+1)=0.0
      CX(LVL+2)=XINC
      CY(LVL+1)=0.0
      CY(LVL+2)=YINC
      CALL LINE(CX,CY,LVL,1,0,0)
      CALL VT100
      CALL ENDPLOT
      RETURN
      END
C
C
      SUBROUTINE DRAWCAL(C,LVL)
C     =========================
C
C This produces a plot (.LIS) file for the correlation criterion
C versus threshold level (EPSON plotter)
C
C Note that the coordinate arrays, CX and CY, are dimensioned
C 2 elements larger than appears necessary.These extra
C elements contain the initial x & y values, and their
C increments.These must be entered explicitly when not calling
C the SCALE subroutine.
C
      INTEGER LVL
      REAL C(0:255),CX(258),CY(258),XINC,YINC
C
      XINC=FLOAT(LVL)/16.0
      YINC=0.1
      DO I=1,258
        CX(I)=0.0
        CY(I)=0.0
      END DO
C
C Prepare plotfile for array C
C
      CALL STARTPLOT(5,'CORRAB6')
      CALL PLOT(1.0,1.0,-3)
      CALL AXIS(0.0,0.0,'THRESHOLD',-9,16.0,0.0,0.0,XINC)
      CALL AXIS(0.0,0.0,'CORRELATION',11,10.0,90.0,0.0,YINC)
      DO I 0,LVL-1
        CX(I+1)=FLOAT(I)
        CY(I+1)=C(I)
      END DO
      CX(LVL+1)=0.0
      CX(LVL+2)=XINC
      CY(LVL+1)=0.0
      CY(LVL+2)=YINC
      CALL LINE(CX,CY,LVL,1,0,0)
      CALL ENDPLOT
      RETURN
      END
```

TWOTONE.FOR

```
      PROGRAM THRESHOLD
C     ==========
C
C This program allows the user to convert an image
C into a binary image by specifying input and output
C files, threshold, minimum grey level and maximum
C grey level.The optimum threshold can be determined
C by using other programs, such as LEVELENT.FOR.
C
C
C
C      Rewritten 04/08/1986
C
      INTEGER SIZE(2),INPOINT,OUTPOINT,ISIZE
      INTEGER THLD,MIN,MAX
      INCLUDE 'STARDIR:ERRPAR.FOR'
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
      CALL RDIMAG('INPUT',FMT_SL,2,SIZE,NDIM,INPOINT,ISTAT)
      CALL WRIMAG('OUTPUT',FMT_SL,SIZE,2,OUTPOINT,ISTAT)
      MIN=0
      CALL RDKEYI('MINIMUM',.TRUE.,1,MIN,JUNK,ISTAT)
      MAX=255
      CALL RDKEYI('MAXIMUM',.TRUE.,1,MAX,JUNK,ISTAT)
      THLD=128
      CALL RDKEYI('LEVEL',.TRUE.,1,THLD,JUNK,ISTAT)
      ISIZE=SIZE(1)*SIZE(2)
      CALL CONTRAST(%VAL(INPOINT),ISIZE,THLD,MIN,MAX,%VAL(OUTPOINT))
      CALL FRDATA('OUTPUT',ISTAT)
      CALL EXIT
      END
C
C
      SUBROUTINE CONTRAST(IN,K,THLD,MIN,MAX,OUT)
C     ==========
C
C Converts the input image to a binary image.
C
      INTEGER K,IN(K),OUT(K),THLD,MIN,MAX
C
      DO I=1,K
        IF (IN(I) .LE. THLD) THEN
            OUT(I)=MIN
          ELSE
            OUT(I)=MAX
        END IF
      END DO
      RETURN
      END
```

TESTBOX.FOR

```
      PROGRAM DRAW_SQUARE
C     ===================
C
C This program produces a .BDF picture file containing four
C blocks, four points and a cross on contrasting backgrounds.
C This test picture can be used in the evaluation of thresholding
C methods as follows:After reducing contrast and smoothing
C the test image and adding noise, a thresholding procedure
C can be applied.The 'difference' between the thresholded
C result and  the original can be used as a measure of
C 'goodness of method'.
C
C
C
C             Written   01/09/1986
C             Modified  03/10/1986
C
      INTEGER SIZE(2),OUTPOINT,LOW,HIGH
      INCLUDE 'STARDIR:FMTPAR.FOR'
      INCLUDE 'STARDIR:ERRPAR.FOR'
C
      SIZE(1)=128
      SIZE(2)=128
      CALL WRIMAG('OUTPUT',FMT_SL,SIZE,2,OUTPOINT,ISTAT)
C
      LOW=0
      CALL RDKEYI('LOW',,TRUE.,1,LOW,JUNK,ISTAT)
C
      HIGH=255
      CALL RDKEYI('HIGH',,TRUE.,1,HIGH,JUNK,ISTAT)
C
      CALL SQUARE(LOW,HIGH,%VAL(OUTPOINT))
C
      CALL EXIT
C
      END
C
C
      SUBROUTINE SQUARE(LOW,HIGH,BOX)
C     ===============
C
C This draws the square and cross required.
C
      INTEGER LOW,HIGH,BOX(128,128)
C
      DO J=1,128
        DO I=1,128
          BOX(I,J)=LOW
        END DO
      END DO
C
      DO J=16,35
        DO I=16,35
          BOX(I,J)=HIGH
        END DO
        DO I=92,111
          BOX(I,J)=HIGH
        END DO
      END DO
      DO J=92,111
        DO I=16,35
          BOX(I,J)=HIGH
```

```
        END DO
        DO I=92,111
          BOX(I,J)=HIGH
        END DO
      END DO
C
        DO I=35,94
          BOX(I,64)=HIGH
        END DO
C
      DO J=35,94
        BOX(64,J)=HIGH
      END DO
C
      BOX(45,45)=HIGH
      BOX(84,45)=HIGH
      BOX(45,84)=HIGH
      BOX(84,84)=HIGH
      RETURN
      END
```

FOURFILT.FOR

```
      PROGRAM FOURFILT
C     ================
C
C     FOURIER FILTER AN INPUT IMAGE WITH A GIVEN FILTER IMAGE
C
      LOGICAL RTFLAG, FTFLAG, FIFLAG
      CHARACTER*70 CHARBUF, CDELT1, CDELT2, CRPIX2, CRVAL2
      INTEGER INPOINT, INSIZE(2), RAW1POINT, RAW2POINT, RAWSIZE(2),
     *        OUTPOINT, OUTSIZE(2), FILPOINT, FILSIZE
      INCLUDE 'STARDIR:ERRPAR.FOR'
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
        CALL WRUSER ('ENTER INPUT IMAGE NAME',ISTAT)
        CALL RDIMAG ('IMAGEIN',FMT_R,2,INSIZE,NDIM,INPOINT,ISTAT)
        IF ((ISTAT .NE. ERR_NORMAL) .OR. (NDIM .NE. 2)) THEN
          CALL WRERR ('BADIN')
          CALL EXIT
        ENDIF
        CALL RDDSCR ('IMAGEIN','CDELT1',1,CDELT1,JUNK,ISTAT)
        IF (ISTAT .EQ. ERR_DSCNFR) THEN
            CALL RDKEYR ('DX',.FALSE.,1,DX,JUNK,ISTAT)
          ELSE
            CALL CTOR (CDELT1,DX,ISTAT)
        ENDIF
        CALL RDDSCR ('IMAGEIN','CDELT2',1,CDELT2,JUNK,ISTAT)
        IF (ISTAT .EQ. ERR_DSCNFR) THEN
            CALL RDKEYR ('DY',.FALSE.,1,DY,JUNK,ISTAT)
          ELSE
            CALL CTOR (CDELT2,DY,ISTAT)
        ENDIF
        WRITE (CHARBUF,'(35H X AND Y DIMENSIONS OF INPUT IMAGE:,2I5)')
     *          INSIZE
        CALL WRUSER (CHARBUF,ISTAT)
        RAWSIZE(1) = IFIX(ALOG(FLOAT(INSIZE(1)))/ALOG(2.0) + 0.999)
        RAWSIZE(2) = IFIX(ALOG(FLOAT(INSIZE(2)))/ALOG(2.0) + 0.999)
        CALL WRUSER ('ENTER TRANSFORM DIMENSIONS (2^M * 2^N)',ISTAT)
        CALL RDKEYI ('M',.TRUE.,1,RAWSIZE(1),JUNK,ISTAT)
        CALL RDKEYI ('N',.TRUE.,1,RAWSIZE(2),JUNK,ISTAT)
        CALL WRUSER ('DO YOU WANT RAW TRANSFORM IMAGE? (T/F)',ISTAT)
        RTFLAG = .FALSE.
        CALL RDKEYL ('RTFLAG',.TRUE.,1,RTFLAG,JUNK,ISTAT)
        CALL WRUSER ('DO YOU WANT FILTERED TRANSFORM IMAGE? (T/F)',
     *                ISTAT)
        FTFLAG = .FALSE.
        CALL RDKEYL ('FTFLAG',.TRUE.,1,FTFLAG,JUNK,ISTAT)
        CALL WRUSER ('DO YOU WANT FILTERED OUTPUT IMAGE? (T/F)',ISTAT)
        FIFLAG = .TRUE.
        CALL RDKEYL ('FIFLAG',.TRUE.,1,FIFLAG,JUNK,ISTAT)
        RAWSIZE(1) = 2 ** RAWSIZE(1)
        RAWSIZE(2) = 2 ** RAWSIZE(2)
        CALL GETDYN ('RAW1',FMT_R,RAWSIZE(1)*RAWSIZE(2),RAW1POINT,ISTAT)
        CALL GETDYN ('RAW2',FMT_R,RAWSIZE(2)*(RAWSIZE(1)+2),RAW2POINT,
     *               ISTAT)
        CALL GETDATA (%VAL(INPOINT),INSIZE(1),INSIZE(2),%VAL(RAW1POINT),
     *                RAWSIZE(1),RAWSIZE(2))
        CALL FWD_TRANSFORM (%VAL(RAW1POINT),RAWSIZE(1),RAWSIZE(2),
     *                      %VAL(RAW2POINT),RAWSIZE(2),RAWSIZE(1)+2)
        DU = 1.0 / (DX * FLOAT(RAWSIZE(1)))
        DV = 1.0 / (DY * FLOAT(RAWSIZE(2)))
        CALL RDDSCR ('IMAGEIN','CRVAL2',1,CRVAL2,JUNK,ISTAT)
        IF (ISTAT .NE. ERR_DSCNFR) THEN
          CALL RDDSCR ('IMAGEIN','CRPIX2',1,CRPIX2,JUNK,ISTAT)
```

```
          CALL CTOR (CRVAL2,RLAT,ISTAT)
          CALL CTOI (CRPIX2,ILAT,ISTAT)
          TLAT = RLAT + FLOAT(INSIZE(2) - ILAT) * DY
          BLAT = RLAT - FLOAT(ILAT - 1) * DY
          IF (TLAT*BLAT .GT. 0.0) THEN
            DU = DU / COSD(MIN(ABS(TLAT),ABS(BLAT)))
          ENDIF
        ENDIF
        IF (RTFLAG) THEN
          OUTSIZE(1) = RAWSIZE(1) + 1
          OUTSIZE(2) = RAWSIZE(2) + 1
          CALL WRUSER ('ENTER RAW TRANSFORM OUTPUT IMAGE NAME',ISTAT)
          CALL WRIMAG ('IMAGEOUT',FMT_R,OUTSIZE,2,OUTPOINT,ISTAT)
          CALL PUTTRANS (%VAL(RAW2POINT),RAWSIZE(2),RAWSIZE(1)+2,
     *                   %VAL(OUTPOINT),OUTSIZE(1),OUTSIZE(2),DU,DV)
          CALL FRDATA ('IMAGEOUT',ISTAT)
          CALL CNPAR ('IMAGEOUT',ISTAT)
        ENDIF
        IF (FTFLAG .OR. FIFLAG) THEN
          CALL WRUSER ('ENTER FILTER INPUT IMAGE NAME',ISTAT)
          CALL RDIMAG ('IMAGEFIL',FMT_R,1,FILSIZE,NDIMS,FILPOINT,ISTAT)
          IF ((ISTAT .NE. ERR_NORMAL) .OR. (NDIMS .NE. 1)) THEN
            CALL WRERR ('BADIN')
            CALL EXIT
          ENDIF
          CALL RDDSCR ('IMAGEFIL','CDELT1',1,CDELT1,JUNK,ISTAT)
          IF (ISTAT .NE. ERR_NORMAL) THEN
            CALL WRERR ('BADIN')
            CALL EXIT
          ENDIF
          CALL CTOR (CDELT1,DF,ISTAT)
          CALL FILTER (%VAL(RAW2POINT),RAWSIZE(2),RAWSIZE(1)+2,DU,DV,
     *                 %VAL(FILPOINT),FILSIZE,DF)
          IF (FTFLAG) THEN
            OUTSIZE(1) = RAWSIZE(1) + 1
            OUTSIZE(2) = RAWSIZE(2) + 1
            CALL WRUSER ('ENTER FILTERED TRANSFORM OUTPUT IMAGE NAME',
     *                   ISTAT)
            CALL WRIMAG ('IMAGEOUT',FMT_R,OUTSIZE,2,OUTPOINT,ISTAT)
            CALL PUTTRANS (%VAL(RAW2POINT),RAWSIZE(2),RAWSIZE(1)+2,
     *                     %VAL(OUTPOINT),OUTSIZE(1),OUTSIZE(2),DU,DV)
            CALL FRDATA ('IMAGEOUT',ISTAT)
            CALL CNPAR ('IMAGEOUT',ISTAT)
          ENDIF
          IF (FIFLAG) THEN
            CALL REV_TRANSFORM (%VAL(RAW2POINT),RAWSIZE(2),RAWSIZE(1)+2,
     *                          %VAL(RAW1POINT),RAWSIZE(1),RAWSIZE(2))
            CALL WRUSER ('ENTER FILTERED OUTPUT IMAGE NAME',ISTAT)
            CALL WRIMAG ('IMAGEOUT',FMT_R,INSIZE,2,OUTPOINT,ISTAT)
            CALL CYDSCR ('IMAGEIN','IMAGEOUT',ISTAT)
            CALL PUTDATA (%VAL(RAW1POINT),RAWSIZE(1),RAWSIZE(2),
     *                    %VAL(OUTPOINT),INSIZE(1),INSIZE(2))
            CALL FRDATA ('IMAGEOUT',ISTAT)
          ENDIF
        ENDIF
        CALL EXIT
      END
      SUBROUTINE GETDATA (ARRAYI,MI,NI,ARRAYO,MO,NO)
C     ==========
C
C     PAD OUT DATA WITH INTERPOLATED VALUES AND MULTPLY
C     BY CHECKERBOARD OF +/-1'S TO SHIFT SPECTRUM
```

```
C
      REAL ARRAYI(MI,NI), ARRAYO(MO,NO), FIRST(2048), LAST(2048),
     *     SLOPE, INTERCEPT
C
      CHECK(I,J) = FLOAT(1 - 2 * MOD(I+J,2))
C
        J = 1
        DO I = 1 , MI
          ARRAYO(I,J) = CHECK(I,J) * ARRAYI(I,J)
          FIRST (I) = ARRAYI(I,J)
        ENDDO
        IF (MO .GT. MI) THEN
          INTERCEPT = ARRAYI(IM,J)
          SLOPE = (ARRAYI(1,J) - ARRAYI(IM,J))/FLOAT(MO-MI+1)
          DO I = MI + 1 , MO
            FIRST(I) = INTERCEPT + SLOPE * FLOAT(I-MI)
            ARRAYO(I,J) = CHECK(I,J) * FIRST(I)
          ENDDO
        ENDIF
        DO J = 2 , NI - 1
          DO I = 1 , MI
            ARRAYO(I,J) = CHECK(I,J) * ARRAYI(I,J)
          ENDDO
          IF (MO .GT. MI) THEN
            INTERCEPT = ARRAYI(IM,J)
            SLOPE = (ARRAYI(1,J) - ARRAYI(IM,J))/FLOAT(MO-MI+1)
            DO I = MI + 1 , MO
              ARRAYO(I,J) = CHECK(I,J) * (INTERCEPT + SLOPE*FLOAT(I-MI))
            ENDDO
          ENDIF
        ENDDO
        J = NI
        DO I = 1 , MI
          ARRAYO(I,J) = CHECK(I,J) * ARRAYI(I,J)
          LAST (I) = ARRAYI(I,J)
        ENDDO
        IF (MO .GT. MI) THEN
          INTERCEPT = ARRAYI(IM,J)
          SLOPE = (ARRAYI(1,J) - ARRAYI(IM,J))/FLOAT(MO-MI+1)
          DO I = MI + 1 , MO
            LAST(I) = INTERCEPT + SLOPE * FLOAT(I-MI)
            ARRAYO(I,J) = CHECK(I,J) * FIRST(I)
          ENDDO
        ENDIF
        IF (NO .GT. NI) THEN
          DJ = FLOAT(NO - NI + 1)
          DO J = NI + 1 , NO
            Y = FLOAT(J - NI) / DJ
            DO I = 1 , MO
              ARRAYO(I,J) = CHECK(I,J) * (LAST(I)+(FIRST(I)-LAST(I))*Y)
            ENDDO
          ENDDO
        ENDIF
        RETURN
      END
      SUBROUTINE PUTTRANS (ARRAYI,MI,NI,ARRAYO,MO,NO,DU,DV)
C     ====================
C
C     OUTPUT MAGNITUDE OF TRANSFORM
C
      CHARACTER*20 BUFFER
      REAL ARRAYI(MI,NI), ARRAYO(MO,NO), RE, IM
```

```
C
        CALL ITOC (MO/2+1,BUFFER,ISTAT)
        CALL WRDSCR ('IMAGEOUT','CRPIX1',BUFFER,1,ISTAT)
        CALL ITOC (NO/2+1,BUFFER,ISTAT)
        CALL WRDSCR ('IMAGEOUT','CRPIX2',BUFFER,1,ISTAT)
        CALL RTOC (0.0,BUFFER,ISTAT)
        CALL WRDSCR ('IMAGEOUT','CRVAL1',BUFFER,1,ISTAT)
        CALL WRDSCR ('IMAGEOUT','CRVAL2',BUFFER,1,ISTAT)
        CALL RTOC (DU,BUFFER,ISTAT)
        CALL WRDSCR ('IMAGEOUT','CDELT1',BUFFER,1,ISTAT)
        CALL RTOC (DV,BUFFER,ISTAT)
        CALL WRDSCR ('IMAGEOUT','CDELT2',BUFFER,1,ISTAT)
        DO J = 1 , NO - 1
          JJ = NO + 1 - J
          DO I = 1 , MO/2 + 1
            RE = ARRAYI(J,I*2-1)
            IM = ARRAYI(J,I*2)
            TEMP = SQRT(RE*RE + IM*IM)
            ARRAYO(I,J) = TEMP
            ARRAYO(MO+1-I,JJ) = TEMP
          ENDDO
        ENDDO
        DO I = 1 , MO/2 + 1
          RE = ARRAYI(1,I*2-1)
          IM = ARRAYI(1,I*2)
          TEMP = SQRT(RE*RE + IM*IM)
          ARRAYO(I,NO) = TEMP
          ARRAYO(MO+1-I,1) = TEMP
        ENDDO
        RETURN
      END
      SUBROUTINE PUTDATA (ARRAYI,MI,NI,ARRAYO,MO,NO)
C     ==========
C
C     WRITE OUT DATA WITHOUT INTERPOLATED GUARD BAND AND
C     CHECKERBOARD EFFECT
C
      REAL ARRAYI(MI,NI), ARRAYO(MO,NO)
C
      CHECK(I,J) = FLOAT(1 - 2 * MOD(I+J,2))
C
        DO J = 1 , NO
          DO I = 1 , MO
            ARRAYO(I,J) = CHECK(I,J) * ARRAYI(I,J)
          ENDDO
        ENDDO
        RETURN
      END
      SUBROUTINE FWD_TRANSFORM (AIN, MI, NI, AOUT, MO, NO)
C     ==========
C
C     PERFORM 2D FOURIER TRANSFORM ON REAL DATA TO PRODUCE
C     HALF THE HERMITIAN RESULT
C
      CHARACTER*70 CHARBUF
      INTEGER RWPOINT, IWPOINT, IBRPOINT
      REAL AIN(MI,NI), AOUT(MO,NO)
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
        CALL GETDYN ('RW',FMT_R,MI+1,RWPOINT,ISTAT)
        CALL GETDYN ('IW',FMT_R,MI+1,IWPOINT,ISTAT)
        CALL GETDYN ('IBR',FMT_SL,MI,IBRPOINT,ISTAT)
```

```
      CALL SETUP (%VAL(RWPOINT),%VAL(IWPOINT),%VAL(IBRPOINT),MI,-1.0)
      CONST = 2.0 * SQRT(FLOAT(MI)) * SQRT(FLOAT(NI))
      DO J = 1 , NI/2
        CALL FFT (AIN(1,2*J-1),AIN(1,2*J),%VAL(RWPOINT),%VAL(IWPOINT),
     *            %VAL(IBRPOINT),MI)
        I = 1
        AOUT(2*J-1,2*I-1) = 2.0 * AIN(I,2*J-1) / CONST
        AOUT(2*J-1,2*I) = 0.0
        AOUT(2*J,2*I-1) = 2.0 * AIN(I,2*J) / CONST
        AOUT(2*J,2*I) = 0.0
        DO I = 2 , MI/2 + 1
          AOUT(2*J-1,2*I-1) = (AIN(I,2*J-1) + AIN(MI+2-I,2*J-1))/CONST
          AOUT(2*J-1,2*I) = (AIN(I,2*J) - AIN(MI+2-I,2*J)) / CONST
          AOUT(2*J,2*I-1) = (AIN(I,2*J) + AIN(MI+2-I,2*J)) / CONST
          AOUT(2*J,2*I) = - (AIN(I,2*J-1) - AIN(MI+2-I,2*J-1)) / CONST
        ENDDO
        WRITE (CHARBUF,'(5H ROW ,I4,12H TRANSFORMED)') J
        CALL WRUSER (CHARBUF,ISTAT)
      ENDDO
      CALL FRDATA ('RW',ISTAT)
      CALL FRDATA ('IW',ISTAT)
      CALL FRDATA ('IBR',ISTAT)
      CALL GETDYN ('RW',FMT_R,NI+1,RWPOINT,ISTAT)
      CALL GETDYN ('IW',FMT_R,NI+1,IWPOINT,ISTAT)
      CALL GETDYN ('IBR',FMT_SL,NI,IBRPOINT,ISTAT)
      CALL SETUP (%VAL(RWPOINT),%VAL(IWPOINT),%VAL(IBRPOINT),NI,-1.0)
      DO J = 1 , NO/2
        CALL FFT (AOUT(1,2*J-1),AOUT(1,2*J),%VAL(RWPOINT),
     *            %VAL(IWPOINT),%VAL(IBRPOINT),NI)
        WRITE (CHARBUF,'(8H COLUMN ,I4,12H TRANSFORMED)') J
        CALL WRUSER (CHARBUF,ISTAT)
      ENDDO
      CALL FRDATA ('RW',ISTAT)
      CALL FRDATA ('IW',ISTAT)
      CALL FRDATA ('IBR',ISTAT)
      RETURN
    END
    SUBROUTINE REV_TRANSFORM (AIN, MI, NI, AOUT, MO, NO)
C   ========================
C
C   PERFORM 2D FOURIER TRANSFORM ON HALF A HERMITIAN FUNCTION
C   TO PRODUCE REAL RESULT
C
    CHARACTER*70 CHARBUF
    INTEGER RWPOINT, IWPOINT, IBRPOINT
    REAL AIN(MI,NI), AOUT(MO,NO)
    INCLUDE 'STARDIR:FMTPAR.FOR'
C
      CALL GETDYN ('RW',FMT_R,NO+1,RWPOINT,ISTAT)
      CALL GETDYN ('IW',FMT_R,NO+1,IWPOINT,ISTAT)
      CALL GETDYN ('IBR',FMT_SL,NO,IBRPOINT,ISTAT)
      CALL SETUP (%VAL(RWPOINT),%VAL(IWPOINT),%VAL(IBRPOINT),NO,1.0)
      DO J = 1 , NI/2
        CALL FFT (AIN(1,2*J-1),AIN(1,2*J),%VAL(RWPOINT),%VAL(IWPOINT),
     *            %VAL(IBRPOINT),MI)
        WRITE (CHARBUF,'(8H COLUMN ,I4,20H REVERSE TRANSFORMED)') J
        CALL WRUSER (CHARBUF,ISTAT)
      ENDDO
      CALL FRDATA ('RW',ISTAT)
      CALL FRDATA ('IW',ISTAT)
      CALL FRDATA ('IBR',ISTAT)
      CALL GETDYN ('RW',FMT_R,MO+1,RWPOINT,ISTAT)
```

```
      CALL GETDYN ('IW',FMT_R,MO+1,IWPOINT,ISTAT)
      CALL GETDYN ('IBR',FMT_SL,MO,IBRPOINT,ISTAT)
      CALL SETUP (%VAL(RWPOINT),%VAL(IWPOINT),%VAL(IBRPOINT),MO,1,0)
      CONST = SQRT(FLOAT(MO)) * SQRT(FLOAT(NO))
      DO J = 1 , NO/2
        I = 1
        AOUT(I,2*J-1) = (AIN(2*J-1,2*I-1) - AIN(2*J,2*I)) / CONST
        AOUT(I,2*J) = (AIN(2*J-1,2*I) + AIN(2*J,2*I-1)) / CONST
        DO I = 2 , MO/2 + 1
          AOUT(I,2*J-1) = (AIN(2*J-1,2*I-1) - AIN(2*J,2*I)) / CONST
          AOUT(I,2*J) = (AIN(2*J-1,2*I) + AIN(2*J,2*I-1)) / CONST
          AOUT(MO+2-I,2*J-1) = (AIN(2*J-1,2*I-1) + AIN(2*J,2*I))/CONST
          AOUT(MO+2-I,2*J) = (AIN(2*J,2*I-1) - AIN(2*J-1,2*I)) / CONST
        ENDDO
        CALL FFT (AOUT(1,2*J-1),AOUT(1,2*J),%VAL(RWPOINT),
     *            %VAL(IWPOINT),%VAL(IBRPOINT),MO)
      WRITE (CHARBUF,'(5H ROW ,I4,20H REVERSE TRANSFORMED)') J
      CALL WRUSER (CHARBUF,ISTAT)
      ENDDO
      CALL FRDATA ('RW',ISTAT)
      CALL FRDATA ('IW',ISTAT)
      CALL FRDATA ('IBR',ISTAT)
      RETURN
    END
    SUBROUTINE FILTER (TRNSFRM,MT,NT,DU,DV,FILT,MF,DF)
C   ==================
C
    REAL TRNSFRM(MT,NT), FILT(MF)
C
      DO J = 1 , NT - 1 , 2
        U = FLOAT((NT-J)/2) * DU
        U = U * U
        V = FLOAT(MT/2) * DV
        V = V * V
        KF = NINT(SQRT(U + V)/DF) + 1
        KF = MIN(KF,MF)
        F = FILT(KF)
        TRNSFRM(1,J) = TRNSFRM(1,J) * F
        TRNSFRM(1,J+1) = TRNSFRM(1,J+1) * F
        DO I = 1 , MT/2
          II = MT/2 + I
          III = MT/2 + 2 - I
          V = FLOAT(I-1) * DV
          V = V * V
          KF = NINT(SQRT(U + V)/DF) + 1
          KF = MIN(KF,MF)
          F = FILT(KF)
          TRNSFRM(II,J) = TRNSFRM(II,J) * F
          TRNSFRM(II,J+1) = TRNSFRM(II,J+1) * F
          IF (I .NE. 1) THEN
            TRNSFRM(III,J) = TRNSFRM(III,J) * F
            TRNSFRM(III,J+1) = TRNSFRM(III,J+1) * F
          END IF
        ENDDO
      ENDDO
      RETURN
    END
    INTEGER FUNCTION BIT_REVERSE (I)
C   ============================
C
    COMMON /CFFT/ NU
C
```

```
        K = 0
        IT = I
        DO L = 1 , NU
          II = IT / 2
          K = K*2+(IT-2*II)
          IT = II
        ENDDO
        BIT_REVERSE = K
        RETURN
      END
      SUBROUTINE SETUP (RW,IW,IBR,N,SIGN)
C     ==================
C
      INTEGER IBR(N), B_R, BIT_REVERSE
      REAL RW(*),IW(*)
      COMMON /CFFT/ NU
C
        NU = 0
C       WHILE (2**NU) <> N DO
   10     IF (2**NU .EQ. N) GO TO 20
          NU = NU + 1
          GO TO 10
C       END WHILE
   20   RN = FLOAT(N)
        DO K = 0 , N/4
          COSINE = COS(6.283185*FLOAT(K)/RN)
          RW (K + 1)                 =  COSINE
          RW (N / 2 - K + 1)      = -COSINE
          RW (N / 2 + K + 1)      = -COSINE
          RW (N - K + 1)             =  COSINE
          IW (N / 4 - K + 1)      =  COSINE * SIGN
          IW (N / 4 + K + 1)      =  COSINE * SIGN
          IW (N * 3 / 4 - K + 1) = -COSINE * SIGN
          IW (N * 3 / 4 + K + 1) = -COSINE * SIGN
        ENDDO
        DO K = 0 , N - 1
          B_R = BIT_REVERSE(K)
          IBR(K + 1) = B_R
          IF (B_R .LT. K) THEN
            TR            = RW (K + 1)
            TI            = IW (K + 1)
            RW (K + 1)    = RW (B_R + 1)
            IW (K + 1)    = IW (B_R + 1)
            RW (B_R + 1) = TR
            IW (B_R + 1) = TI
          ENDIF
        ENDDO
        RETURN
      END
      SUBROUTINE FFT (R,I,RW,IW,IBR,N)
C     ==============
C
      INTEGER IBR(N), B_R, COMP
      REAL R(*), I(*) ,RW(*), IW(*)
      COMMON /CFFT/ NU
C
        N_SKIP = 1
        K_SKIP = N
        DO COMP = 1 , NU
          M_SKIP = K_SKIP / 2
          NW = 1
          DO I_SKIP = 1 , N_SKIP
```

```
        COSINE = RW (NW)
        SINE   = IW (NW)
        NW = NW + 2
        DO I_SAM = (I_SKIP - 1) * K_SKIP + 1 ,
*                  (I_SKIP - 1) * K_SKIP + M_SKIP
          J_SAM = I_SAM + M_SKIP
          TR = R (J_SAM) * COSINE - I (J_SAM) * SINE
          TI = R (J_SAM) * SINE + I (J_SAM) * COSINE
          R (J_SAM) = R (I_SAM) - TR
          I (J_SAM) = I (I_SAM) - TI
          R (I_SAM) = R (I_SAM) + TR
          I (I_SAM) = I (I_SAM) + TI
        ENDDO
      ENDDO
      K_SKIP = M_SKIP
      N_SKIP = N_SKIP * 2
    ENDDO
    DO K = 0 , N - 1
      B_R = IBR(K+1)
      IF (K .LT. B_R) THEN
        TR           = R (K + 1)
        TI           = I (K + 1)
        R (K + 1)    = R (B_R + 1)
        I (K + 1)    = I (B_R + 1)
        R (B_R + 1) = TR
        I (B_R + 1) = TI
      ENDIF
    ENDDO
    RETURN
  END
```

CHINHAT.FOR

```
      PROGRAM GENERATE_FILTER
C     =======================
C
C This generates a 'chinese hat' filter profile for use
C on test pictures.It is an approximation of the type of
C smoothing caused by a real camera aperture.
C The program is based on program SHARP.FOR on STARLINK.
C
C
C
C       Written   02/10/1986
C
      CHARACTER*70 CHARBUF
      INTEGER SIZE,FILTPOINT
      REAL CUTOFF,DR
      INCLUDE 'STARDIR:ERRPAR.FOR'
      INCLUDE 'STARDIR:FMTPAR.FOR'
C
      SIZE=256
      CALL WRIMAG('FILTER',FMT_R,SIZE,1,FILTPOINT,ISTAT)
      CUTOFF=0.5
      CALL RDKEYR('CUTOFF',.TRUE.,1,CUTOFF,JUNK,ISTAT)
      DR=0.5/FLOAT(SIZE-2)
      CALL RTOC(DR,CHARBUF,ISTAT)
      CALL WRDSCR('FILTER','CDELT1',CHARBUF,1,ISTAT)
      CALL GENERATE(%VAL(FILTPOINT),SIZE,DR,CUTOFF)
      CALL FRDATA('FILTER',ISTAT)
      CALL EXIT
      END
C
C
      SUBROUTINE GENERATE(FILTER,SIZE,DR,CUTOFF)
C     ==========================
C
C This generates the filter profile
C
      INTEGER SIZE
      REAL CUTOFF,DR,FILTER(SIZE)
      REAL A,B,C
      DATA PI/3.141593/
C
      A=CUTOFF/2
      DO I=1,SIZE
        B=0.0
        C=0.0
        IF (DR*FLOAT(I-1) .LE. CUTOFF) THEN
            B=2*ACOS(DR*FLOAT(I-1)/CUTOFF)
            C=(DR*FLOAT(I-1)/A)*SQRT(1.0-(DR*FLOAT(I-1)/CUTOFF)**2)
            FILTER(I)=(B-C)/PI
          ELSE
            FILTER(I)=0.0
        END IF
      END DO
      RETURN
      END
```

ROUNDOFF.FOR

```
      PROGRAM ROUND_IMAGE
C     ===================
C
C This program can be used to round-off REAL grey-level
C images to INTEGER ones.Digital grey-level lookup tables
C only display INTEGER images, thus automatically rounding
C off REAL ones.However, it is useful to perform this
C rounding seperately in order to obtain the correct
C results with ,e.g.,STATS.
C
C
C
C          Written   02/10/1986
C
      INTEGER INPOINT,OUTPOINT,SIZE(2)
      INCLUDE 'STARDIR:FMTPAR.FOR'
      INCLUDE 'STARDIR:ERRPAR.FOR'
C
      CALL RDIMAG('INPUT',FMT_R,2,SIZE,NDIM,INPOINT,ISTAT)
C
      CALL WRIMAG('OUTPUT',FMT_SL,SIZE,2,OUTPOINT,ISTAT)
C
      CALL REAL_TO_INT(%VAL(INPOINT),SIZE,%VAL(OUTPOINT))
C
      CALL EXIT
      END
C
C
      SUBROUTINE REAL_TO_INT(IN,SZ,OUT)
C     =================================
C
C This converts the REAL image to an INTEGER image.
C
      INTEGER SZ(2),OUT(SZ(1),SZ(2))
      REAL IN(SZ(1),SZ(2))
C
      DO J=1,SZ(2)
        DO I=1,SZ(1)
          OUT(I,J)=NINT(IN(I,J))
        END DO
      END DO
      RETURN
      END
```

NOISE.FOR

```
      PROGRAM GAUSSIAN_NOISE
C     ======================
C
C This program converts the uniformly distributed random
C numbers generated by the FORTRAN 'RAN()' function into
C normally distributed random numbers.As 'seed' for the
C RAN() function, the current time is used.The resulting
C numbers are rounded to integers for use on grey-scale
C images as Gaussian noise of known mean and standard
C deviation.The standard deviation is user-selected with
C a default of 1.The mean is zero, but can be scaled using
C 'CADD' on Starlink.A 'noise-image' file of size 128 X 128
C pixels is created.This can be added to images of the same
C dimensions using 'ADD'.
C
C
C
C        Written   09/09/1986
C        Modified  16/09/1986
C
      INTEGER SEED,SIZE(2),SZ,OUTPOINT,SD
      REAL RNUM(128,128)
      INCLUDE 'STARDIR:FMTPAR.FOR'
      INCLUDE 'STARDIR:ERRPAR.FOR'
C
C Set the size of the output image
C
      SIZE(1)=128
      SIZE(2)=128
C
C Write output image file
C
      CALL WRIMAG('OUTPUT',FMT_SL,SIZE,2,OUTPOINT,ISTAT)
C
C Set standard deviation
C
      SD=1
      CALL RDKEYI('SDEV',.TRUE.,1,SD,JUNK,ISTAT)
C
C Generate the seed
C
      CALL GENERATE(SEED)
C
C Generate the uniform random numbers
C
      CALL UNIFORM(RNUM,SEED)
C
C Generate the Gaussian random integers & write to output
C
      CALL GAUSS(RNUM,SD,%VAL(OUTPOINT))
C
C Exit to system
C
      CALL EXIT
      END
C
C
      SUBROUTINE GENERATE(SEED)
C     =========================
C
C This routine generates a random 'seed' for the FORTRAN function
C RAN() based on current time.The subroutine itself is based on an
```

```
C integer function, FUNCTION SEED, by J.L.Jonas of the Department
C of Physics and Electronics, Rhodes University.
C
      INTEGER SEED,IH,IM,IS
      CHARACTER*10 T
C
      CALL TIME(T)
      READ (T,'(I2,1X,I2,1X,I2)')IH,IM,IS
      SEED=(IH*10000+IM*100+IS)*2+1
      RETURN
      END
C
C
      SUBROUTINE UNIFORM(RNUM,SEED)
C     ==================
C
C This routine generates uniformly distributed random numbers
C between 0 and 1 using the RAN() function.
C
      INTEGER SEED
      REAL RNUM(128,128)
C
      DO J=1,128
        DO I=1,128
          RNUM(I,J)=RAN(SEED)
        END DO
      END DO
      RETURN
      END
C
C
      SUBROUTINE GAUSS(RNUM,SD,GROUT)
C     ===============
C
C This routine converts the uniformly distributed random numbers
C previously generated into normally distributed random integers
C suitable for an image file.The method used is described in
C 'Handbook of Mathematical Functions', 1964, Ed. M.Abramowitz &
C I.A.Stegun.The noise has standard deviation SD.
C
      INTEGER GROUT(128,128),SD
      REAL RNUM(128,128),GNUM(128,128),A
      DATA PI/3.14159/
C
C Check that no odd-numbered element is zero
C
      DO J=1,128
        DO I=1,64
          IF (RNUM(2*I-1,J) .EQ. 0.0) THEN
            RNUM(2*I-1,J)=1.0
          END IF
        END DO
      END DO
C
C Create Gaussian noise
C
      A=0.0
      DO J=1,128
        DO I=1,64
          A=SQRT(-2*LOG(RNUM(2*I-1,J)))
          GNUM(2*I-1,J)=A*COS(2*PI*RNUM(2*I,J))
          GNUM(2*I,J)=A*SIN(2*PI*RNUM(2*I,J))
```

```
        END DO
      END DO
C
C Scale to correct s.d. and convert to inteser
C
      DO J=1,128
        DO I=1,128
          GROUT(I,J)=NINT(SD*GNUM(I,J))
        END DO
      END DO
      RETURN
      END
```

SNR.FOR

```
      PROGRAM SIGNAL_TO_NOISE
C     ========================
C
C Program created 07/12/1986 to determine the signal-to-noise
C ratio for test images created by TESTBOX.FOR and NOISE.FOR.
C
      INTEGER INPOINT1,INPOINT2,SIZE1(2),SIZE2(2)
      REAL SNR
      INCLUDE 'STARDIR:FMTPAR.FOR'
      INCLUDE 'STARDIR:ERRPAR.FOR'
C
      CALL RDIMAG('INPUT1',FMT_SL,2,SIZE1,NDIM,INPOINT1,ISTAT)
C
      CALL RDIMAG('INPUT2',FMT_SL,2,SIZE2,NDIM,INPOINT2,ISTAT)
C
      CALL RATIO(%VAL(INPOINT1),%VAL(INPOINT2),SIZE1,SNR)
C
      WRITE (6,6000)SNR
C
6000  FORMAT(1H ,'Signal-to-Noise Ratio = ',F8.4)
      END
C
C
      SUBROUTINE RATIO(IN1,IN2,SZ,SNR)
C     ================================
C
      INTEGER SZ(2),IN1(SZ(1),SZ(2)),IN2(SZ(1),SZ(2))
      REAL NSUM,SSUM,SNR
C
      SSUM=0.0
      NSUM=0.0
      DO J=1,SZ(2)
        DO I=1,SZ(1)
          SSUM=SSUM+FLOAT(IN1(I,J)**2)
          NSUM=NSUM+FLOAT(IN2(I,J)**2)
        END DO
      END DO
C
      SNR=-10.0*LOG10(NSUM/SSUM)
      RETURN
      END
```

XCORPIC.FOR

```
      PROGRAM CROSS_CORR_TEST
C     ======================
C
C This program is designed for use with the test picture
C generated by the program TESTBOX.FOR.After the test
C image has been reduced in contrast, smoothed and had
C noise added to it, the various thresholding methods are
C applied to it.The result of thresholding should ideally
C return the original image.It is reasonable, therefore,
C to assume that a method can be evaluated on the basis
C 'how close it comes' to reproducing the original binary
C image.To measure this 'goodness', a cross correlation
C function is used.The correlation between the original
C and final images is found.Further information can be
C found in notes by A.Brink, M.Sc Book 3, Date 04/10/1986
C pp 216-220.
C
C
C        Written   06/10/1986
C
      INTEGER AINPOINT,BINPOINT,ASIZ(2),BSIZ(2)
      REAL CORR
      INCLUDE 'STARDIR:FMTPAR.FOR'
      INCLUDE 'STARDIR:ERRPAR.FOR'
C
      CALL RDIMAG('INPUT1',FMT_SL,2,ASIZ,NDIM,AINPOINT,ISTAT)
C
      CALL RDIMAG('INPUT2',FMT_SL,2,BSIZ,NDIM,BINPOINT,ISTAT)
C
C Note that ASIZ and BSIZ should be equal.
C
      CALL CORRELATE(%VAL(AINPOINT),%VAL(BINPOINT),ASIZ,CORR)
C
C Write the correlation coefficient to screen.
C
      WRITE (6,6000)CORR
C
6000  FORMAT(1H ,'*** Correlation coefficient = ',F10.7,' ***')
C
      CALL EXIT
      END
C
C
      SUBROUTINE CORRELATE(AIN,BIN,SZ,CORR)
C     =====================
C
C This calculates the correlation coefficient for the two
C given images.The value will lie between -1 and +1, with
C 0 indicating no correlation.
C
      INTEGER SZ(2),AIN(SZ(1),SZ(2)),BIN(SZ(1),SZ(2))
      REAL CORR,EX,EX2,EY,EY2,EXY,VX,VY,PROB
C
      PROB=1.0/(SZ(1)*SZ(2))
      EX=0.0
      EX2=0.0
      EY=0.0
      EY2=0.0
      EXY=0.0
      CORR=0.0
      DO J=1,SZ(2)
        DO I=1,SZ(1)
```

```
      EX=EX+(AIN(I,J)*PROB)
      EX2=EX2+((AIN(I,J)**2)*PROB)
      EY=EY+(BIN(I,J)*PROB)
      EY2=EY2+((BIN(I,J)**2)*PROB)
      EXY=EXY+(AIN(I,J)*BIN(I,J)*PROB)
    END DO
END DO
VX=EX2-(EX**2)
VY=EY2-(EY**2)
IF (VX*VY .GT. 0.0) THEN
  CORR=(EXY-EX*EY)/SQRT(VX*VY)
END IF
RETURN
END
```

APPENDIX B

DEC VAX 11/730 RMF OPERATING PROCEDURE

DEC VAX 11/730 RMF OPERATING PROCEDURE

The version of RMF used has been modified for use with the DEC VAX 11/730 (VAX) by Mr J.L. Jonas of the Department of Physics and Electronics at Rhodes University. The following steps serve as a brief guide to using this program to send image data files from an IBM PC microcomputer to the VAX.

(1) Insert the disk containing RMF into drive "A" and the disk containing the image file, "JUNK.PIC", created by VAR6BIT.BAS, into drive "B" of the IBM PC.

(2) Disk drive "A" should be the logged drive (the prompt "A>" should appear). If this is not the case, type "A:" and press <Return>.

(3) Type "RMF" and press <Return>. RMF guides the user through all operations. For a list of available commands, press <?>. These commands are listed here:

B - Set Baud rate

D - Toggle full/half Duplex setting

F - Directory of files on local disk

K - Define a function key

R - Receive files from connected system

S - Send files to connected system

T - Terminal mode

V - Save current program configuration

Q - Quit and return to DOS.

In any of these modes, press <Ctrl> and <\> simultaneously to return to the main menu or to abort any prompt.

(4) Press <T> to enter terminal mode. The IBM PC will function as a VAX computer terminal. Log in as usual by entering the required Username and Password.

(5) Press <Ctrl> and <\> to return to the main menu. The IBM PC will remain logged into the VAX.

(6) Press <S> to send the image file to the VAX. The user is required to enter "Micro file name", "VAX file name" and "Type (B/T)". In this case the micro file name is "B:JUNK.PIC" and the VAX file name can be any desired name appended by ".PIC". "Type" should be specified as binary by pressing <B>. Press <Return> after each response. File transfer commences immediately after the last response and all errors are reported on the screen.

(7) When file transfer is completed, the user is returned to the main menu. If another image file is to be transferred, insert the disk containing the file into drive "B" and repeat step (6). When all required files have been transferred, re-enter terminal mode and log out of the VAX in the usual fashion. Return to the main menu and press <Q> to return to IBM PC DOS.

APPENDIX C

COMPUTATION FORMULAE FOR THE MAXIMUM CORRELATION METHOD

COMPUTATION FORMULAE FOR THE MAXIMUM CORRELATION METHOD

The process of calculating the correlation between the original and bilevel images using the equations of Chapter 3.3.2 is very slow. If the grey-level histograms are used instead of the actual images, these calculations can be performed much more quickly. The correlation criterion is again given by

$$\rho_{xy}(T) = \frac{E_{xy}(T) - E_x E_y(T)}{\sqrt{V_x V_y(T)}} \quad ; \quad 0 \leq T < n \qquad (C.1)$$

where now

$$E_x = \sum_{g=0}^{n} g\, p_g \qquad (C.2.1)$$

$$E_y(T) = \sum_{g=0}^{T} \mu_0(T)\, p_g + \sum_{g=T+1}^{n} \mu_1(T)\, p_g \qquad (C.2.2)$$

$$E_{xy}(T) = \sum_{g=0}^{T} g\, \mu_0(T)\, p_g + \sum_{g=T+1}^{n} g\, \mu_1(T)\, p_g \qquad (C.2.3)$$

and

$$V_x = E_{xx} - [E_x]^2 \qquad (C.3.1)$$

$$V_y(T) = E_{yy}(T) - [E_y(T)]^2 \qquad (C.3.2)$$

where

$$E_{xx} = \sum_{g=0}^{n} g^2\, p_g$$

$$E_{yy}(T) = \sum_{g=0}^{T} \mu_0^2(T)\, p_g + \sum_{g=T+1}^{n} \mu_1^2(T)\, p_g$$

where p_g are the grey-level probabilities (3.1) and $\mu_0(T)$ and $\mu_1(T)$ are the below- and above-threshold means of the histogram. It is not immediately apparent that these computational formulae are equivalent to the equations presented in Chapter 3.3.2. Equivalence can be

shown as follows:

Consider an arbitrary $k \times l$ matrix (image) x_{ij}, where $x_{ij} \varepsilon Z$, $x_{ij} \geq 0$ (Z is the set of all integers). If we take the sum of all elements x_{ij}, we obtain

$$\sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij} = 0.f_0 + 1.f_1 + 2.f_2 + \ldots + n.f_n,\ n \varepsilon Z, n \geq 0 \qquad (C.4)$$

where f_r represents the number of times the integer r occurs, i.e. its frequency. Hence

$$E_x = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij}\,(1/N) = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij}$$

$$= \frac{1}{N}\,(0.f_0 + 1.f_1 + 2.f_2 + \ldots + n.f_n),\ n \varepsilon Z,\ n \geq 0$$

$$= \frac{1}{N} \sum_{g=0}^{n} g\, f_g$$

$$= \sum_{g=0}^{n} g\, p_g$$

where p_g is the probability of the grey-level (integer) g occurring, given by equation (3.1).

Similarly

$$E_y(T) = \sum_{i=1}^{k} \sum_{j=1}^{l} y_{ij}\,(1/N)$$

$$= \sum_{\substack{i=1 \\ x_{ij} \leq T}}^{k} \sum_{J=1}^{l} \mu_0\,(T)\,(1/N) + \sum_{\substack{i=1 \\ x_{ij} > T}}^{k} \sum_{j=1}^{l} \mu_1\,(T)\,(1/N)$$

$$= \sum_{g=0}^{T} \mu_0 (T)\, p_g + \sum_{g=T+1}^{n} \mu_1 (T)\, p_g$$

The rest of the formulae follow logically from these.