Image Analysis and Segmentation Based on the Circular Pipeline Video Processor

Jon M. Albritton
University of Central Florida

IMAGE ANALYSIS AND SEGMENTATION BASED ON THE CIRCULAR PIPELINE VIDEO PROCESSOR

BY

JON M. ALBRITTON, JR.
B.S.E., University of Central Florida, 1982

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program of the
College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1984

ABSTRACT

Visual inspection of printed circuit boards has generally depended on human inspectors. However, a system has been developed which allows for automated visual inspection using robotics and modern image processing techniques. This paper first introduces automatic visual inspection processes, overviews the Automatic Board Assembly, Inspection and Test (ABAIT) system, reviews image processing concepts and describes the Circular Pipeline Video Processor (CPVP). Image data from the CPVP is analyzed and an investigation into alternate segmentation algorithms to identify circuit board features is presented. The relative performance of these algorithms is compared and conclusions drawn.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to those who have helped make my graduate studies and this paper possible.

First and foremost is Martin Marietta Orlando Aerospace. It was through the cooperation of Martin Marietta and the University of Central Florida College of Engineering that the Industrial Associates Graduate Work/Study Program was established and I was able to pursue a graduate degree on a full-time basis. In particular I wish to thank Bob Pettigrew who had faith in me from beginning to end, Stan Buchanan who as my supervisor graciously worked around my schedule, Andy Abercrombie and Scott Carruth, my tutors and co-workers, and especially Ed Soniat Dufossat who unselfishly bailed me out on more than one occasion. Walter Eggerton and Phil Bruce provided support as my functional managers.

I wish to express my gratitude to those at UCF who have earned my respect and made the graduate program a pleasure, difficult but still a pleasure. They are Dr. Madjid Belkerdid, Dr. Richard C. Harden, Dr. Fred O. Simons and Dr. Robert Walker.

Special thanks go to Sharon Bronson who typed the manuscript with unwaivering perfectionism, proved to be an indispensable asset and always had a glass of "Pepsi" on hand.

I especially thank my mother. She has believed in me and supported me all of my life and I know she always will.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

AUTOMATED VISUAL INSPECTION

The Need for Automation

In mass production facilities, product inspection is required to insure that the quality standards are being met. Furthermore, products manufactured for military and aerospace applications usually require a 100 percent inspection rate, both while under construction and in their completed state.

Visual inspection of products for the detection of both functional and cosmetic defects is one of the most difficult of the inspection processes. Past methods have usually relied on human inspectors to determine presence or absence of defects. Aids such as a magnifying lens and a video display system can increase efficiency. However, visual inspection is a repetitive, monotonous procedure and studies (Chin and Harlow 1982) show that the accuracy of human visual inspection declines with dull and routine jobs.

Automated visual inspection has justifiable advantages over human inspectors (Chin and Harlow 1982). As pointed out by Chin and Harlow, these advantages include:

1. freeing humans from the dull and routine;
2. saving human labor costs;
3. performing inspection in unfavorable environments;
4. reducing demand for highly skilled human inspectors;

5. analyzing statistics on test information and keeping records for manufacturing decisions; and

6. matching high-speed production with high-speed inspection.

Despite the apparent advantages of automated visual inspection, it has not been aggresively implemented in most production facilities. A major drawback is the lack of general purpose, ready to deliver systems. Most of the present systems are either highly specialized, slow, bulky, expensive, or require large host computers.

Current Inspection System Availability

Simple electro-optical gauging equipment is available commercially (Chin and Harlow 1982). This equipment bases its pass/fail decision on simple measurement techniques operating on an object's boundaries. Simple edge detection circuitry is used on binary images to determine these boundary locations. General purpose visual inspection systems which are software based are also commercially available. The operation of these systems requires a training phase, in which a human operator must teach the system the features and allowable tolerances, and an inspection phase, when the system scans the product to be inspected comparing the measured features against the previously stored limits.

Advances in vision systems in 1983 made systems capable of identifying and processing shades of gray available. These new systems are based on a hardware/software combination. This is an attempt to decrease processing time, yet maintain some flexibility. Ford Motor Company is planning to use a vision system developed by Synthetic Vision Systems of Ann Arbor, Michigan to check electronic circuits in automotive electronic control modules (Kaplan 1984). Control Automation Inc. of Princeton, N.J. has introduced a visual inspection system which is to be used to inspect printed circuit boards to verify that components are properly inserted (Kaplan 1984). Itran Corporation of Manchester, N.H. has developed an automated vision inspection system requiring minimal set up time and little knowledge of computer programming. After being programmed the system is capable of inspecting objects on a production line.

One of the most extensively inspected subassemblies of electronic equipment is the printed circuit board (PCB). Typical flaws consist of conductor-to-conductor spacing, shorts, opens, conductor width, hole size and hole placement. Furthermore, the artwork (photomask) used to produce the PCBs must also be inspected for similar flaws. Though human inspectors can easily locate obvious flaws, small flaws and tolerances in the 2 to 5 mil range are difficult to detect. Automated visual inspection is well suited for these instances.

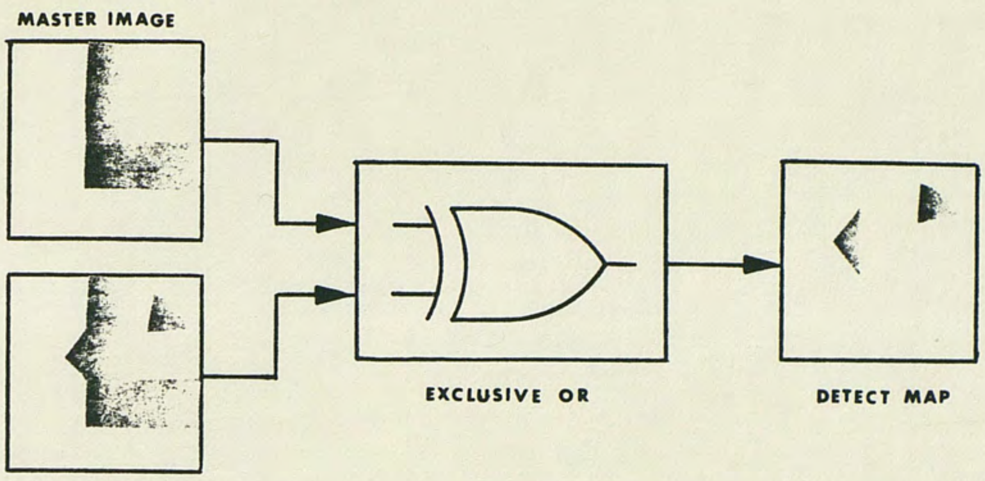
Four Basic Inspection Processes

Four basic approaches to automated visual inspection of PCB's are image subtraction, feature matching, dimensional verification and the syntactic approach. Image subtraction is the most straightforward approach. The basic premise calls for an image of the area under inspection to be compared directly against a perfect image of that area which has previously been stored in a data base. The difference in the two images is analyzed for error information. Implementation schemes using video disc data storage and masked illumination techniques (Chin and Harlow 1982) have been tried, but the basic problems remain. These problems are the requirement of an extremely large data base, precise PCB alignment, constant illumination and accurate scanners.

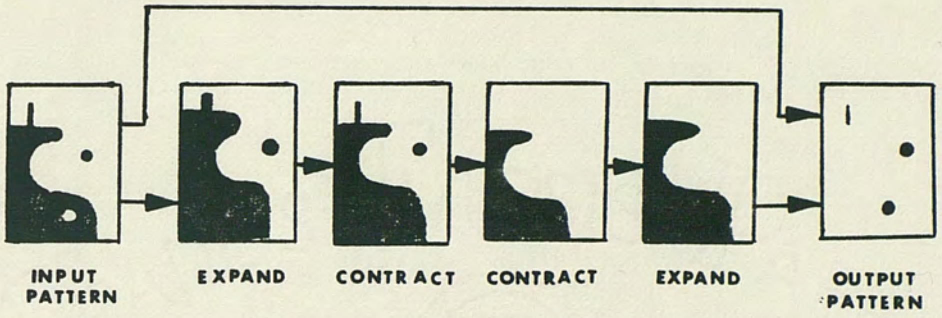
A second approach to PCB inspection is feature matching. Different features such as edges, land placement and holes are extracted and compared against the features extracted from a good PCB. Chin and Harlow (1982) used an interactive system to train an inspection system to look only for specific features on a PCB under test and compare against stored feature data. The requirement for the entire PCB image to be scanned or stored is eliminated, but the system still requires a large data base and is still a comparison based system.

For a more specialized type flaw, the expansion-contraction method can be used. In its operation, areas identified as lands are expanded (enlarged) in all directions. Small gaps become filled in. These same areas are then contracted to a state smaller than the original size. Small artifacts are thus eliminated. After expansion back to the original size, the image subtraction method is used to identify all gaps and artifacts.

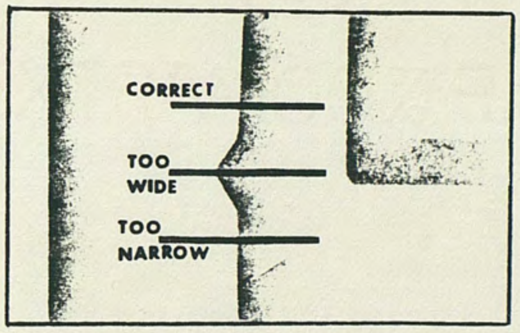
Dimensional verification is straightforward and involves measuring land widths, hole widths, spacing, etc., directly. Problems arise in determining where to make measurements for any given PCB configuration. Large data bases and large processing times are drawbacks to this approach. These first three approaches are illustrated in figure 1.



(a)



(b)



(c)

Fig. 1. Basic approaches to automated visual inspection. (a) Image subtraction (b) The expansion-contraction method (c) Dimensional verification

(Chin and Harlow 1982; redrawn by author)

The syntactic approach uses primitive patterns and structures (i.e. corners, lines, etc.) to describe the PCB. Once the PCB description is complete, the primitives of an area under inspection can be searched for known defects. Some studies have been made in this area (Chin and Harlow 1982), though computational requirements are still large and time consuming.

In summary, a 100 percent inspection rate is often required of today's manufactured products. The visual inspection of printed circuit boards is an area of growing importance and has generally relied on human inspectors. Several approaches have been developed for automated visual inspection processes and are especially suited for PCB inspection.

CHAPTER 2

AUTOMATIC BOARD ASSEMBLY, INSPECTION AND TEST

A System Overview

In an effort to maintain high quality standards and at the same time increase efficiency while decreasing cost, Martin Marietta Orlando Aerospace is developing a computer controlled printed circuit board assembly line termed the Automated Board Assembly, Inspection and Test (ABAIT) system. The system includes raw printed circuit board inspection, component insertion, wave solder and cleaning stations. The part of the system discussed in this report is the visual inspection detail station.

The Visual Inspection Detail Station

The visual inspection detail station consists of both the visual inspection processing equipment and the robot. The layout of the visual inspection station is illustrated in figure 2 and a picture of the station during development and integration is shown in figure 3.

Following the layout of figure 2, typical operation of the detail station is as follows:

1. PCB's to be inspected are placed on table A.
2. The PCB is placed on the feeder tray by the robot and the PCB's bar code is read.
3. The PCB is fed onto the X-Y translation table.

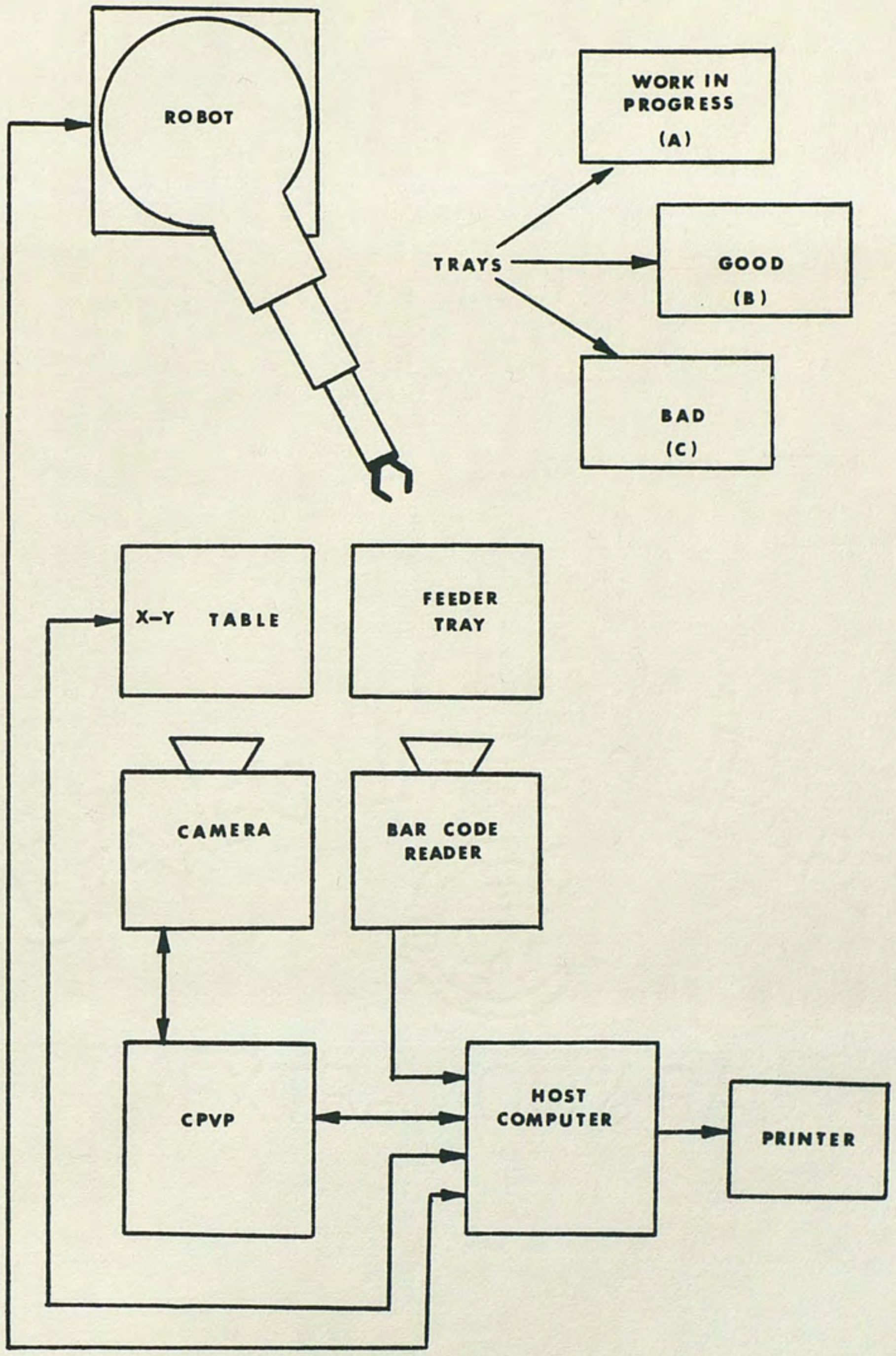


Fig. 2. Visual inspection detail station layout

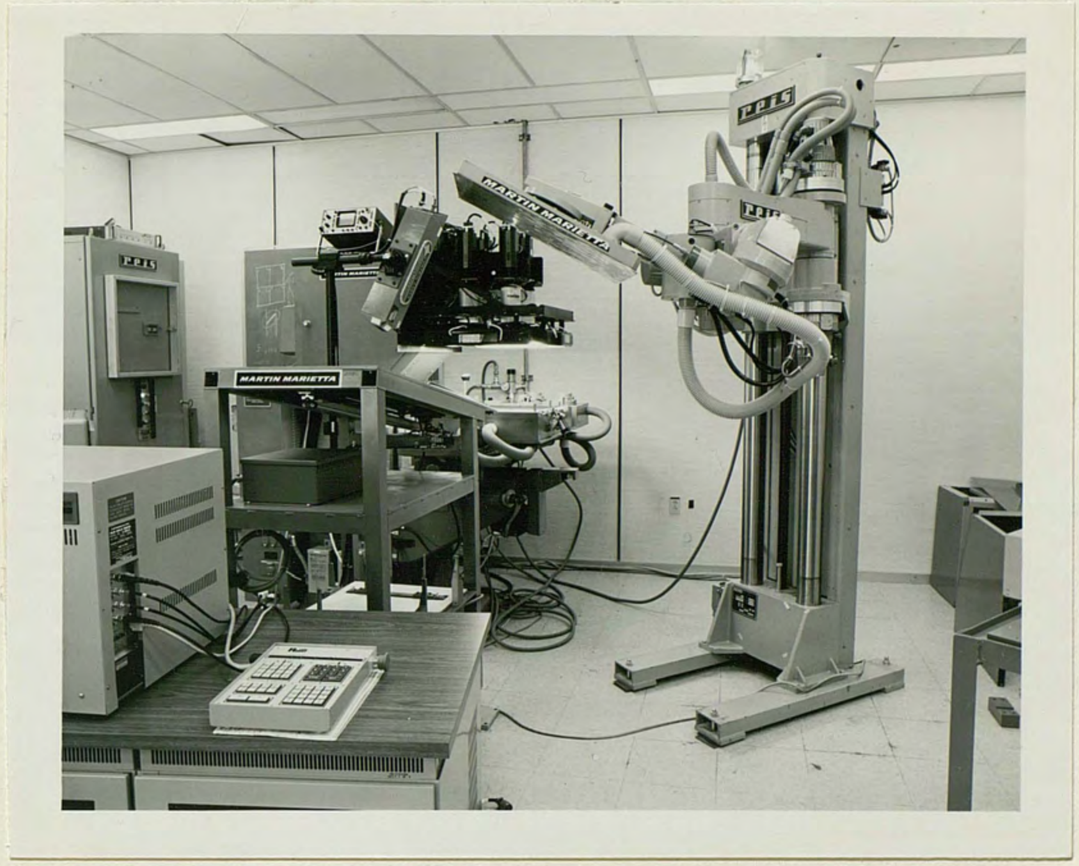


Fig. 3. Visual inspection detail station during system integration

4. The PCB is clamped to the table and the table is stepped through a predetermined sequence.
5. An image of a section of the PCB is obtained at each step by the overhead camera and processed by the image processing equipment.
6. When the inspection is completed, the robot places the PCB onto table B or C, depending on the inspection outcome.
7. Board flaws and their location are printed.
8. The next PCB, having been placed on the feeder tray, is loaded onto the X-Y table and the procedure is repeated.

Objectives of the Inspection Station

The ultimate goal of the visual inspection station is to provide a 100% inspection of the artwork, inner layers of multi-layer PCB's and unpopulated PCB's being produced at the Ocala manufacturing facility. This goal will allow Martin Marietta to find manufacturing defects before they are integrated into major assemblies and thus improve overall product quality. Specific parameters to be inspected for are:

1. Conductor width (minimum)
2. Conductor spacing (minimum)
3. Annular rings
4. Broken conductors
5. Nicks
6. Pinholes
7. Hole sizes

Targeted success rates cal; for a 95% detection rate of detecting actual defects and a maximum error rate of 10% for identifying a flaw when none is present. Furthermore, the system thru-put should be maximized while maintaining a minimum resolution of 0.001", the current industry standard. Eventually, the system is to reliably inspect up to 200,000 PCB's (or artwork) per year with a minimal amount of human intervention.

CHAPTER 3

IMAGE PROCESSING CONCEPTS

Fundamentals of Image Processing

"Picture processing or image processing is concerned with the manipulation and analysis of pictures by computer" (Rosenfeld and Kak 1982). The two main areas of application for image processing are the "...improvement of pictorial information for human interpretation and processing of scene data for autonomous machine perception." (Gonzales 1977). For all cases in this paper, an image is defined by a two-dimensional function f(x,y), where x and y are spatial co-ordinates in the image plane, and the value of the function at any point (x,y) is a measure of the brightness of the image at that point. For monochromatic or black and white images the brightness values are termed gray levels.

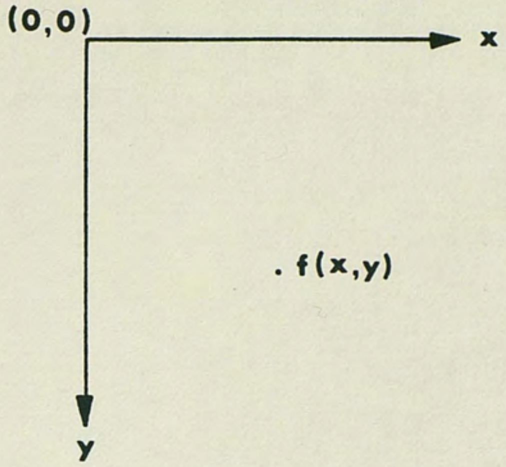
Processing by computer requires digital data. Therefore, images must be digitized before they are of use. A digital image results when a sampling process is used to discretize the image in both spatial co-ordinates and brightness (Gonzales 1977). Samples in the spatial domain are usually taken at a regularly spaced array of points and the sample brightness is usually the image brightness quantized to a set of discrete equally spaced gray levels. A single element of the digitized image is called a pixel, an abbreviation for picture element.

Cartesian co-ordinates are a common choice for spatial representations. This convention is shown in figure 4a. Often the neighborhood of a particular pixel is specified, where a neighborhood is simply a predefined set of pixels in the vicinity of the pixel under scrutiny. A common convention for identifying the co-ordinates of a pixel and its eight closest neighbors follows the Cartesian convention and is shown in figure 4b. This is called a 3x3 neighborhood.

Operations on digital images, whether for enhancement or the extraction of information, fall into one of three main categories; point, local, or geometric operations. In point operations, the output gray level of a pixel depends only on the value of that pixel at the input and the operation performed. For local operations, the output level of a pixel depends only on the input levels of a neighborhood of that point. In geometric operations, the output level of a pixel depends only on the input level(s) of some other point(s) defined by a geometrical transformation (Rosenfeld and Kak 1982). Furthermore, these operations can be combined to meet processing requirements.

Image Enhancement

The improvement of pictorial information is known as image enhancement. Enhancement techniques seek to process an image so that the result is better suited for analysis. It differs from image restoration in that there is no concentrated effort to restore the image to its ideal state. Actually, there is no general standard for image enhancement because there is no general standard for the image quality required for beneficial analysis. The standard is



(a)

(x-1,y-1)	(x,y-1)	(x+1,y-1)
(x-1,y)	(x,y)	(x+1,y)
(x-1,y+1)	(x,y+1)	(x+1,y+1)

(b)

Fig. 4. Cartesian co-ordinate convention. (a) Pixel identification (b) Co-ordinates of a 3x3 neighborhood of the pixel at (x,y)

highly dependent on the particular application. Furthermore, the definition of image enhancement stops short of information extraction (Pratt 1978).

The three common areas of image enhancement are gray scale modification, sharpening, and smoothing. Gray scale modifications can be made directly through contrast manipulations or indirectly through histogram modifications and a subsequent gray level transformation. The direct method is straightforward and can simply consist of multiplying each gray level by the same constant. Alternatively, consider an image quantized to J levels, where J is the integer denoting the maximum number of gray levels, but whose range is a subset of J. Restricting the output to J levels and using a linear mapping scheme, the image range can be enhanced as shown in figure 5. This method assumes non-uniform spacing side-effects are acceptable.

A non-linear enhancement known as contrast stretching is presented by Gonzalez (1977). Let r denote any gray level in the original image and s denote the corresponding transformed gray level in the enhanced image given by the transformation $s=T(r)$. If $T(r)$ has the form of figure 6a, the resultant image tends to have a higher contrast. Levels below m are compressed and darkened and levels above m are expanded and lightened. In the limiting case, see figure 6b, a binary image results.

An indirect method of contrast enhancement is through the gray level histogram. The gray level histogram is a function showing, for each gray level, the number of pixels in the image that have that gray level (Castleman 1979). Normalizing the histogram by

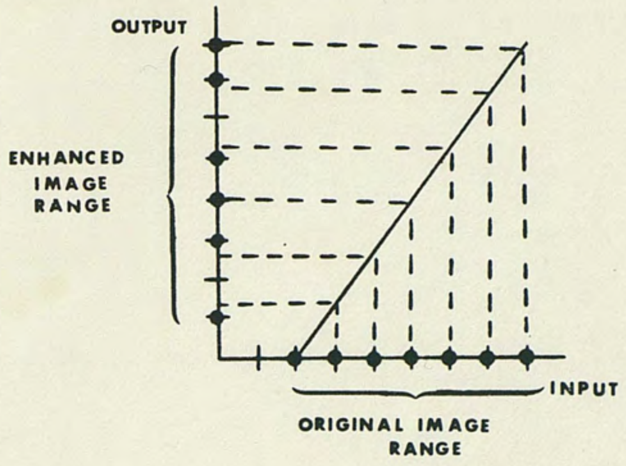
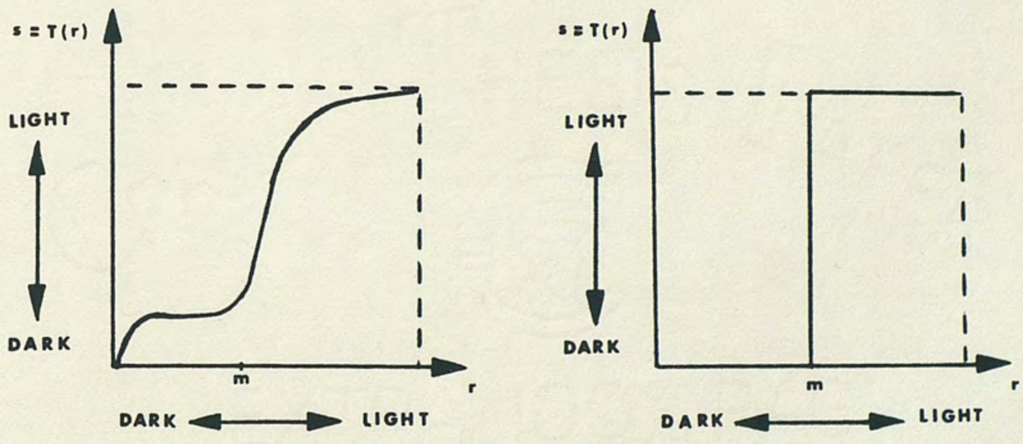


Fig. 5. Discrete image contrast enhancement (Pratt 1978; redrawn by author)



(a) (b)

Fig. 6. Gray level transformation functions for contrast enhancement (Gonzalez 1977; redrawn by author)

dividing by the area of the image yields the probability density function (pdf) of the image. The histogram of an image is unique but the converse is not true since the histogram contains no spatial information on the pixels. A sample histogram is shown in figure 7.

In contrast enhancement through histogram modification techniques, the original image is rescaled so that the histogram of the resultant image follows some predetermined form. Enhanced images possessing exponential, hyperbolic, Rayleigh and uniform shaped histograms have been studied (Pratt 1978). The most common of these is the uniform shape resulting from histogram equalization. The equalization process can be considered to be a point transformation $s=T(r)$ such that the output is uniform over some range. An illustration of this principle is shown in figure 8.

Images with accented edges are often more visually pleasing than the original image. Sharpening or edge crispening is used to accentuate edges. Techniques for edge enhancement include passing the image through a high-frequency bandpass filter, unsharp masking (Pratt 1978), and direct convolutional filtering. The bandpass filter is normally implemented using Fourier transforms. The unsharp masking is a method which requires an image to be scanned at normal resolution and then at a lower resolution. The two images are combined in a way that gives sharper edges with slight overshoot and undershoot. Discrete convolutional filtering employs the use of a convolutional array or convolutional mask. In general, an output MxM image array Q is formed by discrete convolution of the input

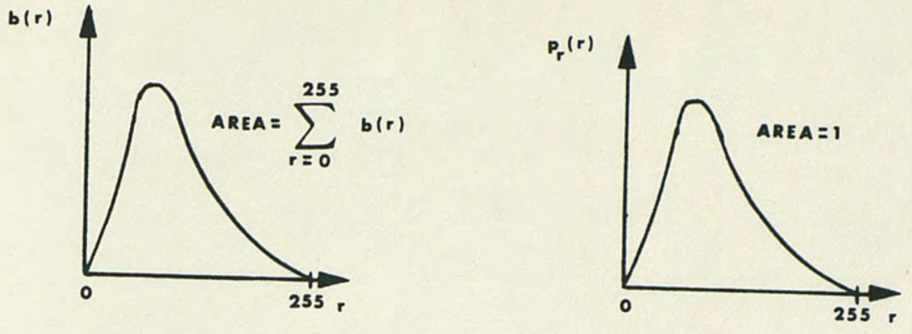
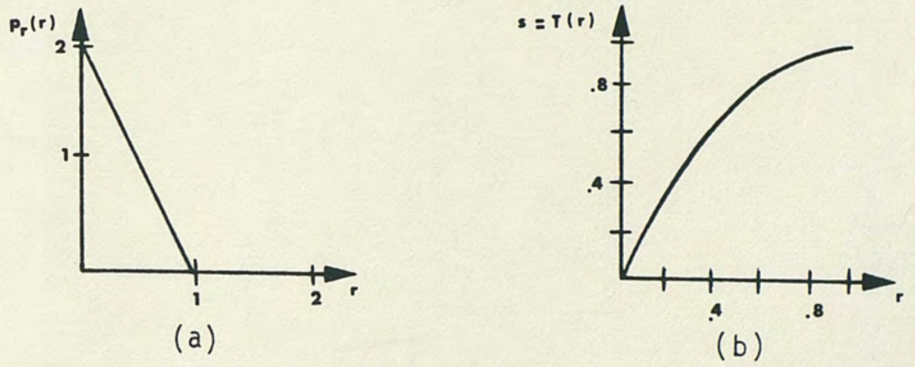


Fig. 7. A histogram and its probability density function representation



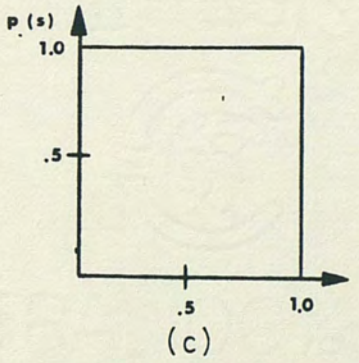


Fig. 8. Illustration of the uniform density transformation method. (a) Original probability density function (b) Transformation function (c) Resulting uniform density

(Gonzalez 1977; redrawn by author)

NxN image array F with the LxL convolutional array H according to the relation

$$Q(m_1,m_2) = \sum_{n_1=1}^{N-2} \sum_{n_2=1}^{N-2} F(n_1,n_2)\ H(m_1-n_1+1,m_2-n_2+1).$$

Some typical high-pass masks are shown in figure 9.

Image smoothing is the term used to refer to an operation used to reduce the noise in digital images. The approach can be either classical in nature or via "spatial ad hoc processing techniques" (Pratt 1978). Classical techniques normally imply a two-dimensional lowpass filter. A transfer function is derived to meet frequency specifications and implemented through either Fourier transforms or difference equations. The Circular Pipeline Video Processor performs most efficiently using spatial filtering techniques due to its specialized architecture. The rest of this section presents several spatial smoothing techniques.

Neighborhood averaging is the simplest case of smoothing. Given an NxN image, a smoothed image is generated by replacing the center pixel of each neighborhood with the average gray level value of that neighborhood. For a 3x3 neighborhood centered at co-ordinates (x,y) with value f(x,y), the new gray level becomes

$$g(x,y) = \frac{1}{8}\left[f(x-1,y-1)+f(x,y-1)+f(x+1,y-1)+f(x-1,y)\right.$$
$$\left.+f(x+1,y)+f(x-1,y+1)+f(x,y+1)+f(x+1,y+1)\right].$$

Variations on this scheme include different neighborhoods, including the pixel itself in the average and requiring that the neighborhood

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Fig. 9. Several high pass masks

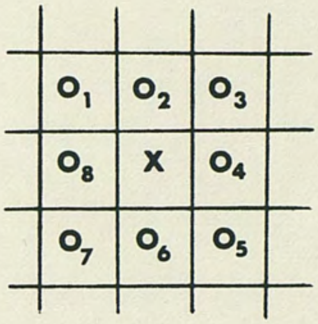


Fig. 10. A neighborhood representation after discrete convolution with a 3x3 mask

(Pratt 1978; redrawn by author)

$$\frac{1}{8}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig. 11. Several lowpass masks

average and pixel value differ by some minimum threshold before replacing the pixel.

Convolutional masks may also be used for lowpass spatial filtering. Following discrete convolution with a 3x3 mask, a neighborhood of the resulting image is shown in figure 10, where O_i ,i=1,2...8, represents the outputs of the convolution and X is the pixel under test. The operation is now similar to that of averaging a neighborhood. Several lowpass form convolutional masks are shown in figure 11.

Averaging of multiple images is another smoothing technique given it is possible to obtain multiple images of the same scene. If a noisy image is denoted as

$$g(x,y) = f(x,y) = f(x,y) + n(x,y)$$

where f(x,y) is the original image and n(x,y) is a noise component which is assumed to be uncorrelated with zero mean, then an image g(x,y) is formed by averaging M different noisy images

$$g(x,y) = \frac{1}{M}\sum_{i=1}^{M} g_i(x,y).$$

Papoulis (1965), Castleman (1979), and Gonzales (1977) all show that as the number of images increases the resultant image g(x,y) more closely resembles the original image f(x,y). Expressing this mathematically

$$E\left\{\bar{g}(x,y)\right\} \rightarrow f(x,y) \quad \text{and} \quad \sigma^2_{\bar{g}(x,y)} \rightarrow \frac{1}{M}\,\sigma^2_{n(x,y)}$$

as $M \rightarrow \infty$.

As M increases, the variability of each pixel decreases and the signal to noise ratio increases by a factor equal to the square root of the number of images averaged. Practical constraints involve the ability to maintain precise image alignment and the inaccuracy evolving from recursive averaging.

The median filter is a non-linear signal processing technique that is especially useful for noise suppression in images. In the one dimensional case, the median filter consists of a sliding window covering an odd number of pixels. The center pixel is replaced with the median value of the window. The median value is defined as that value for which (N-1)/2 pixels are smaller or equal and (N-1)/2 pixels are larger or equal. The median filter is an edge preserving filter. Steps and ramps are left undisturbed while noise is suppressed. The median filter is extended to two dimensions by extending the window to cover a neighborhood of pixels. The center pixel is then replaced by the median value. As pointed out by Pratt (1978), the median filter is much more effective in reducing high frequency noise than smoothly generated noise. He cautions that the median filter is an ad hoc tool whose performance should be monitored to determine its usefulness.

Segmentation

Given an image, image analysis must be performed to extract a meaningful description of the image contents for further processing. A meaningful description, of course, is a function of the information required from the image. Segmentation is the term applied to the procedure for dividing the image contents into separate regions. Numerous segmentation methods are available. The choice is dependent on factors such as accuracy required, speed and hardware restrictions. The segmentation evaluation criteria is clearly a function of the usefulness of the segmented image.

Segmentation classifies individual pixels into classes or states. Classification methods may depend on an individual pixel's gray level, local properties or global characteristics. The simplest segmentation scheme uses only an individual pixel's brightness for classification. Given a fixed brightness level, all pixels brighter than this level are grouped in one class and all the rest are grouped in another class. This is the same concept presented in the limiting histogram equalization example of figure 6. The method can easily be expanded by adding information from the entire scene or from local neighborhoods to dynamically adjust the threshold gray level.

For images composed of relatively few kinds of regions, the gray level histogram is likely to contain concentrated areas of pixel populations. A simple segmentation approach for this type of image is to select gray level thresholds which separate these populated regions into separate groups and classify each pixel according to the group to which it belongs. This is termed amplitude segmentation.

Extending amplitude segmentation to include images with spectral information leads to a spectral classification technique. Spectral classification uses a clustering or grouping technique to identify similar regions by their color. The color of the pixel defines a point in (red, green and blue) space and pixels in close proximity are grouped together.

Local properties provide numerous features to use as segmentation criteria. Edges, lines and spots can be detected by local operators and used to segment images. The busyness of a pixel based on its neighborhood can help separate regions of different texture. The average neighborhood gray level is a basic local property.

Spatial classification utilizes several local properties to provide a more powerful approach. For example, a property or characteristic of a pixel or its neighborhood such as gray level is compared against the average gray level of a set of neighborhoods to determine the degree of similarity. A pixel is thus classified based on a spatially related set of pixels and neighborhoods.

Combinations of the above methods are often used. Local properties can be used to assign an edge value to a pixel based on the strength of the edge. Amplitude segmentation is then used to identify only those pixels with an edge value greater than a minimum threshold as true edges.

The CPVP classifies pixels into states using amplitude segmentation. The threshold(s) is dynamically adjusted using scene information contained in the global histogram of the image. Although the implementation of such a scheme is relatively straightforward, the scheme is not ideal. As will be demonstrated later, a greater emphasis must be placed on prior knowledge of scene content and image enhancement for the approach to be successful.

CHAPTER 4

THE CIRCULAR PIPELINE VIDEO PROCESSOR

The System Hardware

The image processing required by the visual inspection process is performed by the Circular Pipeline Video Processor (CPVP). The CPVP is shown in figure 12 and a block diagram highlighting its internal functions as well as certain external support functions is shown in figure 13. It is not the purpose of this paper to provide a detailed description of the CPVP hardware and software. However, to appreciate the data analysis performed, analysis tools required and the direction taken in investigating alternate segmenting routines, a basic understanding of the CPVP and its support equipment is necessary.

Using figure 13 as a guide, the following list briefly describes the CPVP system.

1. Host Computer - The host computer is a Hewlett Packard HP9836 and is responsible for executive control of the entire visual inspection station. Control is exercised over the HP Interface Bus.
2. Mass Storage - Mass storage is supplied by an HP7908 hard disc and is responsible for storing all programs, algorithms, and data bases required by the inspection process.

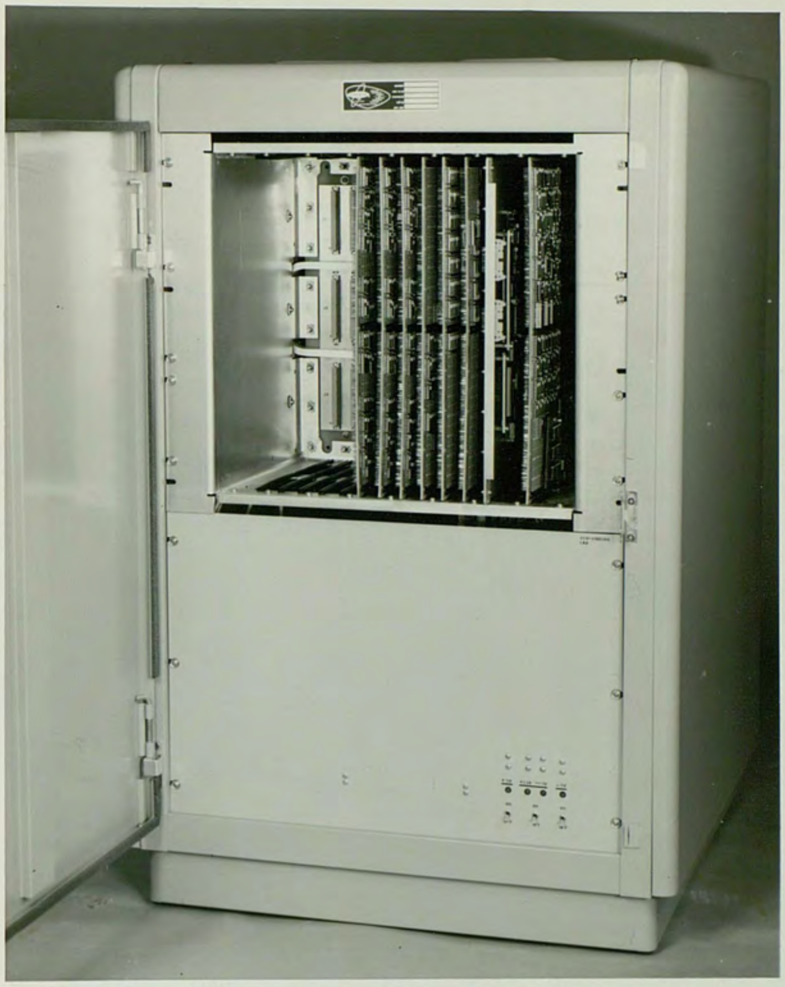


Fig. 12. The Circular Pipeline Video Processor

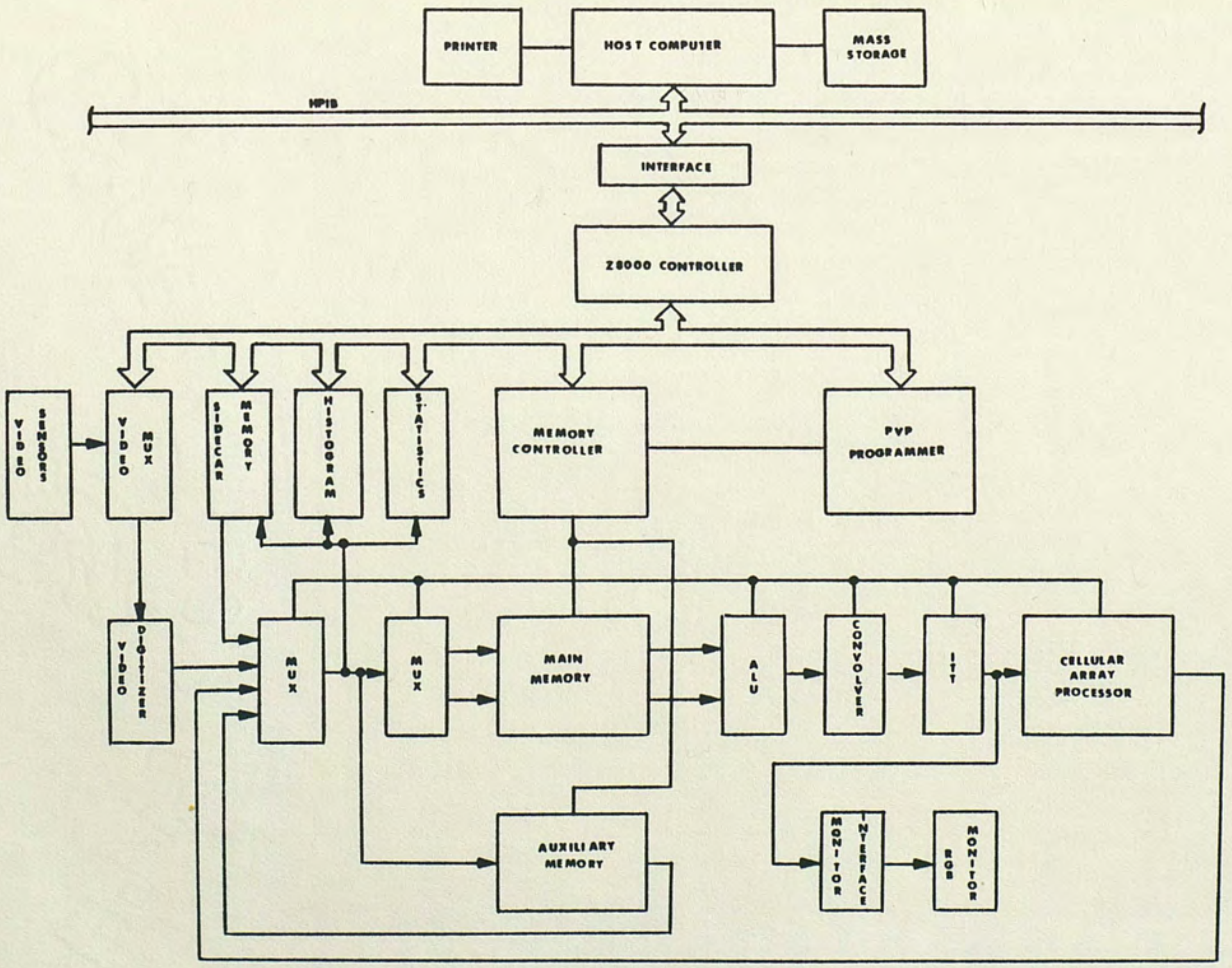


Fig. 13. Functional diagram of the CPVP

3. Printer - The HP2671 printer provides a hard copy listing of flaw type and flaw location for each PCB inspected.
4. Z8000 Controller - The CPVP controller is an Advanced Micro AM96/4116A MonoBoard Computer. It is a Z8002 based single board computer including both serial and parallel ports and a vectored interrupt structure. It is responsible for executing all algorithms and commands within the CPVP itself.
5. Interface - The interface provides an HPIB port for the Z8000 controller.
6. Video Sensors - The video sensors are four Fairchild CCD3100 solid-state black and white television cameras. The sensor device is a 488x380 element buried-channel charge-coupled-device.
7. Video Mux - The video mux selects one of four video lines for input to the CPVP while distributing sync signals to all four cameras.
8. Video Digitizer - The video digitizer is based on the TRW1007 A/D converter and converts the raster scan video to 8 bit standard TTL levels.
9. Sidecar Memory - Sidecar memory is a utility memory capable of holding one line of video information. It provides an interface between the frame storage memories and the Z8000 controller.

10. Histogram - The histogram hardware computes and stores the global histogram of an image as it enters the main memory.

11. Memory Controller - The memory controller controls all high speed data transfers into or out of any of the memories.

12. Main Memory - The main memory is capable of storing two 8 bit images and serves as the working memory for most processes.

13. Auxiliary Memory - Auxiliary memory is used for off line storage and can hold four complete images.

14. PVP Programmer - The pipeline video processor programmer controls real time data acquisition and high speed pipe operations.

15. ALU - The arithmetic logic unit permits algebraic manipulation of an image or between two images.

16. Convolver - The convolver allows discrete convolution of the image with a 3x3 mask.

17. ITT - The image transform table performs a mapping function. Pixel values may be mapped to new values or it may be used to create a pseudo-color output.

18. Monitor Interface - The monitor interface provides sync signals and D/A conversion for displaying the contents of main memory.

19. Monitor - The monitor is an RGB color monitor capable of accepting external sync signals.

20. Cellular Array Processor - The cellular array processor (CAP) performs operations on a pixel based on its own state and/or the state of a set of its neighbors.

With the basic functional blocks of the CPVP explained, the overall operation of the system can be illustrated. The term circular pipeline video processor is indicative of the image processing hardware architecture. As seen in figure 14, the data can be envisioned as leaving main memory, traveling through a pipe of processing operations and returning to main memory. This type of architecture permits multiple operations to occur with each pass through the pipe, thus reducing processing time.

The System Software

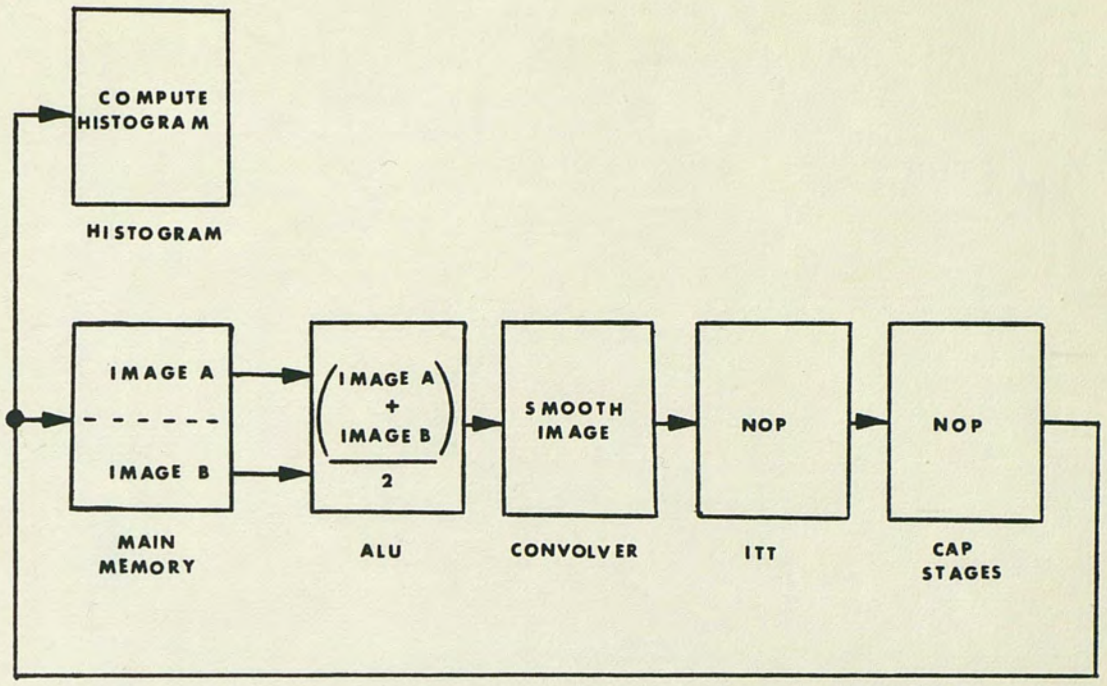
Software associated with the CPVP is concentrated in three main areas; HP Basic, CAPS Macros, and Z8000 assembly language. HP Basic is used by the HP9836 host computer to control the visual inspection station. Commands are issued and responses interpreted using Basic programs.

CAPS Macros are the algorithms developed by the Martin Marietta Image Processing Lab for flaw detection on PCB's. CAPS is discussed in greater detail in the data analysis section.

Z8000 Assembly language is used for internal CPVP operations. Most debug and test routines are written in assembly code. CAPS Macros are also reduced to assembly code for implementation.

An Operational Illustration

In order to tie the hardware and software together, a sample processing sequence is illustrated in figure 14 and described below:



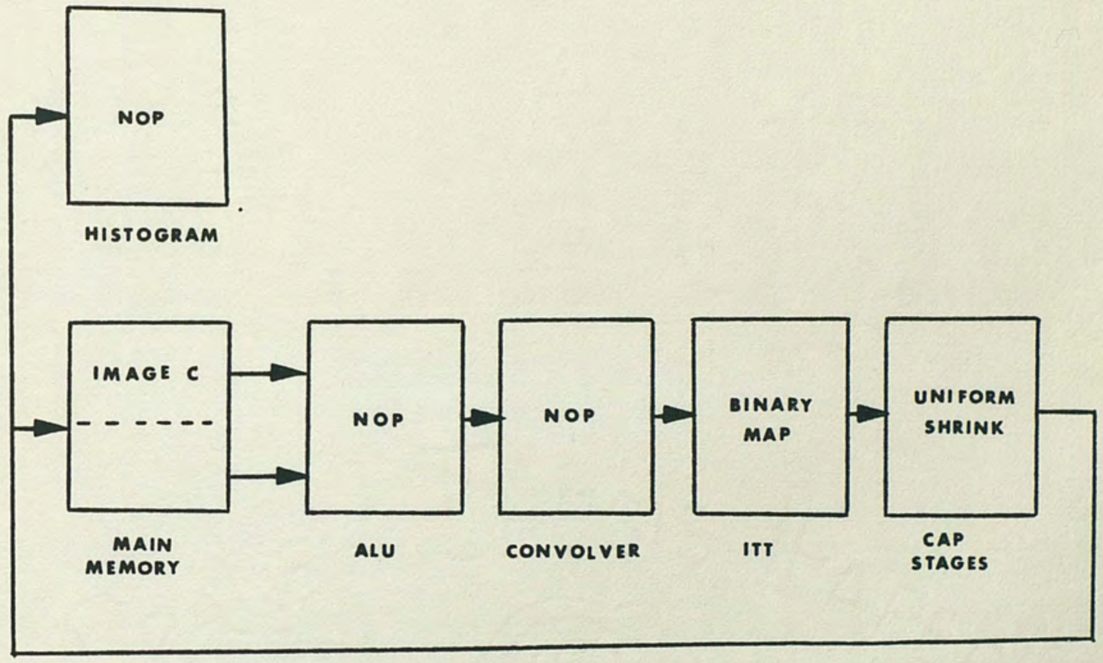


Fig. 14. A possible CPVP operational sequence

Initialization:

1. An image of a PCB is captured and stored in one half of main memory.
2. The same scene is again captured and stored in the other half of main memory.

First Pass:

1. These two images (image A and image B) are averaged together pixel by pixel in the ALU.
2. The resulting image is smoothed via neighborhood averaging in the convolver.
3. The ITT and CAP stages perform no operation.
4. A global histogram is computed as the image returns to memory.

Second Pass:

1. The image passes through the ALU and convolver unaffected.
2. Based on the global histogram, a threshold value is determined and the ITT maps the gray levels into two states corresponding to land and background.
3. The CAP stages perform a non-destructive (no land is obliterated) uniform shrink on all lands. Assuming a ten pixel minimum land width, any land reduced to a two pixel width in less than four shrinks is too thin and tagged accordingly. Note that one shrink removes one pixel from each side of the land.

4. Sections of the image which have been tagged as containing flaws are reported to the host computer. Flaw descriptions are printed and the process is ready to repeat with a new image.

CHAPTER 5

IMAGE DATA ANALYSIS

Purpose of Investigation

The principles and feasibility of the automated visual inspection station, including the CPVP, have been proven in numerous demonstrations. However, there has been concern over the accuracy of the current segmentation algorithms. All flaw detection schemes in the CPVP operate on images segmented into two or three states as compared to the unsegmented 256 states. Obviously, the ability of the CPVP to correctly identify PCB features to a high degree of accuracy is of paramount importance if the flaw detection algorithms are to yield reasonable results.

In the early development of smoothing, segmentation, and flaw detection algorithms, simulated image data was used. Since that time the CPVP hardware has become operational and the final camera configuration and lighting conditions established. Printed circuit boards from the Ocala manufacturing facility have been obtained and used as demonstration boards. The flaw detection success rate has not been 100 percent with these boards. In addition to other reasons, inconsistent and inaccurate segmentation is viewed as a possible cause for the degraded operation. The rest of this paper contains an analysis of the CPVP using actual image data and investigates alternate segmentation techniques. The purpose of this analysis and investigation is to provide a basis for improving the accuracy and reliability of the visual inspection process.

Investigative Tools

The first requirement for analyzing image data was to develop the capability to collect the image data and transport it to an image processing facility. Subsequent analysis could then take advantage of current on-site systems. Investigative tools were developed in HP Basic, Z8000 assembly code, Fortran and CAPS. DEC control language routines were also invoked. The following list details the software required for the image transfer:

1. Assembly code to present quarter-line segments of the CPVP main memory to an HPIB accessible I/O buffer.
2. Assembly code to transfer Z8002 RAM contents to the HPIB interface.
3. HP Basic to read Z8002 ROM (or ZSCAN emulator RAM) as ASCII data.
4. HP Basic to interrogate the CPVP for image data and store this data on an IPL formatted floppy disc.
5. DCL to transfer an image from an IPL formatted disc to an image plane in the DeAnza Array Processor.

This software development was a significant task and involved the efforts of a group of engineers working separately and collectively. The image transfer path is shown in figure 15.

The purpose of the image transfer to the CAPS system was to use its capabilities to evaluate image data. The CAPS system includes the CAPS language, a minicomputer to decode CAPS instructions and an array processor to execute the instructions. Features

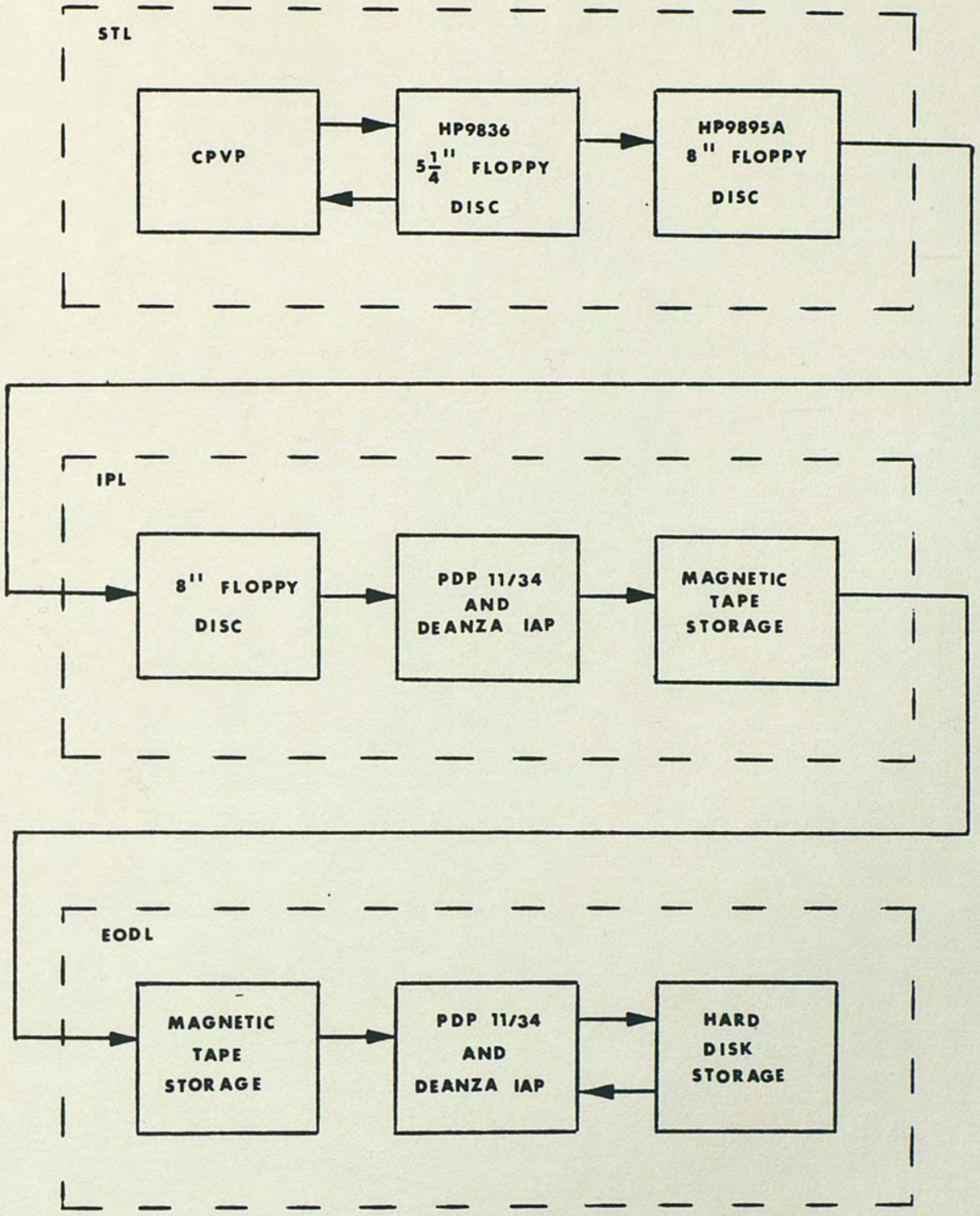


Fig. 15. Image transfer path

of the system include:

1. PDP 11/34 software which sets the various registers, transformation tables, etc. of the DeAnza array processor to operate on two 512x512 pixel images in 1/30 second.
2. Four unique 512x512 pixel images accessible for image array processing.
3. Fast image plane to image plane transfer (1/30 second).
4. Fast image transfer between the DeAnza array processor and either magnetic tape or disc via PDP 11/34 buffering.
5. Image scroll and zoom capability.
6. Display cursor control in both movement and size manually available at joystick console.

Image Data Description

The CPVP image data collected for analysis was obtained during the final days of the visual inspection detail station integration. This section describes the sample images and the conditions under which they were made.

The procedure for obtaining an image used the following steps:

1. A sample PCB was manually loaded onto the X-Y table.
2. The restraining system (clamps and vacuum) were activated.
3. The X-Y table was manually positioned.
4. A frame capture command was issued to the CPVP.
5. The captured image was transferred to the CAPS system as outlined previously.

System lighting included a combination of circular and linear fluorescent lights. The luminance was not measured, but uncontrolled external sources such as room lighting and light from a large window provided minimal illumination as compared to the direct lighting.

Camera system #2 was used for all image captures. Associated camera settings are provided by figure 16.

Twenty sample images were transferred from the CPVP to the CAPS system and are listed in Table 1. A few floppy discs contained several bad sectors of 128 bytes each resulting in invalid image data. For the images transferred on discs containing bad sectors, a linear interpolation scheme was used to replace invalid pixels with the average value of their upper and lower neighbors. Official background material for artwork had not been established. Two sheets of HP thermal paper provided a suitable background while providing a basis for future repeatability. Images 16 through 20 were captured in immediate succession without disabling the restraining system or repositioning the X-Y table. However, an additional terminating load was apparently placed on the camera during the capture of image 19. The result was a uniform decrease in intensity below the level of images 16 through 18 and 20. These repetitive captures provided a basis for implementing multiple image and histogram averaging techniques. This report deals predominately with inner layers, though the concepts and results can be extended to cover outer layers and artwork.

System Response

In order to study segmentation using the CPVP, it was necessary to know how the system "sees" the PCB under test. In effect, this is the system response. The system in this sense includes the

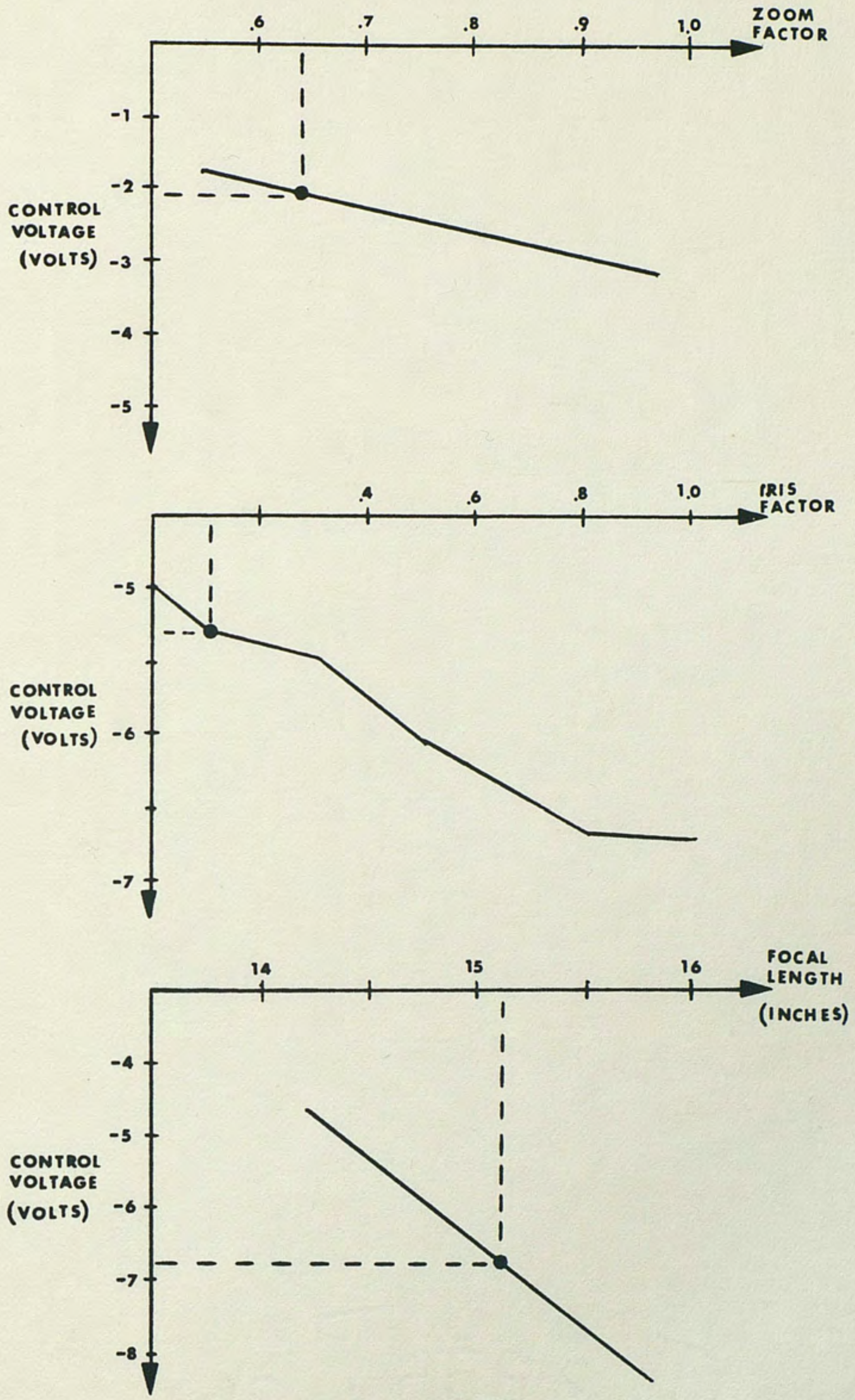


Fig. 16. Camera settings for image captures

TABLE 1

IMAGE DATA BASE

Image Number	Description	Comments
1	Inner layer	
2	Inner layer	
3	Inner layer	
4	Inner layer	
5	Inner layer	
6	Inner layer	Three interpolated 1/4 line segments
7	Outer layer	Reflowed solder
8	Outer later	Not reflowed
9	Outer layer	Not reflowed
10	Outer layer	Not reflowed
11	Artwork	
12	Artwork	Three interpolated 1/4 line segments
13	Artwork	
14	Artwork	
15	Resolution chart	
16	Inner layer	16 through 20 are different captures of the same image.
17	Inner layer	
18	Inner layer	
19	Inner layer	
20	Inner layer	
21	Inner layer	Multiple image average of 16 and 17.
22	Inner layer	Multiple image average of 16, 17, 18.
23	Inner layer	Multiple image average of 16, 17, 18 and 20.

complete image path from the surface of the PCB, through the optics, the camera, the video preprocessor, the digitizer and into the CPVP memory. Although each section of this pathway has its own response, affected by shadows, CCD blooming and antialiasing filters, the primary concern was to determine total system effects and to draw some correlation between the actual PCB and the PCB as seen by the CPVP.

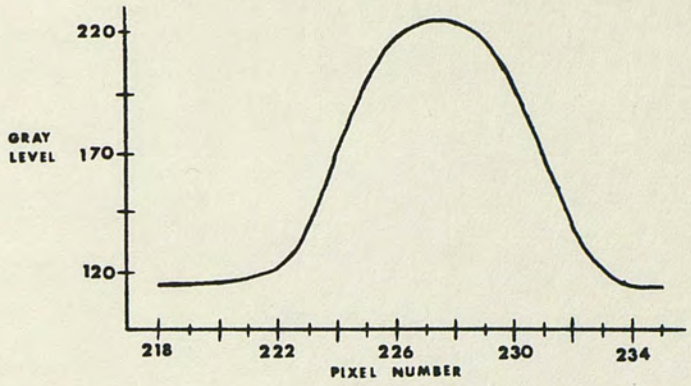
The approach chosen to model the system response was to model the response in the X and Y directions independently as two first order responses. Although an image normally has a high degree of correlation between the horizontal and vertical directions, there are valid reasons for the independently modeled approach.

The CCD sensor is inherently a two-dimensional system since all cells are illuminated simultaneously. The output video signal, however, is in a raster scan format resulting in a one-dimensional time varying signal. This signal is subsequently subjected to filtering and processing in the one dimensional domain before being stored in memory. Also, PCB fault finding algorithms perform many operations in the X and Y directions independently, especially dimensional checks. Since accurate segmentation is critical for dimensional checks, the effects of segmentation based on system response in the X and Y directions is important.

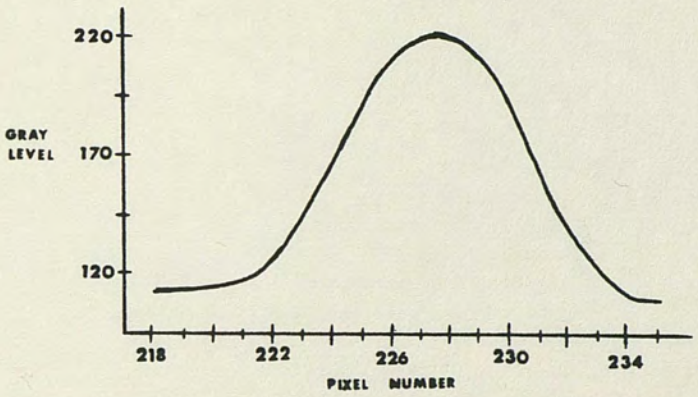
Edge plots in the X and Y direction were constructed from sample edge regions of images 16, 17, 18 and 20. Using the CAPS system, pixel values from corresponding regions of each image were obtained. The average pixel value for a given co-ordinate was then computed and plotted. Edge plots based on the original stored image data are shown in figures 17a and 18a. Treating the transition from background to land and vice versa as step inputs to the system, the rise and fall times and associated bandwidths based on first-order models representing the system response in the X and Y direction are listed in Table 2.

An immediate result of the edge plots and models is the ability to determine the pixel resolvability. Based on the rise and fall times of the edge plots (or the frequency response of the first-order model) there is an amount of pixel classification uncertainty. Assuming that a pixel must be less than or equal to the average background level to be classified as background and greater than or equal to the average land level to be classified as a land, then there are regions of classification uncertainty. The number of uncertain pixels corresponding to this classification strategy for figures 17 and 18 are listed in Table 3.

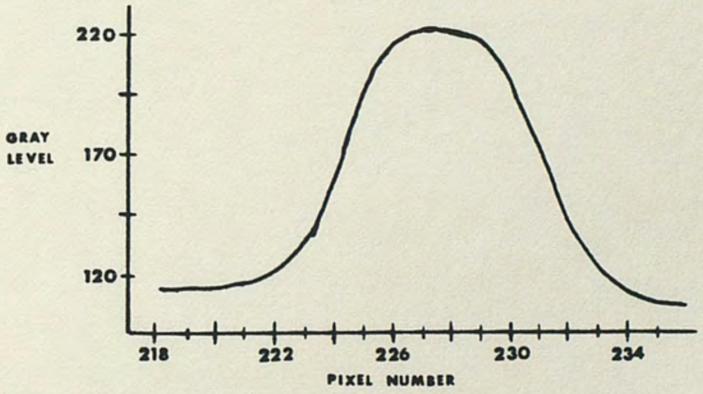
The uncertainty problem is compounded by smoothing techniques employed by the CPVP. Smoothing is necessary to reduce noise and sensor anomalies. But, the most easily implemented approach of neighborhood smoothing is also very costly in terms of system response and resolvability. To illustrate this, images 16, 17, 18 and 20 were subjected to the left-most lowpass convolutional mask of figure 11. The same edge regions were then examined as for the original images and the edge plots constructed. These edge plots and the corresponding system parameters are shown in figures 17b and 18b and Table 2. The number of uncertain pixels has increased as shown in Table 3.



(a)

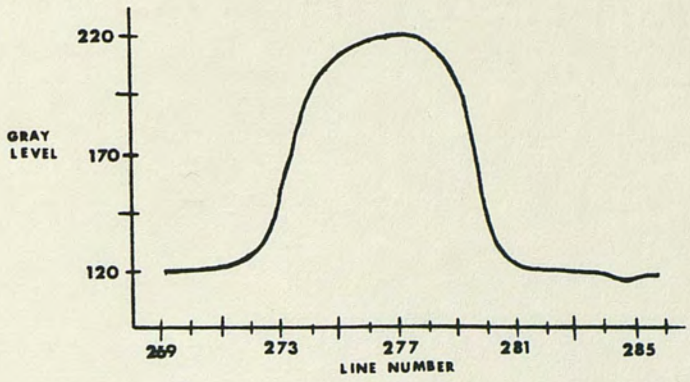


(b)

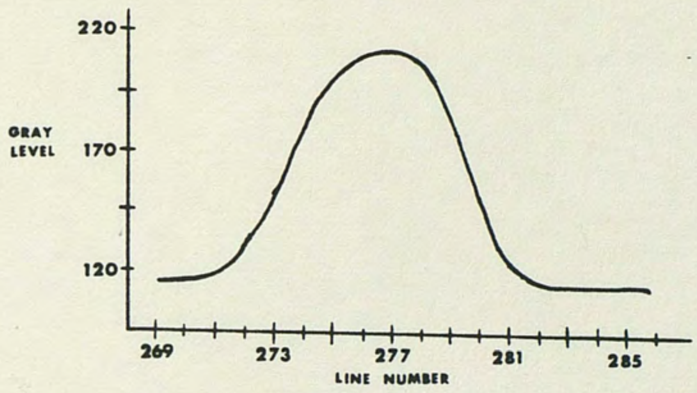


(c)

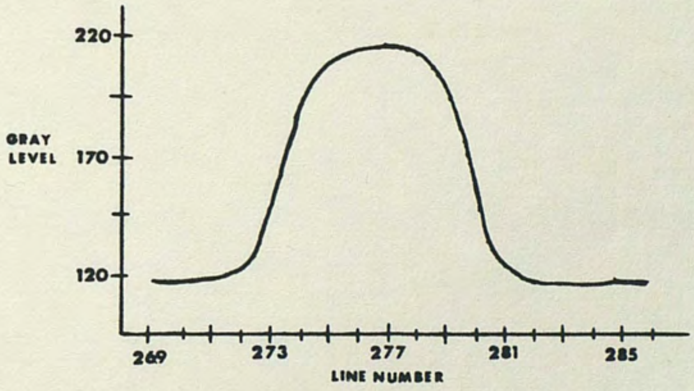
Fig. 17. X-edge profiles. (a) Original image (b) Average smoothed image (c) Median smoothed image



(a)



(b)



(c)

Fig. 18. Y-edge profiles. (a) Original image (b) Average smooth image (c) Median smoothed image

TABLE 2

EDGE RESPONSE

		X Edge	Y Edge
Original Image	Rise Time	575 ns	445 ns
	Bandwidth	609 KHz	787 KHz
	Fall Time	575 ns	334 ns
	Bandwidth	609 KHz	1050 KHz
Average Smoothed Image	Rise Time	741 ns	649 ns
	Bandwidth	472 KHz	540 KHz
	Fall Time	695 ns	463 ns
	Bandwidth	504 KHz	755 KHz
Median Smoothed Image	Rise Time	574 ns	463 ns
	Bandwidth	609 KHz	755 KHz
	Fall Time	574 ns	426 ns
	Bandwidth	609 KHz	821 KHz

TABLE 3

PIXEL UNCERTAINTY

		Number of Uncertain Pixels	
		X Edge	Y Edge
Original Image	Rising Edge	3	3
	Falling Edge	3	2
Average Smoothed Image	Rising Edge	4	4
	Falling Edge	4	3
Median Smoothed Image	Rising Edge	3	3
	Falling Edge	3	3

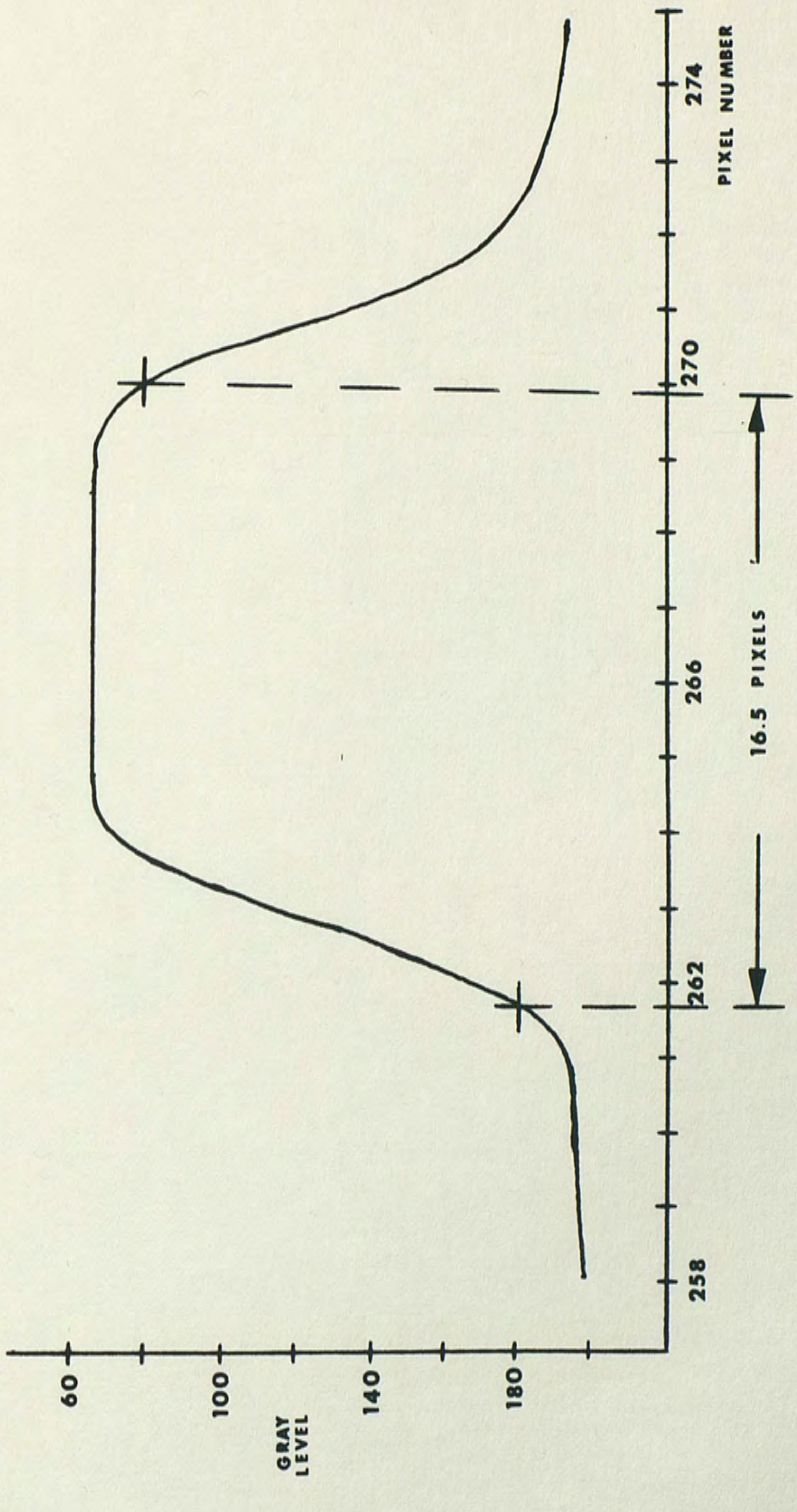


Fig. 19. Resolution chart edge profile

Although not readily implemented via the CPVP hardware, the effects of a median filter were also studied and illustrated in the same manner as the previous two cases. Note the edge preserving quality of the median filter.

Additionally, a USAF #1951 target was used as a resolution chart. An unsmoothed X-edge profile was taken using this chart and is shown in figure 19. The edge profile characterizes a 0.0005 millimeter wide bar on a clear film. Background material was the normal white paper. The bar width of 16.5 pixels was determined by using thresholds equal to 10% and 90% of the difference between the average bar (object) gray level and the average background gray level for the leading and trailing edges respectively. The 16.5 pixel width corresponds to 0.303 millimeters per pixel or 0.0012 inches per pixel. This spatial relationship between pixel count and object size, along with the pixel uncertainty discussed earlier, can be used to aid in choosing an acceptable threshold at which to segment the image.

An obvious selection scheme is to choose a threshold such that the segmented land has the same dimensions as the true land. This implies that a single threshold cannot be placed at either the average object or the average background level. These schemes would place the threshold in a noisy region, particularly near the background level. Also, experience has shown that PCB backgrounds posess a degree of non-uniformity. Background levels vary with respect to their location on the PCB.

The threshold must be selected to segment the image in the uncertain area to avoid serious noise problems. Furthermore, the level chosen is a function of the system response, the smoothing technique employed and the pixel values present. In order to properly segment the image to the degree of accuracy desired, a consistent segmentation method must be used and the true dimensions determined using prior established knowledge of the system response. The next chapter looks at several approaches for choosing a threshold for subsequent segmentation.

CHAPTER 6

ALTERNATE SEGMENTATION TECHNIQUES FOR THE CPVP

Background Research of Threshold Techniques

This section describes various threshold selection techniques as presented in recent publications and used in industry with varying degrees of success. Most of the techniques presented will be limited to dealing with images comprised of two types of regions, background and object, or background and land in the PCB case.

For the general case, a threshold operator can be viewed as a test involving a function T of the form

$$T(x,y,N(x,y),\ g(x,y))$$

where $g(x,g)$ is the gray level of the pixel at co-ordinate (x,y), $N(x,y)$ involves a local property at (x,y) and x and y are position dependent variables. Each point (x,y) is tested and if $g(x,y) \geq T(x,y,N(x,y),\ g(x,y))$ then (x,y) is classified as an object point. Otherwise, it is a background point (Weszka 1977).

One of the earliest techniques for thresholding was the standard histogram method. This technique called for the threshold to be placed at the minimum between the peaks of the histogram. Drawbacks include long flat valleys making threshold selection difficult and the exclusion of edge information in the decision process. Weszka (1977) points out that Doyle has suggested that a "p-tile" method can enhance the standard histogram method providing the objects occupy a known percentage of the image. This scheme selects a threshold which causes at least q% of gray levels to map into the object, where q is a predetermined percentage. According to Weszka (1977), others have suggested an alternate approach for making the standard method more useful. Basically, they computed a histogram in which all pixels were not weighted equally. Rather, the value of a difference operator at a point influenced that point's weight in the histogram. Higher weight was given to low edge value points, thus sharpening the histogram.

For histograms with broad valleys and unequal size peaks, Weszka (1977) suggested the use of a digital "Laplacian" operator to produce a strongly bimodal histogram. The "Laplacian" is computed by taking the absolute difference between a pixel's gray level and the average gray level of its neighborhood. Using only points which have Laplacian values in the upper percentile to construct the histogram results in a relatively symmetrical histogram with a sparsely populated valley. Analogous techniques to the Laplacian based technique have been developed using gradient operators and histogramming only those points with high gradient values. The valley should correspond to the gray levels at which edge transitions are the strongest.

In an attempt to minimize the misclassification error, a process of Gaussian curve fitting has been used. Rosenfeld and Kak (1982), Weszka (1977) and numerous others have derived the formulation for which the average probability of error is minimized. For a bimodal histogram, the threshold is chosen at the intersection of the two Gaussian curves representing the two peaks with the restriction that the intersection occurs between the two modes.

Thresholding schemes have also been devised for turning valleys into peaks (Weszka 1977), sharpening the peaks (Peleg 1977), and combining peaks. Additional schemes incorporate greater scene information in the decision process using local and geometric properties. Scatter plots are often used to incorporate more variables than the histogram contains. Basically, though, the thresholding process reduces to the task of finding a value T which separates the image into background and object.

Most thresholding schemes developed measure their effectiveness by their ability to segment images into recognizable or at least usable parts. Furthermore, suggested improvements are, more often than not, costly to implement in hardware. The CPVP relies strictly on standard histogram techniques for threshold selection. Hardware and time constraints limit the invocation of local properties into histograms. Therefore, the rest of this paper deals with thresholding techniques restricted to the constraints of the CPVP.

Histogram Enhancement

There are two possible avenues for histogram improvement. The image can be processed to improve the histogram before it is computed or the histogram can be operated on directly. The two techniques will be referred to as image smoothing and histogram smoothing.

As discussed in Chapter 3, image smoothing reduces noise in the image. Another result of image smoothing is a smoother histogram. Images obtained for use in multiple histogram averaging were used to create images smoothed by a median filter and smoothed by a lowpass convolutional mask. Results for one and two passes of the lowpass mask were included. The corresponding histograms are given in figures 20 and 21 based on linear and logarithmic scales respectively. The log scale emphasizes histogram valley region activity.

Direct histogram smoothing techniques have not been discussed yet. The histogram smoothing techniques employed for this paper are the uniformly weighted averaging window and the median filter window. The algorithms used for histogram manipulation were developed on the HP9836 computer for reasons of accessibility. A listing of these smoothing subroutines is contained in appendix A.

The first histogram smoothing operation applied was the window averaging method. Basically, a variable length window of 3,5,7 or 9 bins was moved along the histogram. The histogram value at the center bin of the window was replaced by the average value of the histogram within the window. The resulting smoothed histograms corresponding to image 16 are shown in figure 22.

The second histogram smoothing technique is similar to the averaging window except that the center histogram value was replaced by the median value within the window. This approach attempts to remove artificially low or high value bins while preserving the sharpness of the peaks. In particular, images 1 through 5 each had a bin an order of magnitude lower than its neighbors. This seemed a highly unlikely occurrence and would severely affect threshold selection if not compensated for. The median filter is able to do what the averaging window cannot in a case such as this. Results of the median smoothing can be seen in figure 23.

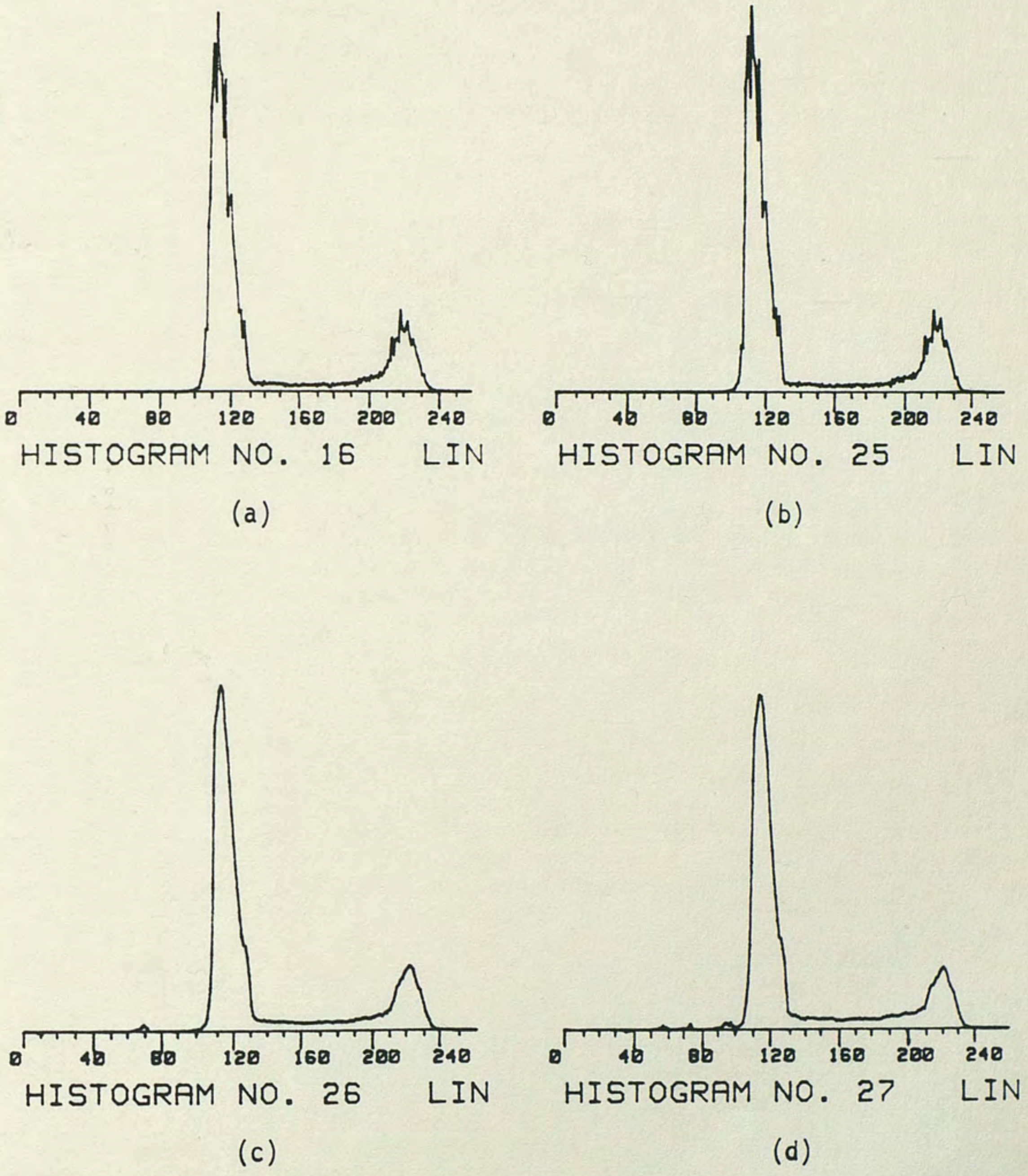
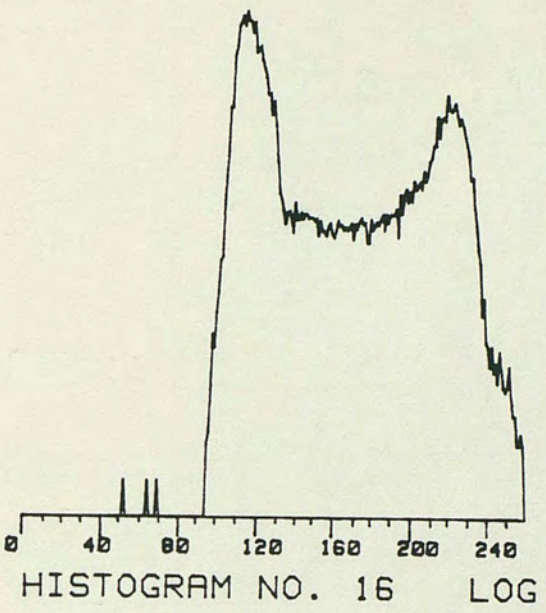
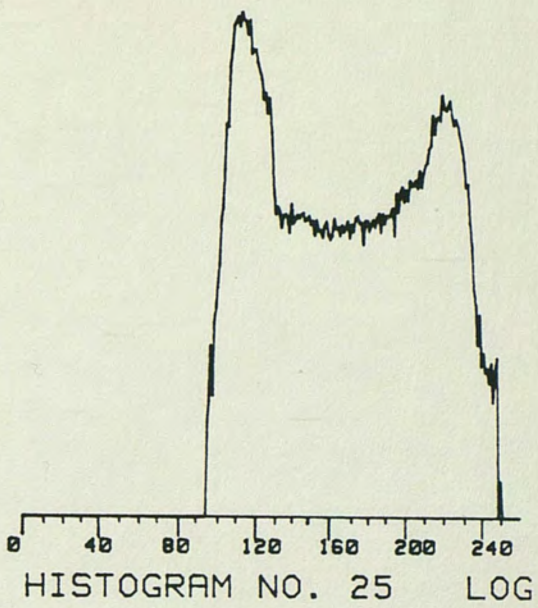


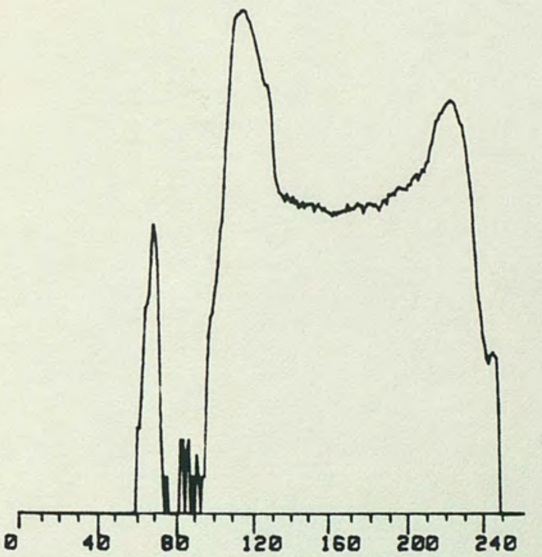
Fig. 20. Linear scaled histograms of smoothed images. (a) Image 16 unsmoothed (b) Image 16 median smoothed (c) Image 16 average smoothed once (d) Image 16 average smoothed twice



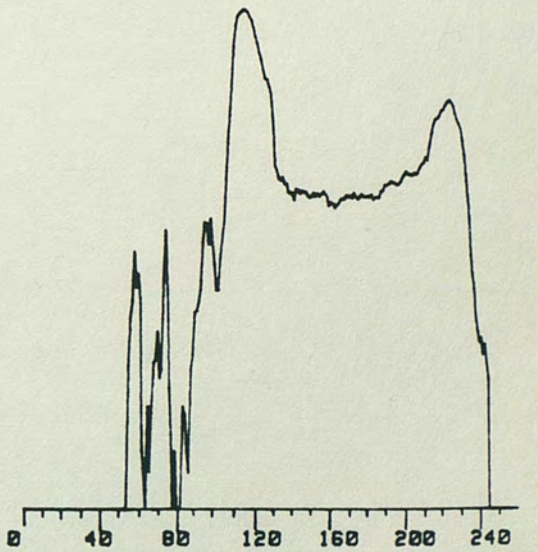
(a)



(b)



(c)



(d)

Fig. 21. Log scaled histograms of smoothed images. (a) Image 16 unsmoothed (b) Image 16 median smoothed (c) Image 16 average smoothed once (d) Image 16 average smoothed twice

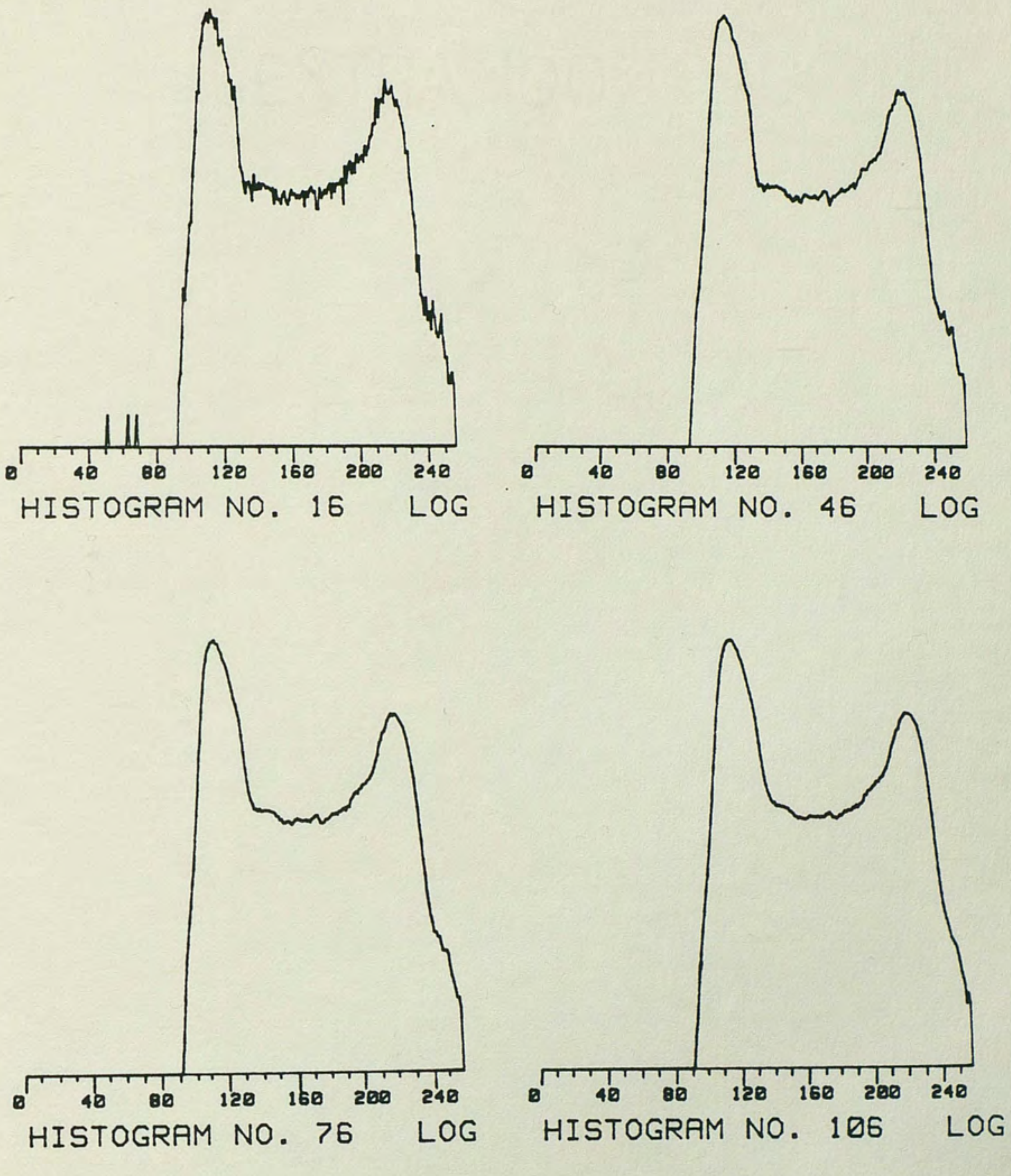


Fig. 22. Average smoothed histograms for window sizes 1, 3, 5 and 7

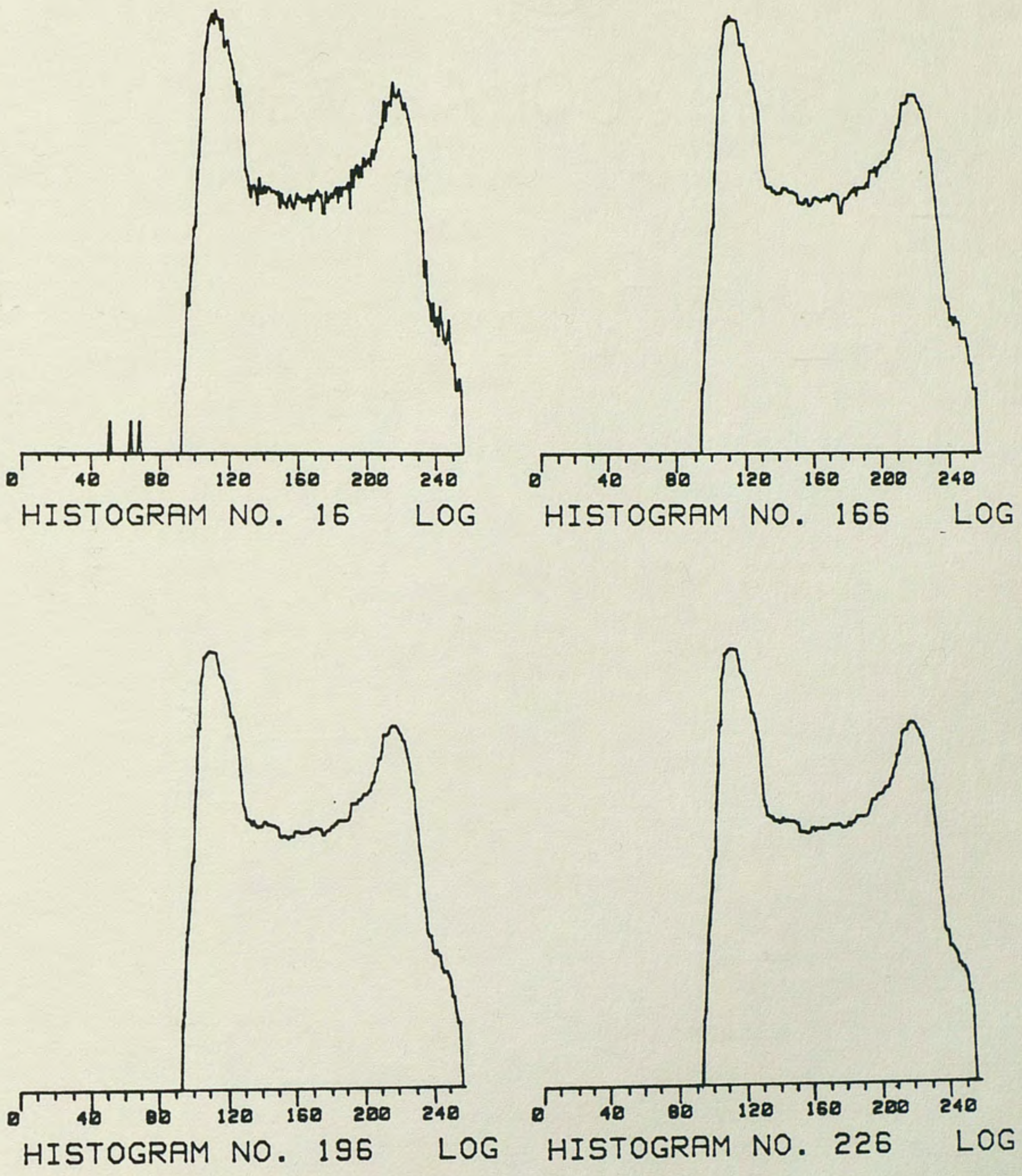


Fig. 23. Median smoothed histograms for window sizes 1, 3, 5 and 7

The third histogram smoothing technique useo the histograms of images 16, 17, 18 and 20, four histograms of the same image. The histograms were averaged together as a group to provide the resultant histogram. Given N histograms with values a_i , i = 0,1,...255, then the resulting histogram values are given by

$$b(i) = \frac{1}{N}\sum_{j=1}^{N} a_{i,j} \quad \text{for } i = 0, 1, \ldots 255$$

where $a_{i,j}$ = the value at the ith bin of the jth histogram

i = the bin

N = number of histograms and

b(i) = the resultant bin value.

Figure 24 shows the result of this method.

Histogram Model

In addition to the basic histogram enhancement techniques, a model for a typical bimodal histogram was developed. The purpose of the model was to use general tendencies of the original histograms to construct histograms which were more easily and consistently assigned threshold values. The histogram model consists of two Gaussian based functions representing the two peaks and a polynomial function for the valley region. The subroutine used to create the model is given in Appendix A. Since many of the threshold selections to follow depend on parameters directly or indirectly derived from the modeling subroutine, the procedure used is briefly explained below. Wherever possible, variable names are kept consistent between the description and the actual program. Figure 25 serves as a guide.

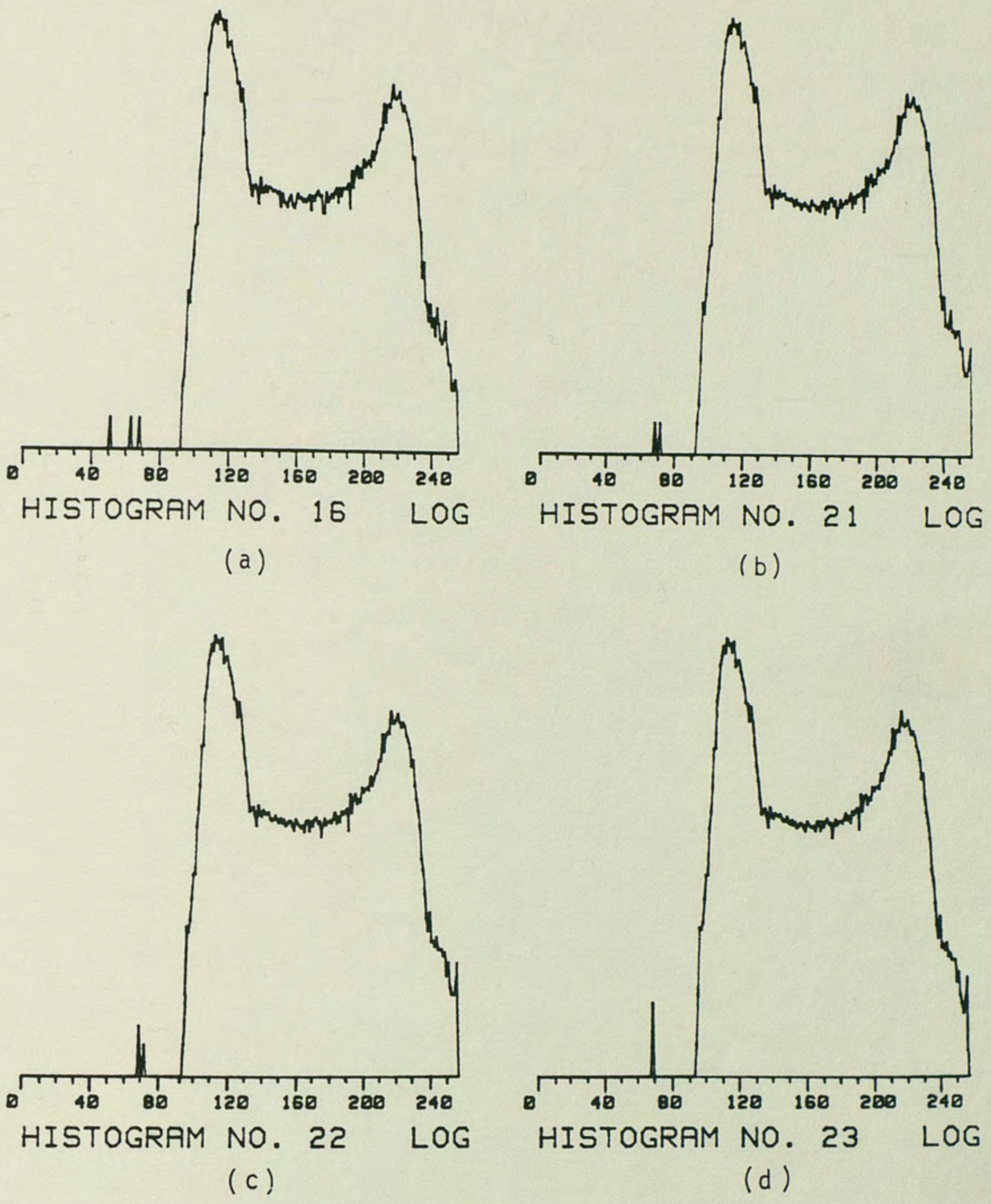
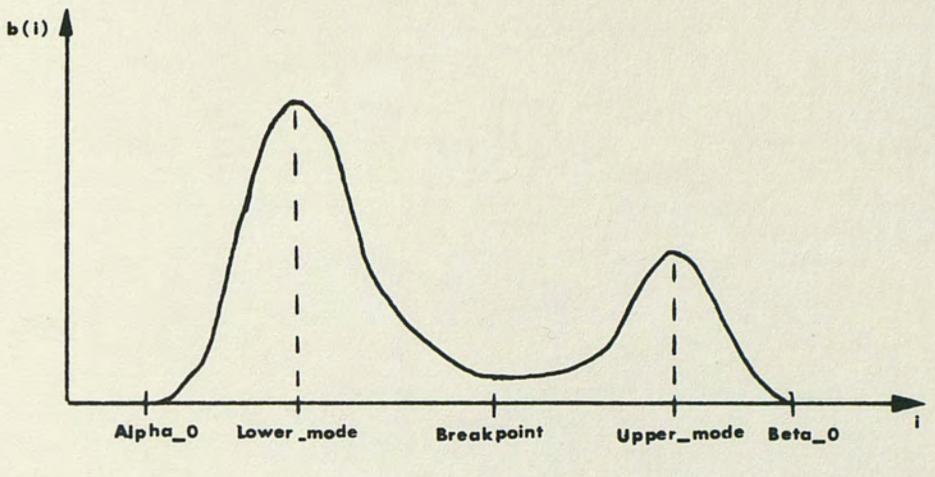


Fig. 24. Multiple histograms averaged together. (a) Image 16 histogram (b) Average of image 16 and 17 histograms (c) Average of image 16, 17 and 18 histograms (d) Average of image 16, 17, 18 and 20 histograms



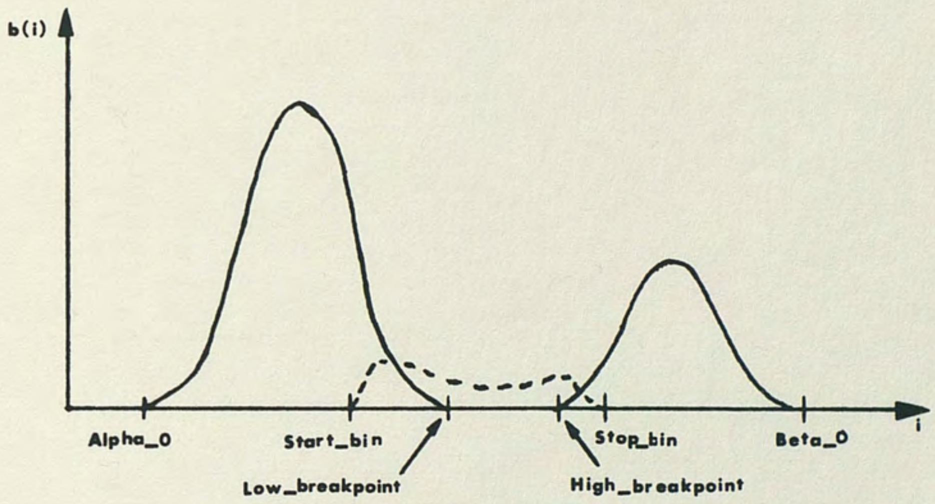


Fig. 25. Histogram model parameters

1. Alpha_0 and Beta_0 were determined. Alpha_0 and Beta_0 are defined as the first and last bins which contain a minimum of 10 pixels and have at least 9 adjacent bins with a minimum of 10 pixels each. The value 10 was large enough eliminate insignificant bins but small enough to impose boundaries without noticeably affecting results.
2. A Breakpoint value of (Alpha_0 + Beta_0)/2 was established to provide a temporary peak separation point.
3. The Lower_mode and Upper_mode, defined as the bins in the lower and upper peaks which contained the most pixels, were found.
4. The Low_breakpoint and High_breakpoint, which correspond to the bins diametrically opposed to Alpha_0 and Beta_0 with respect to Lower_mode and Upper_mode were determined.
5. The sum of the histogram values for each region of the histogram were computed. Lower_sum equals the summation of histogram values from Alpha_0 to Low_breakpoint, inclusive. Middle_sum is the summation from Low_breakpoint to High_breakpoint, exclusive, and Upper_sum is the summation from High_breakpoint to Beta_0, inclusive.
6. The lower and upper regions were divided by their respective sums to create peaks normalized to an area of one.

7. The mean and variance of these two regions were computed by:

$$\text{Lower\_mean} = \bar{b}_l = \sum_{i = \text{Alpha\_0}}^{\text{Low\_breakpoint}} i \cdot b(i)$$

$$\text{Lower\_variance} = \sigma_l^2 = \sum_{i = \text{Alpha\_0}}^{\text{Low\_breakpoint}} (i - \bar{b}_l)^2 \cdot b(i)$$

$$\text{Upper\_mean} = \bar{b}_u = \sum_{i = \text{High\_breakpoint}}^{\text{Beta\_0}} i \cdot b(i)$$

$$\text{Upper\_variance} = \sigma_u^2 = \sum_{i = \text{High\_breakpoint}}^{\text{Beta\_0}} (i - \bar{b}_u)^2 \cdot b(i)$$

where i is the histogram bin and b(i) is the histogram value at bin i.

8. Treating the two normalized peaks as probability density functions, the mean and variance of each peak were used to model the peaks as discrete Gaussian probability functions. Each modeled peak was determined by the relation

$$\text{Pdf}(i) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(i - \bar{b})^2}{2\sigma^2}} \quad \text{for } i = \text{Alpha\_0 to Beta\_0}$$

where i = the histogram bin number

σ = lower or upper variance respectively and

$\bar{b}$ = lower or upper mean respectively.

9. Subject to several constraints, the modeled histogram peaks were scaled to maintain proportionality to each other and to the original histogram and then subtracted from the original histogram. The result was the dashed center region of figure 25. The start and stop bins are functions of the original histogram values, model values, and statistics of the associated density function and were established to avoid abrupt slope changes.
10. The resultant center region was modeled as various order polynomial functions.
11. The resulting bimodal histogram model is of the following form:

$$b(i) = K_1\left[\frac{1}{\sqrt{2\pi}\sigma_l^{2}}\, e^{\frac{-(i-\bar{b}_l)^2}{2\sigma l^2}}\right]\delta_1(i\text{-Alpha\_0})\delta_1(\text{Beta\_0-}i)$$

$$+\left[K_2 i^5+K_3 i^4+K_4 i^3+K_5 i^2+K_6 i+K_7\right]$$

$$\delta_1(i\text{-Start\_bin})\delta_1(\text{Stop\_bin-}i)$$

$$+\ K_8\left[\frac{1}{\sqrt{2\pi}\sigma_u^{2}}\, e^{\frac{-(i-\bar{b}_u)^2}{2\sigma_u^2}}\right]\delta_1(i\text{-Alpha\_0})\delta_1(\text{Beta\_0-}i)$$

for $i = 0, 1, \ldots 255$

where K_1 and K_8 are proportionality constants K_2 through K_7 are polynomial coefficients and δ_1 is the unit step function.

With the basic premise for developing the histogram model established, a more detailed look at a few items is needed. First, although the Gaussian density model is based on a sample mean and variance, initial approximations were not satisfactory. The partitioning of the original histogram into three regions left the lower and upper regions with significantly greater values at the respective breakpoints than at the absolute lower and upper limits. This caused the mean value of the peaks to shift toward the center. Therefore, the final modeling program substituted the modes for the means of the two peaks.

The polynomial equation representing the histogram valley is based on a least squares curve fitting routine utilizing the Gauss-Jordan elimination method to solve for the coefficients. The curve fit subroutine was adapted from a program by Miller (1981). As currently dimensioned, it allows up to 256 data points and a fifth order solution, both of which can be increased. The adapted subroutine can be found in Appendix A and a brief explanation of the Gauss-Jordan method in Appendix B. To determine the order of the polynomial to be used and the type of smoothing to use on the histogram before modeling, the valley region of a series of histograms was modeled. Unsmoothed, averaging window smoothed, and median window smoothed histogram valley regions were modeled using second, third, fourth, and fifth order polynomials. Eleven histograms of inner layer boards were modeled. For each curve fit of the valley region a correlation coefficient based on a comparison of the calculated model values and the original values was calculated. Appendix C contains an explanation of the equation used by the histogram modeling subroutine for computing the correlation coefficient.

In order to reduce the data to a manageable amount and still be able to observe the effects of pre-processing the histogram and using various order polynomial models, the mean correlation coefficient value for each method of pre-processing (in conjunction with the third, fourth, and fifth order polynomial models) was computed. These mean values were then plotted to demonstrate the effects of histogram smoothing and polynomial order. These plots are shown in figures 26 and 27. The most obvious conclusion drawn from these plots is that the fifth order polynomial consistently provides the highest levels of correlation. After an initial improvement, correlation values tend to decrease as larger median filter windows are used to smooth the histogram. Small median filter windows remove large excursions while large windows create a staircase effect. Averaging windows, on the other hand, reduce the large excursions, but tend to create smoother bin to bin transitions as the window size is increased. Hence, the correlation coefficient generally improves as the window size increases.

Because of the relatively high mean correlation coefficients of the fifth order curve fits and the ease of window averaging a histogram, it was initially decided to model only the original histogram and a three bin wide averaging window smoothed version using a fifth order approximation for the valley region. The data base established for these histograms, however, pointed out an apparent discrepancy. Though excursions from the true curve were indeed minimized, the polynomial curve tended to oscillate about the original set of data points. Reducing the order of the

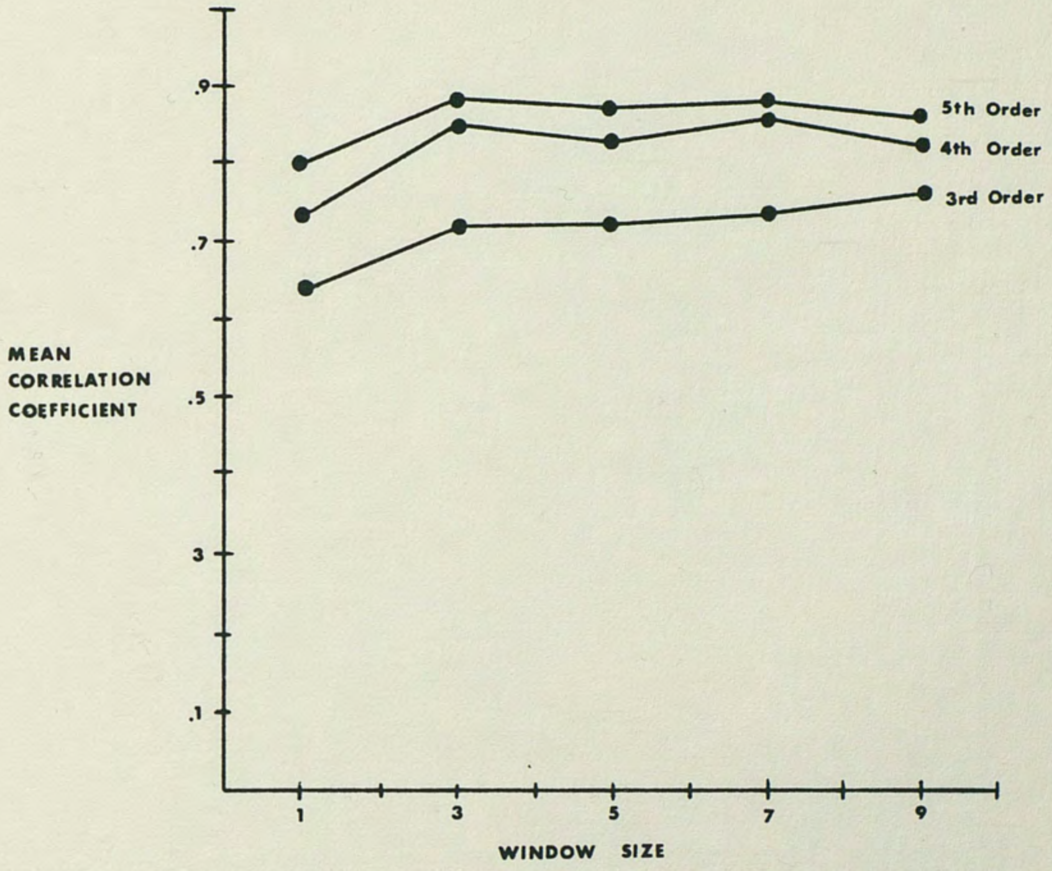


Fig. 26. Comparison of mean correlation coefficients for various order polynomial fits for median smoothed histograms

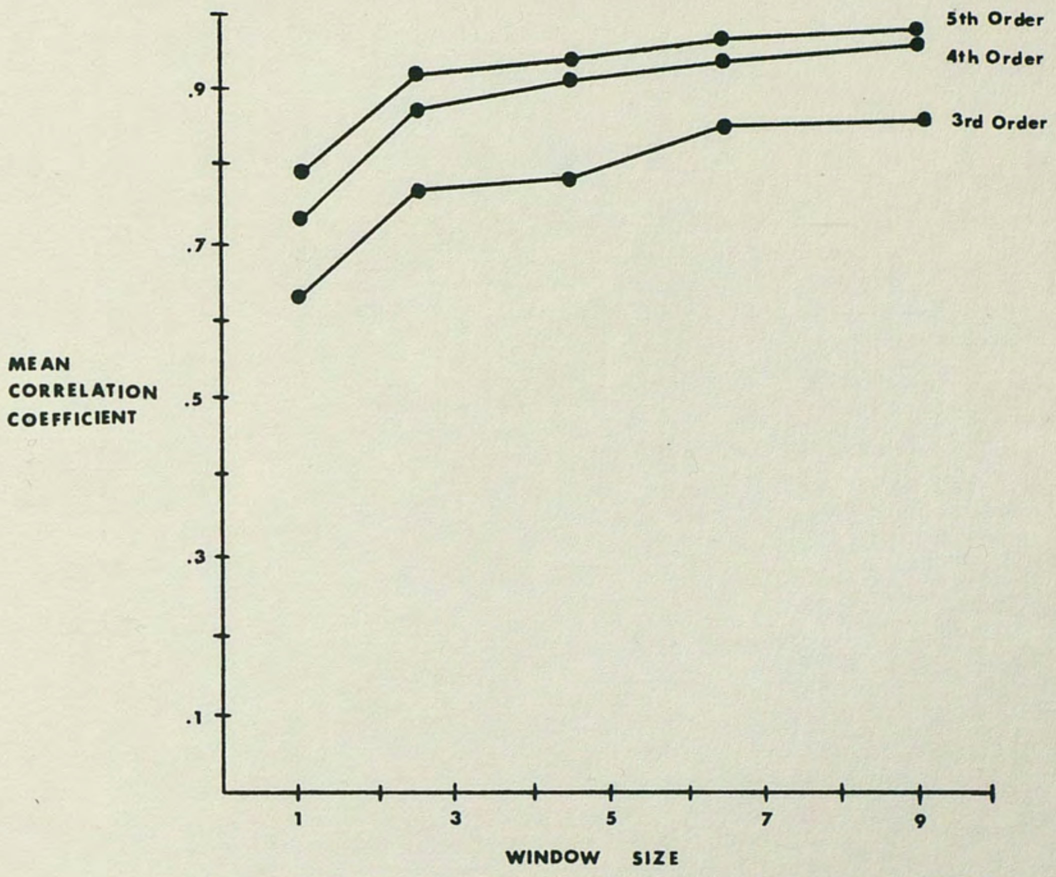


Fig. 27. Comparison of mean correlation coefficients for various order polynomial fits for average smoothed histograms

polynomial model reduced the oscillation effect while increasing the residual errors. In keeping with the goal of determining the general characteristics of the valley region, models using second, third and fourth order polynomial curve fits for the window averaged histogram were added to the histogram data base. A comparison of a histogram and its model is shown in figure 28.

Threshold Selection

The histogram enhancement and modeling techniques discussed in the previous section were used to create a histogram data base to facilitate an investigation into alternate thresholding algorithms. As with previous work, the histogram data base was restricted to inner layer board histograms. This resulted in a group of bimodal histograms with the dominate population occurring at the lower pixel values.

Once the data base was established, several thresholding schemes were implemented. Though the approaches varied from relatively simple to moderately difficult, a prime constraint was the ability to easily implement these algorithms using the current CPVP configuration. The median smoothing filter used to smooth an image is an exception to this constraint, but, a median smoothed image has been included because of the unique yet promising median filter characteristics. The following paragraphs describe the threshold selection techniques performed on the histogram data base by the program in Appendix A.

Method 1 - The first approach used was the standard histogram method of selecting the segmentation threshold as the histogram bin

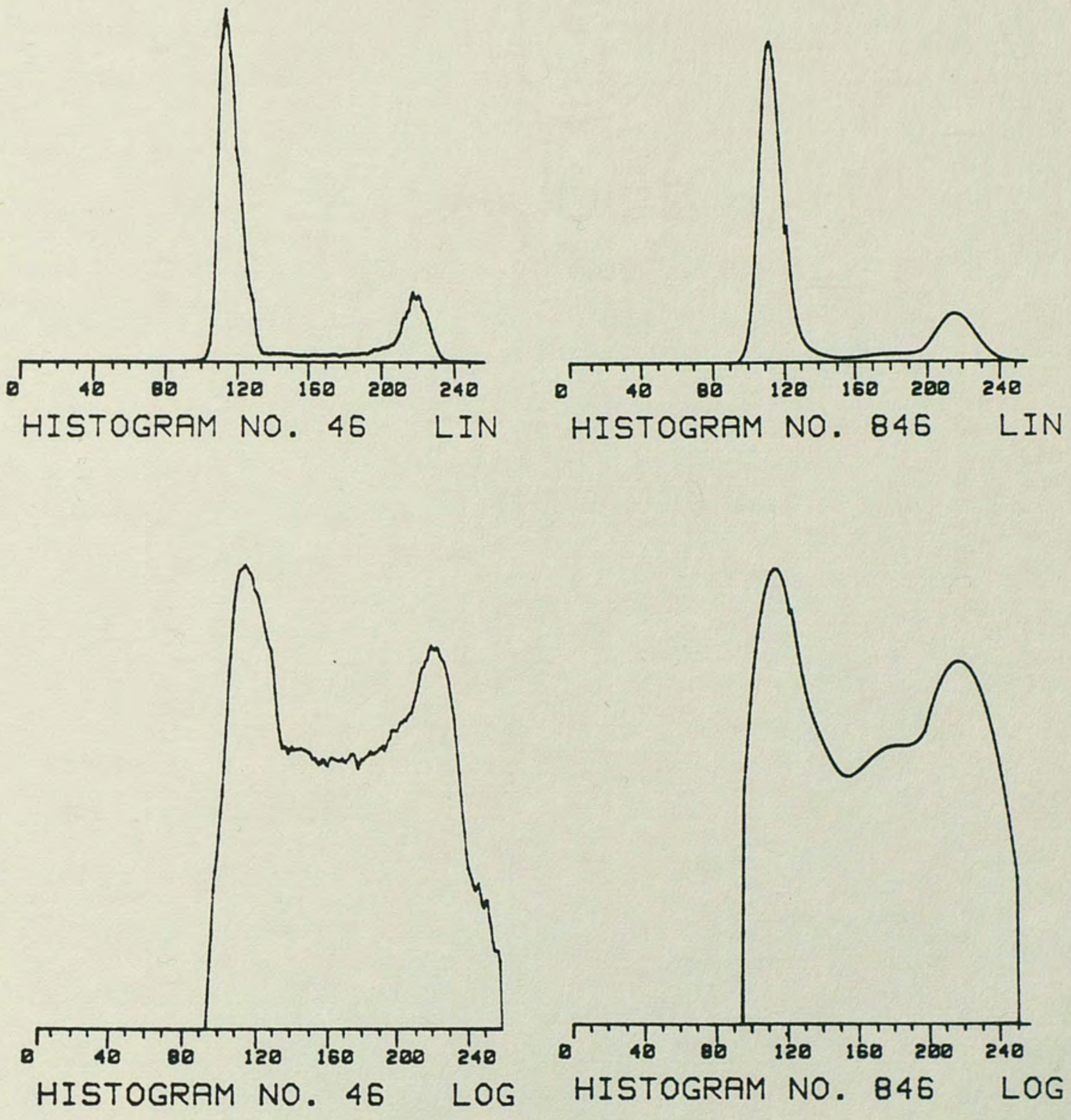


Fig. 28. Comparison of a histogram and its model

with the minimum number of pixels. There is, of course, the added restriction that the threshold occur in the valley region of the bimodal histogram to avoid trivial cases. This restriction was included in all threshold selections. There were two other possibilities taken into account. In the case of multiple non-neighboring minimum bins (a minimum bin is the bin which contains the least number of pixels) the bin closest to the upper mode was selected. In the case of adjacent minimum bins, the center bin of the group was selected. Occasionally this led to a half-bin increment which was reported for the sake of thoroughness.

Method 2 - The second method selected the threshold as the bin halfway between the lower and upper modes. Though not realizable in hardware, bin values with remainders of 0.5 were reported to avoid a loss of information. The threshold can easily be rounded up or truncated, but once done the remainder information is not retrievable.

Method 3 - The third approach chose the threshold as the bin halfway between the absolute minimum and absolute maximum active histogram bins. The method is similar to that used by several automatic television tracker systems. The tracker approach also allows for different offsets between the minimum and maximum limits to adjust the sensitivity of the system. Though simplistic in its approach, the simple features of a PCB seemed to make a look at this approach worthwhile.

Method 4 - The fourth method is identical to the third method with two exceptions. First, the minimum pixel count for a bin to be considered valid was increased from one to ten. Second, a bin had to have at least ten adjacent valid neighboring bins to be considered an absolute minimum or maximum. The result was a noise cleaning effect on the histograms and is noticeable in the modeled histograms such as those of figure 28. This noise cleaning approach synthesizes the integration effects found in the analog circuitry of several television tracking systems.

Method 5 - The fifth approach is based on minimizing the pixel classification error based on the probability densities of the two populations corresponding to background and land. As discussed earlier, an interim step of the histogram modeling process was to compute Gaussian based functions to describe the two histogram peaks. Based on this interim model (one that does not include the valley region approximation) a threshold was found to minimize pixel misclassification. A general derivation for the optimum threshold is given by Rosenfeld and Kak (1982). For the two class Gaussian density modeled histogram the optimum threshold reduces to the bin between the two modes with the minimum pixel count which corresponds to the intersection of the two functions. Therefore, the threshold selection reported for this approach was the intersection bin of the two Gaussian functions.

Method 6 - The sixth method utilized the completed histogram model for selecting the threshold. The polynomial curve used to represent the valley region provided a smoothed approximation exhibiting the general tendencies of the original histogram. The threshold was again selected based on a relative minimum as in method one.

Implementation of these threshold methods resulted in a collection of possible threshold values to use for segmenting images. With these thresholds established, original plans called for each image in the data base to be displayed via the CAPS system, manually thresholded at a continuously increasing intensity and the results noted. Bearing in mind that thresholding at too low a level causes background to be classified as land and thresholding at too high a level causes land areas to be classified as background, a suitable threshold range which would effectively eliminate background clutter and still maintain sufficient land width for subsequent processing was to be determined. Evaluation of the different thresholding schemes could then include a comparison of the various threshold results against the acceptable range. Because of equipment failure the images stored on magnetic tape and also on a hard disc unit were not retrievable in the time frame necessary for this paper.

However, several general tendencies of the generated thresholds can be noted. The thresholds computed by the different algorithms are given in Table 4. Including both smoothed and unsmoothed histograms the most consistent thresholds occurred at the center bin between the two modes, at the center bin between the minimum and maximum bins with T=10 and at the minimum bin of the Gaussian curve fit. Maximum threshold variations were two, three and five bins for these three cases. The least consistent selection occurred for the standard histogram method or the minimum valued bin between the

TABLE 4

THRESHOLD VALUES

Histogram Number	Minimum Bin Between Modes	Center Bin Between Modes	Center Bin Between α_0 and β_0 T=0	Center Bin Between α_0 and β_0 T=10	Minimum Bin of Gaussian Fit	Minimum Bin of 2nd Order Curve Fit	Minimum Bin of 3rd Order Curve Fit	Minimum Bin of 4th Order Curve Fit	Minimum Bin of 5th Order Curve Fit
0	137	142.5	140.5	140	142	132	124.5	142	144
30	136	143	159	140	141	128	128.5	125	126
60	135	142	160.5	141	139	126	129	137.5	126.5
150	136.5	141.5	155.4	140	139	124	130	140.5	143
180	135	142	157.5	140	140	125	129	141	142.5
1	124	152	166	151.5	146	129.5	129.5	152	128
31	137	152	160	154	143	130.5	131	151	130
61	136	152	158.5	153.5	144	131.5	132	151	131
151	158.5	152	171	154	144	129	129	148.5	153.5
181	137.5	152	159.5	154	145	161.5	142	142.5	152.5
2	124	154	151	150.5	150	135	139.5	150	150.5
32	124	154	151.5	150.5	150	136	138.5	152	153
62	147	153	152.5	151	148	132	135	152	150.5
152	148	153.5	151.5	150.5	149	134	140	150.5	151
182	146.5	153	151.5	150.5	149	132	137	151.5	151

TABLE 4 - Continued

3	124	113.5	100.5	114.5	110	119.5	105	97	93
33	118	113.5	99.5	114.5	110	119.5	104	97	93
63	124	113	100	115	109	119.5	104	98	94
153	125	113.5	99	114.5	109	119.5	104	97	93
183	117	115	99	114.5	112	127	103.5	99	94
4	124	119	124	122	106	99	99.5	118	120.5
34	124	118.5	126	122	106	97	97.5	97	118
64	124	119	122	122	107	100	100	119	120
154	125	119	125.5	122	106	101	101.5	119	100
184	103.5	117.5	125.5	122	104	129	99.5	115.5	116
5	124	123	134.5	127	110	135	102	122	119.5
35	124	123	134.5	127.5	110	134	108	117.5	119.5
65	124	123	136.5	127	110	133.5	104.5	122	120
155	125	122.5	134.5	127	110	98	98	120	122
185	125	123	134.5	127	110	135	104	121.5	119
6	124	122	137.5	123.5	115	127.5	111	104	100
36	124	121	138	123.5	114	127	111	104	100
66	124	121.5	140.5	124	115	129.5	112	106	101
156	125	122	137.5	123.5	116	130	113	105	100
186	125	122.5	137	123.5	116	101	102	102.5	100

TABLE 4 - Continued

16	176	164.5	173.5	171.5	151	174	154	148.5	148
46	175	164.5	173.5	171.5	151	171	153.5	148.5	149
76	157	164.5	173	171.5	152	172.5	154	150.5	151
166	175.5	164.5	173	171.5	150	171	153	147	145.5
196	156	164.5	173	171.5	151	172	153.5	147.5	145.5
17	175	168.5	174.5	173	156	183	151.5	150.5	140
47	164	167	174.5	173	155	174.5	156.5	152	152.5
77	163	167.5	174	172.5	156	176	157	155.5	157
167	164	166.5	174	173	153	174	156	150.5	150.5
197	165.5	167.5	173	173	155	175.5	156.5	151.5	150.5
18	175	166.5	175	171	154	176.5	152	145.5	140
48	164	167	159	172	156	175.5	156.5	151.5	151.5
78	163	167	156.5	171.5	156	175	156.5	154	156
168	163	167	174.5	171.5	156	176	156	151.5	150.5
198	163	166.5	158	172	154	174	155.5	152.5	153
20	175	164.5	173	169.5	154	177.5	154.5	149.5	146.5
50	158	164.5	172.5	169	154	178	155	149.5	146.5
80	157	164	158	169	153	172	153.5	149	148
170	158.5	163	173	170	149	171	152	147.5	147
200	157.5	163.5	173	170	150	175	154	156.5	153

modes. Values varied up to 35 bins. Consistency of the polynomial curve fits fell between these two extremes.

Obviously the effectiveness of each method is not directly proportional to the complexity of the algorithm. Thresholds based on geometrical properties are much easier and much faster to compute than those based on Gaussian or polynomial modeling. Based on implementation effort, processing time and consistency, the geometrical based algorithms seem to provide the best alternative. Though this paper arbitrarily chose the midpoint between the two modes as the threshold, it could easily be fixed elsewhere or adjusted dynamically. Of course parameters discussed elsewhere in this paper must be considered. For example, a geometrical based amplitude thresholding scheme must assume that background and land areas are uniform in intensity and different enough that a threshold in the "uncertain" region would not mislabel a background region. This would require some control over the characteristics of the PCB's and the dynamic range of the image. The greater the dynamic range the more flexibility in the threshold range. Most importantly a definitive correlation between true land size, segmented land size and threshold level would have to be established.

Table 4 points out general tendencies but does not attempt to provide an absolutely final answer. Further work to be done must include establishing a larger data base and developing tools to more easily collect image and threshold data. It is felt that the results in this paper help to point out areas needing further investigation while providing the foundation necessary for carrying out this work.

SUMMARY AND CONCLUSION

This paper progressed from introducing the concept of automated visual inspection in general to looking at specific threshold selection algorithms. The need for automated visual inspection was examined and several current vision systems attempting to meet these needs were presented. Four basic approaches to automated printed circuit board inspection were given.

The Automatic Board Assembly, Inspection and Test (ABAIT) system was described. In particular, emphasis was placed on the visual inspection detail station, its operation, and its objectives. To establish a foundation for further work, fundamentals of image processing were reviewed. Image enhancement including image smoothing and image sharpening was stressed along with segmentation principles. The Circular Pipeline Video Processor (CPVP) was then introduced. The CPVP hardware and related software were described followed by a sample operational illustration.

The purpose for investigating alternate segmentation approaches was discussed followed by an analysis of captured CPVP imagery. The analysis was introduced by a description of the analysis tools developed during the course of this paper, particularly the image transfer capability, and a description of the subsequent image data base. An analysis of these images resulted in the evaluation of the CPVP image capture path as having a first order response.

The investigation of alternate segmentation techniques began with researching techniques presented in recent publications and in general industry use. Histogram enhancement was discussed and a histogram model developed. A histogram data base was established using captured CPVP imagery and the various histogram enhancement and modeling techniques developed earlier in the paper. Thresholds for segmentation based on several algorithms were selected using this histogram data base. A comparison of the relative performance of these algorithms indicated that the consistency of the thresholds was not directly proportional to the complexity of the algorithm. The most consistent threshold selection was made based only on the modes of the histogram.

Evaluating the results obtained from the image analysis and the alternate segmentation techniques, a threshold selection scheme which selects the threshold based on the histogram modes, CPVP bandwidth and pixel resolvability appears to be the most promising.

The concept of the automated visual inspection process has been successfully demonstrated and with the incorporation of the analysis tools developed in this paper, the results of the image analysis and the results of the alternate threshold selection techniques it is felt that the system will be able to progress from a proof of principle system to a fully operational system capable of accurately inspecting printed circuit boards for Martin Marietta Aerospace.

APPENDIX A

COMPUTER PROGRAM

```
10!    RE-STORE "10!   RE-STORE "HISTOGRAM7"
20     OPTION BASE 0
30     DIM Norm_histo_val(0:255),Pdfa(0:255),Pdfb(0:255),Pdf(0:255)
40     DIM Pdfc(0:255),Pdfd(0:255),Pdfu(0:255),Pdfv(0:255),Histo_val_sum(0:255)
50     DIM Const_vector(6),Coeff(6,6),Soln_vector(6),Error_vector(6)
60     DIM Soln_matrix(6,1),B_coeff(6,6),X_value(256),Data_vector(120)
70     DIM Y_data(246),Y_calculated(256),Residual(256),Data_matrix(120,6)
80     INTEGER Histo_val(0:255),Median_value(0:255),Histo_value(1:5,0:255)
90     INTEGER Histo_bin,Top_of_sort,Sort_value,Temp(9),Bin_number
100    INTEGER Work_matrix(6,3)
110    INTEGER Num_rows,Num_columns,Num_const_vctrs,Row_index,Column_index
120    INTEGER Poly_order,Max_length,J,K,L,M
130    ON ERROR GOSUB Error_handler
140    FOR J=0 TO 9      ! TURN OFF OPERATING SYSTEM KEY LABELS
150       ON KEY J LABEL " " GOTO 150
160    NEXT J
170    PRINT CHR$(12)      ! CLEAR TEXT.
180    DISP CHR$(12)       ! CLEAR DISPLAY.
190    GCLEAR              ! CLEAR GRAPHICS.
200    GRAPHICS OFF        ! DISABLE GRAPHICS.
210 Loop1:                 ! LABEL AND ACTIVATE SOFT KEYS.
220    ON KEY 0 LABEL "INPUT_HISTO" GOSUB Input_histo ! READ HISTO FROM KEYBOARD.
230    ON KEY 1 LABEL "PRINT_HISTO" GOSUB Print_histo ! PRINT THE HISTO FILE.
240    ON KEY 2 LABEL "CHANGE_FILE" GOSUB Change_file ! CHANGE VALUES IN FILE.
250    ON KEY 3 LABEL "PLOT_HISTO" GOSUB Plot_histo   ! PLOT THE HISTOGRAM.
260    ON KEY 5 LABEL "AVERAGE_SMOOTH" GOSUB Average_smooth! SMOOTH HISTOGRAM.
270    ON KEY 6 LABEL "MEDIAN_SMOOTH" GOSUB Median_smooth ! SMOOTH HISTOGRAM.
280    ON KEY 7 LABEL "GROUP_AVERAGE" GOSUB Group_average ! AVG HISTOS TOGETHER.
290    ON KEY 8 LABEL "PDF_MODEL" GOSUB Pdf_model ! MODEL THE HISTOGRAM.
300    GOTO Loop1
```

```
310 Input_histo:  ! THIS SUBROUTINE READS IN HISTOGRAM VALUES FROM THE
321               !   KEYBOARD AND WRITES THEM TO A HISTO FILE.
330     PRINT CHR$(12)
340     GCLEAR
350     GRAPHICS OFF
360     PRINT TABXY(1,20);"ENTER THE HISTOGRAM FILE NUMBER THAT VALUES"
370     PRINT "  ARE TO BE STORED IN. "
380     INPUT File_number  ! INPUT FILE_NUMBER FROM KEYBOARD.
390     PRINT CHR$(12)
400     IF File_number<0 OR File_number>100 THEN     ! ERROR TRAP.
410        PRINT "FILE NUMBER MUST BE 0 TO 100 INCLUSIVE"
420        BEEP 1500,.3
430        GOTO Input_histo
440     END IF
450     PRINT CHR$(12)
460     DISP CHR$(12)
470     FOR Bin_number=0 TO 255    ! LOOP READS IN VALUES FROM KEYBOARD.
480        DISP "   ENTER BIN";Bin_number;"VALUE    ";
490        INPUT Histo_val(Bin_number)
500        PRINT TAB((Bin_number MOD 10)*7+6);Histo_val(Bin_number);
510     NEXT Bin_number
520 Correct:     !                  ! THIS SECTION ALLOWS FOR CORRECTIONS
530     PRINT CHR$(12)              !   AFTER VALUES ARE READ IN.
540     FOR Bin_number=0 TO 255     ! PRINT FORMATTED HISTO VALUES.
550        PRINT TAB((Bin_number MOD 10)*8);Histo_val(Bin_number);
560     NEXT Bin_number
570     DISP "IS A CORRECTION NECESSARY? (Y/N)   ";
580     INPUT Correction_nec$
590     IF Correction_nec$<>"Y" THEN GOTO Create_file
600        DISP "HOW MANY VALUES NEED TO BE CORRECTED  ";
610     INPUT Correction_num
620     IF Correction_num>0 THEN
630       FOR Correction=1 TO Correction_num
640         DISP " WHICH HISTOGRAM BIN IS TO BE CHANGED";
650         INPUT Bin_number
660         DISP " WHAT IS THE CORRECT HISTOGRAM VALUE";
670         INPUT Corrected_value
680         Histo_val(Bin_number)=Corrected_value    ! HISTO_VAL IS CORRECTED.
690       NEXT Correction
700       GOTO Correct
710     END IF
720     GOSUB Create_file
730     DISP "READY"
740     BEEP 1500,.2
750     RETURN
```

```
760 Print_histo:      ! THIS SUBROUTINE PRINTS HISTOGRAM VALUES
770                   !   PREVIOUSLY STORED ON DISC.
780     PRINT
790     GCLEAR
800     DISP CHR$(12);"ENTER HISTO FILE NUMBER TO PRINT  ";
810     INPUT File_number
820     GOSUB Get_histo_file
830     DISP "IS PRINTOUT TO GO TO THE PRINTER (Y/N)  ";! GOES TO SCREEN IF <>Y.
840     INPUT Print_device$
850     IF Print_device$="Y" THEN PRINTER IS 706        ! 706 IS LINE PRINTER.
860     PRINT
870     PRINT TAB(26);"HISTOGRAM NO.";File_number
880     PRINT
890     FOR Col_header=0 TO 9
900        PRINT TAB(Col_header*7+6);Col_header;        ! PRINT COLUMN LABELS.
910     NEXT Col_header
920     PRINT
930     PRINT
940     FOR Bin_number=0 TO 255   ! PRINT ROW LABELS AND HISTO VALUES.
950        IF INT(Bin_number/10)=Bin_number/10 THEN PRINT TAB(0);Bin_number/10;
960        PRINT TAB((Bin_number MOD 10)*7+6);Histo_val(Bin_number);
970     NEXT Bin_number
980     PRINT
990     PRINTER IS 1              ! REASSIGN SCREEN AS PRINT DEVICE.
1000    BEEP 1500,.2
1010    DISP "READY"
1020    RETURN
```

```
1030 Change_file:    !  THIS SUBROUTINE ALLOWS INDIVIDUAL VALUES IN THE HISTO
1040                 !    FILES TO BE CHANGED (CORRECTED) AND THE FILE UPDATED.
1050   GCLEAR
1060   PRINT CHR$(12)
1070   DISP "ENTER HISTO FILE NO. TO HAVE VALUES CHANGED   ";
1080   INPUT File_number
1090   GOSUB Get_histo_file
1100 Input_change:    !
1110   PRINT TAB(26);"HISTOGRAM NO.";File_number
1120   PRINT
1130   FOR Col_header=0 TO 9
1140      PRINT TAB(Col_header*7+6);Col_header;
1150   NEXT Col_header
1160   PRINT
1170   PRINT
1180   FOR Bin_number=0 TO 255
1190      IF INT(Bin_number/10)=Bin_number/10 THEN PRINT TAB(0);Bin_number/10;
1200      PRINT TAB((Bin_number MOD 10)*7+6);Histo_val(Bin_number);
1210   NEXT Bin_number
1220      DISP "HOW MANY VALUES NEED TO BE CORRECTED ? (0 TO END)   ";
1230   INPUT Correction_num
1240   IF Correction_num>0 THEN
1250    FOR Correction=1 TO Correction_num
1260     DISP " WHICH HISTOGRAM BIN IS TO BE CHANGED ";
1270     INPUT Bin_number
1280     DISP " WHAT IS THE CORRECT HISTOGRAM VALUE ";
1290     INPUT Corrected_value
1300     Histo_val(Bin_number)=Corrected_value
1310    NEXT Correction
1320    GOTO Input_change
1330   END IF
1340   PURGE "HISTO"&VAL$(File_number)  ! DELETE THE INCORRECT FILE.
1350   GOSUB Create_file                ! RESTORE WITH UPDATED FILE.
1360   BEEP 1500,.1
1370   PRINT CHR$(12)
1380   DISP "READY"
1390   RETURN
```

```
1400 Plot_histo: !  THIS SUBROUTINE PLOTS A HISTOGRAM FILE ON A LINEAR OR
1410            !   LOGARITHMIC SCALE WITH SOLID BARS OR ONLY THE OUTLINE.
1420   PRINT
1430   GCLEAR
1440   Xorg=120           ! GRAPHICS X ORIGIN.
1450   Yorg=320           ! GRAPHICS Y ORIGIN.
1460   DISP "ENTER HISTO FILE NUMBER TO PLOT  ";
1470   INPUT File_number
1480   GOSUB Get_histo_file
1490 Enter_scale:    ! INPUT WHAT SCALE IS TO BE USED IN PLOTTING.
1500   DISP "LINEAR OR LOGARITHMIC  SCALE (LIN OR LOG)  ";
1510   INPUT Scale$
1520   IF Scale$<>"LIN" AND Scale$<>"LOG" THEN
1530      PRINT CHR$(12);" INVALID SCALE ENTERED. TRY AGAIN"
1540      BEEP 1500,.2
1550      GOTO Enter_scale
1560   END IF
1570   PRINT CHR$(12)
1580   DISP "IS PLOT POINT-TO-POINT INSTEAD OF BARS (Y/N) ";
1590   INPUT Point_to_point$
1600   GINIT                                  ! INITIALIZE GRAPHICS.
1610   GCLEAR                                 ! CLEAR GRAPHICS ON SCREEN.
1620   GRAPHICS ON                            ! ENERGIZE GRAPHICS.
1630   PEN 1                                  ! NORMAL POLARITY (BLK ON WHT).
1640   Previous_x=Xorg/4                      ! INITIALIZE VALUE.
1650   Previous_y=Yorg/4                      ! INITIALIZE VALUE.
1660   FOR Histo_bin=0 TO 255
1670    IF INT(Histo_bin/40)=Histo_bin/40 THEN
1680      MOVE (Xorg-10)/4,(Yorg-Histo_bin)/4
1690      DRAW Xorg/4,(Yorg-Histo_bin)/4       ! DRAW MAJOR TIC MARKS.
1700      ELSE
1710       IF INT(Histo_bin/10)=Histo_bin/10 THEN
1720          MOVE (Xorg-4)/4,(Yorg-Histo_bin)/4
1730          DRAW Xorg/4,(Yorg-Histo_bin)/4   ! DRAW MINOR TIC MARKS.
1740         ELSE
1750            MOVE Xorg/4,(Yorg-Histo_bin)/4 ! DEFAULT START IS THE AXIS.
1760            IF Point_to_point$="Y" THEN DRAW Xorg/4,(Yorg-Histo_bin)/4
1770        END IF
1780    END IF
1790    IF Point_to_point$="Y" THEN MOVE Previous_x,Previous_y
1800    IF Histo_val(Histo_bin)<=0 THEN        ! THERE SHOULD BE NO NEGATIVE
1810                                           !   HISTO VALUES. DEFAULT TO 0.
1820       DRAW Xorg/4,(Yorg-Histo_bin)/4
1830       Previous_x=Xorg/4                   ! UPDATE POSITIONING VALUES TO
1840       Previous_y=(Yorg-Histo_bin)/4       !   BE USED AFTER AXIS IS DRAWN.
```

```
1850     ELSE
1860      IF Scale$="LIN" THEN                         ! DRAW LINEAR PLOT.
1870        DRAW (Xorg+Histo_val(Histo_bin)/60)/4,(Yorg-Histo_bin)/4
1880        Previous_x=(Xorg+Histo_val(Histo_bin)/60)/4
1890        Previous_y=(Yorg-Histo_bin)/4
1900      END IF
1910        IF Scale$="LOG" THEN                       ! DRAW LOG (BASE 10) PLOT.
1920          DRAW (Xorg+LGT(Histo_val(Histo_bin))*70)/4,(Yorg-Histo_bin)/4
1930          Previous_x=(Xorg+LGT(Histo_val(Histo_bin))*70)/4
1940          Previous_y=(Yorg-Histo_bin)/4
1950        END IF
1960    END IF
1970   NEXT Histo_bin
1980   DRAW Xorg/4,(Yorg-256)/4              !  CLOSE RIGHT END OF PLOT.
1990   MOVE (Xorg-52)/4,(Yorg)/4             !  RELATIVE POSITION FOR LABEL.
2000   DEG                                   !  DEGREE MODE FOR LDIR.
2010   LDIR 270                              !  LABEL DIRECTION IS VERTICAL.
2020   CSIZE 5
2030   LABEL "HISTOGRAM NO.";File_number;"  ";Scale$     ! WRITE LABEL.
2040   FOR Bin_number=0 TO 240 STEP 40                   ! THIS LOOP LABELS
2050       MOVE (Xorg-24)/4,(Yorg+16-Bin_number)/4       !   TIC MARKS.
2060       CSIZE 3
2070       LABEL Bin_number
2080   NEXT Bin_number
2090   PRINT "IS PLOT TO GO TO PRINTER ALSO? (Y/N)"
2100   INPUT Printer_plot$
2110   IF Printer_plot$="Y" THEN                  ! IF PLOT GOES TO PRINTER THEN
2120       DUMP DEVICE IS 706                     !   ASSIGN  DUMP DEVICE TO
2130       DUMP GRAPHICS                          !   PRINTER AND DUMP GRAPHICS.
2140   END IF
2150   PRINT CHR$(12)
2160   DISP "READY"
2170   BEEP 1500,.15
2180   RETURN
```

```
2190 Average_smooth:        !  THIS SUBROUTINE ALLOWS FOR AVERAGING HISTOGRAM
2200                        !    VALUES OF A STORED HISTO FILE USING VARIOUS
2210                        !    SIZE WINDOWS.
2220   GCLEAR
2230   DISP "ENTER HISTO FILE NUMBER TO SMOOTH  ";
2240   INPUT File_number
2250   PRINT CHR$(12)
2260   GOSUB Get_histo_file
2270 Avg_win_size:     !
2280   DISP "ENTER WINDOW SIZE (3 THRU 9)  ";
2290   INPUT Window_size
2300   IF Window_size<3 OR Window_size>9 THEN
2310      PRINT "WINDOW SIZE INVALID"
2320      BEEP 1200,.3
2330      GOTO Avg_win_size
2340   END IF
2350   DISP "NOW SMOOTHING....PLEASE WAIT"
2360   Initial_bin=(Window_size/2)-.5
2370   Final_bin=255-(Window_size/2)+.5
2380   FOR Histo_bin=Initial_bin TO Final_bin
2390      Summ=0
2400         FOR J=0 TO Window_size-1
2410            Summ=Histo_val(Histo_bin-(Window_size/2-.5)+J)+Summ
2420         NEXT J
2430      Histo_val(Histo_bin)=Summ/Window_size
2440   NEXT Histo_bin
2450   DISP "DO YOU WISH TO STORE THIS HISTOGRAM (Y/N) ";
2460   INPUT Store_avg_histo$
2470   IF Store_avg_histo$="Y" THEN
2480      DISP "ENTER HISTO FILE NUMBER  ";
2490      INPUT File_number
2500      GOSUB Create_file
2510   END IF
2520   DISP "READY"
2530   BEEP 1500,.2
2540   RETURN
```

```
2550 Median_smooth:      ! THIS SUBROUTINE ALLOWS FOR MEDIAN SMOOTHING OF
2560                     !   A STORED HISTO FILE WITH VARIABLE SIZE WINDOWS.
2570   GCLEAR
2580   DISP "ENTER HISTO FILE NUMBER TO SMOOTH   ";
2590   INPUT File_number
2600   PRINT CHR$(12)
2610   GOSUB Get_histo_file
2620 Input_win_size:    !
2630   DISP "ENTER WINDOW SIZE (3,5,7 OR 9)  ";
2640   INPUT Window_size
2650   Win_size=Window_size
2660   IF Win_size<>3 AND Win_size<>5 AND Win_size<>7 AND Win_size<>9 THEN
2670       PRINT "WINDOW SIZE MUST BE 3,5,7 OR 9."
2680       BEEP 1500,.3
2690       GOTO Input_win_size
2700   END IF
2710   DISP "NOW SMOOTHING...PLEASE WAIT"
2720   Initial_bin=(Window_size/2)-.5
2730   Final_bin=255-(Window_size/2)+.5
2740   FOR Histo_bin=Initial_bin TO Final_bin
2750     FOR J=0 TO Window_size-1
2760       Temp(J)=Histo_val(Histo_bin-(Window_size/2-.5)+J)
2770     NEXT J
2780     FOR Top_of_sort=0 TO Window_size-2
2790        FOR Sort_value=Top_of_sort+1 TO Window_size-1
2800          IF Temp(Sort_value)<Temp(Top_of_sort) THEN
2810              Hold_value=Temp(Top_of_sort)
2820              Temp(Top_of_sort)=Temp(Sort_value)
2830              Temp(Sort_value)=Hold_value
2840          END IF
2850        NEXT Sort_value
2860     NEXT Top_of_sort
2870     Median_value(Histo_bin)=Temp((Window_size/2-.5))
2880   NEXT Histo_bin
2890   BEEP 1500,.15
2900   DISP "DO YOU WISH TO STORE THIS HISTOGRAM  (Y/N) ";
2910   INPUT Store_mdn_histo$
2920   IF Store_mdn_histo$="Y" THEN
2930      DISP "ENTER HISTO FILE NUMBER  ";
2940      INPUT File_number
2950      FOR J=0 TO 255
2960        Histo_val(J)=Median_value(J)
2970      NEXT J
2980      GOSUB Create_file
2990   END IF
3000   DISP "READY"
3010   BEEP 1500,.2
3020   RETURN
```

```
3030 Group_average:      ! THIS SECTION ALLOWS FROM 2 TO 5 HISTOGRAMS TO
3040                     !    BE AVERAGED TOGETHER AND THE RESULT STORED.
3050   GCLEAR
3060   PRINT CHR$(12)
3070   DISP "HOW MANY HISTOGRAMS TO AVERAGE TOGETHER ";
3080   INPUT Histo_quantity
3090   IF Histo_quantity>5 THEN GOTO Group_average    ! MAXIMUM OF 5 HISTOS.
3100   IF Histo_quantity<2 THEN GOTO Group_avg_end    ! MINIMUM OF 2 HISTOS.
3110   FOR J=1 TO Histo_quantity  ! READ HISTOGRAMS AND TRANSFER INTO
3120                              !   A TWO DIMENSIONAL ARRAY.
3130       DISP "ENTER HISTOGRAM FILE NUMBER";J;" ";
3140       INPUT File_number
3150       GOSUB Get_histo_file
3160       FOR Histo_bin=0 TO 255
3170           Histo_value(J,Histo_bin)=Histo_val(Histo_bin)
3180       NEXT Histo_bin
3190   NEXT J
3200   DISP "WORKING...PLEASE WAIT"
3210   FOR Histo_bin=0 TO 255          ! PERFORM AVERAGING ALGORITHM.
3220       Histo_val_sum(Histo_bin)=0  !  INITIALIZE ARRAY.
3230       FOR J=1 TO Histo_quantity
3240           Histo_val_sum(Histo_bin)=Histo_val_sum(Histo_bin)+Histo_value(J,Hi
sto_bin)
3250       NEXT J
3260       Histo_val(Histo_bin)=Histo_val_sum(Histo_bin)/Histo_quantity
3270   NEXT Histo_bin
3280   BEEP 1500,.1
3290   DISP "DO YOU WISH TO STORE THE RESULTANT HISTOGRAM (Y/N) ";
3300   INPUT Store_resultant$
3310   IF Store_resultant$="Y" THEN
3320       DISP "ENTER HISTO FILE NUMBER ";
3330       INPUT File_number
3340       GOSUB Create_file   ! STORE THE RESULTANT HISTOGRAM.
3350   END IF
3360 Group_avg_end:    !
3370   DISP "READY"
3380   BEEP 1500,.2
3390   RETURN
```

```
3400 Pdf_model:     ! THIS SUBROUTINE MODELS THE HISTOGRAM  BASED ON A
3410                !  COMBINATION OF GAUSSIAN BASED FUNCTIONS AND A
3420                !  POLYNOMIAL FUNCTION.
3430   GCLEAR
3440   PRINT CHR$(12)
3450   DISP "ENTER HISTO FILE_NUMBER TO MODEL  ";
3460   INPUT File_number
3470   GOSUB Get_histo_file                    ! GET HISTOGRAM VALUES.
3480 Input_model:    !
3490   PRINT CHR$(12)
3500    Model$="G"   ! GAUSSIAN BASED MODEL
3510   PRINT "HISTO FILE NUMBER ";File_number
3520   PRINT
3530 Find_alpha_0:   !
3540   FOR Histo_bin=0 TO 255
3550      FOR Offset=0 TO 9
3560         IF Histo_val(Histo_bin+Offset)<=10 THEN GOTO 3600
3570      NEXT Offset
3580      Alpha_0=Histo_bin
3590      GOTO Find_beta_0
3600   NEXT Histo_bin
3610 Find_beta_0:    !
3620   FOR Histo_bin=255 TO 0 STEP -1
3630      FOR Offset=0 TO -9 STEP -1
3640         IF Histo_val(Histo_bin+Offset)<10 THEN GOTO 3680
3650      NEXT Offset
3660      Beta_0=Histo_bin
3670      GOTO Set_breakpoint
3680   NEXT Histo_bin
3690 Set_breakpoint:    !
3700   Breakpoint=(Alpha_0+Beta_0)/2
3710   PRINT "BREAKPOINT";Breakpoint
3720   PRINT
3730   PRINT "ALPHA_0";Alpha_0;TAB(41);"BETA_0";Beta_0
3740 Find_modes:    !
3750   Max_histo_val_l=0           ! INITIALIZE MAXIMUM VALUES.
3760   Max_histo_val_u=0
3770   FOR Histo_bin=Alpha_0 TO Breakpoint
3780      IF Histo_val(Histo_bin)>Max_histo_val_l THEN
3790          Max_histo_val_l=Histo_val(Histo_bin)
3800          Lower_mode=Histo_bin
3810      END IF
3820   NEXT Histo_bin
```

```
3830   FOR Histo_bin=Breakpoint TO Beta_0
3840      IF Histo_val(Histo_bin)>Max_histo_val_u THEN
3850          Max_histo_val_u=Histo_val(Histo_bin)
3860          Upper_mode=Histo_bin
3870      END IF
3880   NEXT Histo_bin
3890   PRINT "MAX_HISTO_VAL_L";Max_histo_val_l;TAB(41);"MAX_HISTO_VAL_U";Max_hist
o_val_u
3900   PRINT "LOWER MODE";Lower_mode;TAB(41);"UPPER MODE";Upper_mode
3910   DISP "WORKING...PLEASE WAIT"
3920 Find_histo_sum:    !
3930   Histo_sum=0
3940   FOR Histo_bin=Alpha_0 TO Beta_0
3950      Histo_sum=Histo_sum+Histo_val(Histo_bin)
3960   NEXT Histo_bin
3970 Compute_pdf:  !
3980   IF Model$="G" THEN
3990      Lower_sum=0
4000      Middle_sum=0
4010      Upper_sum=0
4020      Lower_sum_sqrd=0
4030      Upper_sum_sqrd=0
4040      Low_breakpoint=2*Lower_mode-Alpha_0
4050      High_breakpoint=2*Upper_mode-Beta_0
4060      PRINT "LOW_BREAKPOINT";Low_breakpoint;TAB(41);"HIGH_BREAKPOINT";High_br
eakpoint
4070      FOR Histo_bin=Alpha_0 TO Beta_0
4080        IF Histo_bin<=Low_breakpoint THEN
4090           Lower_sum=Lower_sum+Histo_val(Histo_bin)
4100        END IF
4110        IF Histo_bin>Low_breakpoint AND Histo_bin<High_breakpoint THEN
4120           Middle_sum=Middle_sum+Histo_val(Histo_bin)
4130        END IF
4140        IF Histo_bin>=High_breakpoint THEN
4150           Upper_sum=Upper_sum+Histo_val(Histo_bin)
4160        END IF
4170      NEXT Histo_bin
4180      Total_sum=Lower_sum+Middle_sum+Upper_sum
4190      Lower_exp_val=0
4200      Upper_exp_val=0
4210      Lower_variance=0
4220      Upper_variance=0
4230      FOR Histo_bin=Alpha_0 TO Low_breakpoint
4240         Norm_histo_val(Histo_bin)=Histo_val(Histo_bin)/Lower_sum
4250         Lower_exp_val=Lower_exp_val+(Histo_bin*Norm_histo_val(Histo_bin))
4260      NEXT Histo_bin
```

```
4270      FOR Histo_bin=Low_breakpoint+1 TO High_breakpoint-1
4280        Norm_histo_val(Histo_bin)=Histo_val(Histo_bin)/Middle_sum
4290      NEXT Histo_bin
4300      Middle_average=Middle_sum/(High_breakpoint-Low_breakpoint-2)
4310      FOR Histo_bin=High_breakpoint TO Beta_0
4320        Norm_histo_val(Histo_bin)=Histo_val(Histo_bin)/Upper_sum
4330        Upper_exp_val=Upper_exp_val+(Histo_bin*Norm_histo_val(Histo_bin))
4340      NEXT Histo_bin
4350      N_lower=Low_breakpoint-Alpha_0
4360      N_middle=High_breakpoint-Low_breakpoint-1
4370      N_upper=Beta_0-High_breakpoint
4380      FOR Histo_bin=Alpha_0 TO Low_breakpoint
4390        Lower_variance=Lower_variance+(Histo_bin-Lower_mode)^2*Norm_histo_va
l(Histo_bin)
4400      NEXT Histo_bin
4410      FOR Histo_bin=High_breakpoint TO Beta_0
4420        Upper_variance=Upper_variance+(Histo_bin-Upper_mode)^2*Norm_histo_va
l(Histo_bin)
4430      NEXT Histo_bin
4440      PRINT
4450      PRINT "LOWER EXPECTED VALUE";Lower_exp_val,
4460      PRINT "UPPER EXPECTED VALUE";Upper_exp_val
4470      PRINT "LOWER VARIANCE";Lower_variance," ",
4480      PRINT "UPPER VARIANCE";Upper_variance
4490      PRINT "LOWER STANDARD DEVIATION";SQR(Lower_variance),
4500      PRINT "UPPER STANDARD DEVIATION";SQR(Upper_variance)
4510      PRINT "NUMBER OF LOWER ELEMENTS";N_lower
4520      PRINT "NUMBER OF MIDDLE ELEMENTS";N_middle
4530      PRINT "NUMBER OF UPPER ELEMENTS";N_upper
4540      PRINT
4550      PRINT "LOWER SUM";Lower_sum,
4560      PRINT "MIDDLE SUM";Middle_sum,
4570      PRINT "UPPER SUM";Upper_sum
4580      A=Lower_sum/Total_sum
4590      B=Upper_sum/Total_sum
4600      C=Middle_sum/Total_sum
4610      PRINT "A=";A,"C=";C,"B=";B
4620      PRINT
4630      Normalized_sum=0
4640      Start_bin=INT(Lower_mode+1.50*SQR(Lower_variance)+.5)
4650      Stop_bin=INT(Upper_mode-1.00*SQR(Upper_variance)+.5)
4660      Start_bin_flag=0
4670    FOR Histo_bin=Alpha_0 TO Beta_0
4680      Pdfa(Histo_bin)=A/SQR(2*PI)/SQR(Lower_variance)*EXP(-1*(Histo_bin-Lower
_mode)^2/2/Lower_variance)
4690      Pdfb(Histo_bin)=B/SQR(2*PI)/SQR(Upper_variance)*EXP(-1*(Histo_bin-Upper
_mode)^2/2/Upper_variance)
4700      Pdfv(Histo_bin)=0
4710      M=Histo_bin
```

```
4720       IF Histo_bin>=Start_bin AND Histo_bin<=Stop_bin AND (Histo_val(M)/Total
_sum)(Pdfa(M)+Pdfb(M))) THEN
4730         IF Start_bin_flag=0 THEN
4740           New_start_bin=Histo_bin
4750           Start_bin_flag=1
4760         END IF
4770         New_stop_bin=Histo_bin
4780         Pdfv(Histo_bin)=Histo_val(Histo_bin)/Total_sum-Pdfa(Histo_bin)-Pdfb(
Histo_bin)
4790         IF Pdfv(Histo_bin)<0 THEN
4800           PRINT "PDFA";Pdfa(M);"PDFB ";Pdfb(M);"PDFV ";Pdfv(M);"HISTO_VAL"
;Histo_val(M)/Total_sum
4810           Pdfv(Histo_bin)=0
4820         END IF
4830       END IF
4840 !     Pdfa(M)=0 !***************OPTION***************
4850 !     Pdfb(M)=0 !***************OPTION***************
4860 !     Pdfv(M)=0 !***************OPTION***************
4870 !     Pdf(M)=Pdfa(M)+Pdfv(M)+Pdfb(M) !*****OPTION****
4880     NEXT Histo_bin
4890     END IF
4900     GOSUB Curve_fit
4910     FOR Histo_bin=Alpha_0 TO Beta_0
4920       Pdf(Histo_bin)=Pdfa(Histo_bin)+Pdfv(Histo_bin)+Pdfb(Histo_bin)
4930       Normalized_sum=Normalized_sum+Pdf(Histo_bin)
4940     NEXT Histo_bin
4950     FOR Histo_bin=0 TO 255
4960       IF Histo_bin<Alpha_0 OR Histo_bin>Beta_0 THEN
4970         Histo_val(Histo_bin)=0
4980         ELSE
4990           Histo_val(Histo_bin)=Pdf(Histo_bin)*Total_sum
5000       END IF
5010     NEXT Histo_bin
5020     PRINT
5030     DISP "DO YOU WISH TO STORE THE MODELED HISTO (Y/N) ";
5040     BEEP 1500,.1
5050     INPUT Store_pdf$
5060     IF Store_pdf$<>"Y" THEN GOTO 5110
5070     DISP "ENTER HISTO_FILE NUMBER TO STORE VALUES IN  ";
5080     INPUT File_number
5090     GOSUB Create_file
5100     PRINT CHR$(12)
5110     DISP "READY"
5120     BEEP 1500,.1
5130     RETURN
```

```
5140 Curve_fit:  ! THIS SUBROUTINE MODELS THE VALLEY PORTION
5150            !    OF THE HISTOGRAM WITH A POLYNOMIAL FUNCTION.
5160  GOSUB Get_data
5170  GOSUB Set_up_matrix
5180  GOSUB Square_matrix
5190  GOSUB Gauss_jordan
5200  GOSUB Print_results
5210  RETURN
5220 Get_data:    !
5230  PRINT "START BIN ";Start_bin,"STOP BIN";Stop_bin
5240  PRINT "NEW START BIN";New_start_bin;"NEW STOP BIN";New_stop_bin
5250  Num_rows=New_stop_bin-New_start_bin+1
5260  IF Num_rows<10 OR Num_rows>120 THEN
5270      DISP "NUMBER OF DATA POINTS OVER/UNDER RANGE:EXECUTION HALTED"
5280      PRINT CHR$(12)
5290      PRINT TABXY(0,14);"NEW_START_BIN ";New_start_bin,"NEW_STOP_BIN ";New_st
op_bin
5300      PRINT TABXY(0,15);"NUM_ROWS ";Num_rows
5310      BEEP 1500,.2
5320      STOP
5330  END IF
5340  BEEP 1500,.1
5350  DISP "INPUT POLYNOMIAL ORDER ";
5360  INPUT Poly_order
5370  IF Poly_order>5 OR Poly_order<1 THEN
5380      PRINT TABXY(0,15);"ORDER MUST BE 1 THRU 5"
5390      BEEP 1500,.2
5400      GOTO Get_data
5410  END IF
5420  DISP "WORKING...PLEASE WAIT"
5430  Num_columns=Poly_order+1
5440  FOR I=1 TO Num_rows
5450      X_value(I)=New_start_bin-1+I
5460      Y_data(I)=INT(Pdfv(New_start_bin-1+I)*Total_sum)
5470  NEXT I
5480  RETURN
5490 Set_up_matrix:    !
5500  FOR I=1 TO Num_rows
5510      Data_matrix(I,1)=1
5520      FOR J=2 TO Num_columns
5530          Data_matrix(I,J)=Data_matrix(I,J-1)*X_value(I)
5540      NEXT J
5550      Data_vector(I)=Y_data(I)
5560  NEXT I
5570  RETURN
```

```
5580 Square_matrix:   ! THE MATRIX IS SQUARED UP, NOT LITERALLY SQUARED.
5590  FOR K=1 TO Num_columns
5600      FOR L=1 TO K
5610      Coeff(K,L)=0
5620          FOR I=1 TO Num_rows
5630              Coeff(K,L)=Coeff(K,L)+Data_matrix(I,L)*Data_matrix(I,K)
5640              IF (K<>L) THEN Coeff(L,K)=Coeff(K,L)
5650          NEXT I
5660      NEXT L
5670      Const_vector(K)=0
5680      FOR I=1 TO Num_rows
5690          Const_vector(K)=Const_vector(K)+Y_data(I)*Data_matrix(I,K)
5700      NEXT I
5710  NEXT K
5720  RETURN
5730 Gauss_jordan:   !
5740  Error_flag=0
5750  Inv_print_flag=0
5760  Num_const_vctrs=1
5770  FOR I=1 TO Num_columns
5780      FOR J=1 TO Num_columns
5790          B_coeff(I,J)=Coeff(I,J)
5800      NEXT J
5810      Soln_matrix(I,1)=Const_vector(I)
5820      Work_matrix(I,3)=0
5830  NEXT I
5840  D3=1
5850  FOR I=1 TO Num_columns
5860      Biggest_value=0
5870      FOR J=1 TO Num_columns
5880          IF (Work_matrix(J,3)=1) THEN GOTO Continue_1
5890          FOR K=1 TO Num_columns
5900              IF Work_matrix(K,3)>1 THEN GOTO Error_1
5910              IF Work_matrix(K,3)=1 THEN GOTO Continue_2
5920              IF Biggest_value>=ABS(B_coeff(J,K)) THEN GOTO Continue_2
5930              Row_index=J
5940              Column_index=K
5950              Biggest_value=ABS(B_coeff(J,K))
5960 Continue_2: !
5970          NEXT K
5980 Continue_1: !
5990      NEXT J
6000      Work_matrix(Column_index,3)=Work_matrix(Column_index,3)+1
6010      Work_matrix(I,1)=Row_index
6020      Work_matrix(I,2)=Column_index
6030      IF Row_index=Column_index THEN GOTO Divide_pivot
```

```
6040      D3=-1*D3
6050      FOR L=1 TO Num_columns
6060          Hold_value=B_coeff(Row_index,L)
6070          B_coeff(Row_index,L)=B_coeff(Column_index,L)
6080          B_coeff(Column_index,L)=Hold_value
6090      NEXT L
6100      IF Num_const_vctrs<1 THEN GOTO Divide_pivot
6110      FOR L=1 TO Num_const_vctrs
6120          Hold_value=Soln_matrix(Row_index,L)
6130          Soln_matrix(Row_index,L)=Soln_matrix(Column_index,L)
6140          Soln_matrix(Column_index,L)=Hold_value
6150      NEXT L
6160 Divide_pivot:      !
6170      Pivot_index=B_coeff(Column_index,Column_index)
6180      D3=D3*Pivot_index
6190      B_coeff(Column_index,Column_index)=1
6200      FOR L=1 TO Num_columns
6210          B_coeff(Column_index,L)=B_coeff(Column_index,L)/Pivot_index
6220      NEXT L
6230      IF Num_const_vctrs<1 THEN GOTO Reduce_nonpivot
6240      FOR L=1 TO Num_const_vctrs
6250         Soln_matrix(Column_index,L)=Soln_matrix(Column_index,L)/Pivot_index
6260      NEXT L
6270 Reduce_nonpivot:       !
6280      FOR M=1 TO Num_columns
6290          IF M=Column_index THEN GOTO Continue_3
6300          T=B_coeff(M,Column_index)
6310          B_coeff(M,Column_index)=0
6320          FOR L=1 TO Num_columns
6330              B_coeff(M,L)=B_coeff(M,L)-B_coeff(Column_index,L)*T
6340          NEXT L
6350          IF Num_const_vctrs<1 THEN GOTO Continue_3
6360          FOR L=1 TO Num_const_vctrs
6370             Soln_matrix(M,L)=Soln_matrix(M,L)-Soln_matrix(Column_index,L)*T
6380          NEXT L
6390 Continue_3:  !
6400      NEXT M
6410  NEXT I
6420 Interchange_col: !
6430  FOR I=1 TO Num_columns
6440      L=Num_columns-I+1
6450      IF Work_matrix(L,1)=Work_matrix(L,2) THEN GOTO Continue_4
6460      Row_index=Work_matrix(L,1)
6470      Column_index=Work_matrix(L,2)
```

```
6480        FOR K=1 TO Num_columns
6490            Hold_value=B_coeff(K,Row_index)
6500            B_coeff(K,Row_index)=B_coeff(K,Column_index)
6510            B_coeff(K,Column_index)=Hold_value
6520        NEXT K
6530 Continue_4:    !
6540   NEXT I
6550   FOR K=1 TO Num_columns
6560        IF Work_matrix(K,3)<>1 THEN GOTO Error_1
6570   NEXT K
6580   Error_flag=0
6590   FOR I=1 TO Num_columns
6600        Soln_vector(I)=Soln_matrix(I,1)
6610   NEXT I
6620   IF Inv_print_flag=1 THEN GOTO G_j_return
6630   RETURN
6640 Error_1:    !
6650   Error_flag=1
6660   PRINT "ERROR...MATRIX SINGULAR"
6670   BEEP 1500,.3
6680 G_j_return:    !
6690   RETURN
6700 Print_results:    !
6710   Sum_of_y=0
6720   Sum_of_y_sqrd=0
6730   Sum_res_sqrd=0
6740   FOR I=1 TO Num_rows
6750        Y_calc=0
6760        FOR J=1 TO Num_columns
6770            Y_calc=Y_calc+Soln_vector(J)*Data_matrix(I,J)
6780        NEXT J
6790        Residual(I)=INT(Y_calc+.5)-Y_data(I)
6800        Y_calculated(I)=INT(Y_calc+.5)
6810        Sum_res_sqrd=Sum_res_sqrd+Residual(I)*Residual(I)
6820        Sum_of_y=Sum_of_y+Y_data(I)
6830        Sum_of_y_sqrd=Sum_of_y_sqrd+Y_data(I)*Y_data(I)
6840   NEXT I
6850   Corr_coeff=SQR(1-Sum_res_sqrd/(Sum_of_y_sqrd-Sum_of_y*Sum_of_y/Num_rows))
6860   IF Num_rows=Num_columns THEN E5=SQR(Sum_res_sqrd)
6870   IF Num_rows<>Num_columns THEN E5=SQR(Sum_res_sqrd/(Num_rows-Num_columns))
6880   FOR J=1 TO Num_columns
6890        Error_vector(J)=E5*SQR(ABS(B_coeff(J,J)))
6900   NEXT J
6910   PRINT
```

```
6920  PRINT " X           Y         Y_CALCULATED          RESIDUAL"
6930  PRINT
6940  FOR I=1 TO Num_rows
6950      PRINT X_value(I),Y_data(I),TAB(25);Y_calculated(I),TAB(46);Residual(I)
6960      Pcfv(New_start_bin-1+I)=Y_calculated(I)/Total_sum
6970  NEXT I
6980  PRINT
6990  PRINT
7000  PRINT "  COEFFICIENTS       "  ! ERRORS"
7010  PRINT "";Soln_vector(1);TAB(22)   !;Error_vector(1);"     (ZEROETH ORDER TE
RM)"
7020  FOR I=2 TO Num_columns
7030      PRINT Soln_vector(I);TAB(22)   !;Error_vector(I)
7040  NEXT I
7050  PRINT
7060  PRINT "  CORRELATION COEFFICIENT  ";Corr_coeff
7070  RETURN
```

```
7080 Create_file:   ! THIS SUBPROGRAM WRITES THE HISTOGRAM TO THE DISC.
7090   CREATE BDAT "HISTO"&VAL$(File_number),1,512
7100   PRINT CHR$(12)
7110   ASSIGN @Histo_file TO "HISTO"&VAL$(File_number)
7120   OUTPUT @Histo_file;Histo_val(*)
7130   ASSIGN @Histo_file TO *
7140   RETURN
7150 Get_histo_file:     !  THIS SUBROUTINE READS A HISTO FILE FROM DISC.
7160   ASSIGN @Histo_file TO "HISTO"&VAL$(File_number) ! GET HISTOGRAM VALUES
7170   PRINT CHR$(12)
7180   ENTER @Histo_file;Histo_val(*)                  !  FROM THE  FILE ON
7190   ASSIGN @Histo_file TO *                         !  THE DISK.
7200   RETURN
7210 Error_handler:    !  THIS SUBROUTINE ALLOWS RECOVERY FROM THE MOST
7220                   !    COMMON KEYBOARD INPUT ERRORS.
7230   IF ERRN=54 THEN
7240       PRINT TABXY(1,16);"DUPLICATE FILE NUMBER SPECIFIED...TRY AGAIN"
7250       BEEP 1500,.3
7260       DISP "ENTER FILE_NUMBER ";
7270       INPUT File_number
7280   END IF
7290   IF ERRN=56 THEN
7300       PRINT TABXY(1,16);"NON-EXISTANT FILE SPECIFIED...TRY AGAIN"
7310       BEEP 1500,.3
7320       DISP "ENTER FILE NUMBER ";
7330       INPUT File_number
7340   END IF
7350   IF ERRN<>54 AND ERRN<>56 THEN
7360       PRINT TABXY(0,15);"SOFTWARE UNRECOVERABLE ERROR HAS OCCURRED"
7370       PRINT TABXY(0,16);"PROGRAM EXECUTION IS HALTED"
7380       PRINT TABXY(0,17);"ERRN ";ERRN
7390       BEEP 1500,.3
7400       STOP
7410   END IF
7420   RETURN
7430  END
```

APPENDIX B

THE LEAST-SQUARES CURVE FITTING ALGORITHM

The curve fitting algorithm employed to model the histogram valley region is based on the least-squares criterion and uses the Gauss-Jordan method of elimination for solving the resulting simultaneous equations. The theory behind this approach is briefly described in this appendix.

Let the vector $\bar{r}$ contain the elements r_i which are defined as the residuals and can be expressed by

$$r_i = \hat{y}_i - y_i$$

where y_i are the actual y values corresponding to an x of an (x,y) pair and $\hat{y}_i$ are the calculated values. The least-squares criterion requires that the sum of the residuals squared be minimized. For example, assuming a second order polynomial curve-fitting equation of the form

$$\hat{y}_i = A + Bx_i + Cx_i^2 .$$

Then by substitution

$$r_i = A + Bx_i + Cx_i^2 - y_i$$

and the sum of the residuals squared is

$$\sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (A + Bx_i + Cx_i^2 - y_i)^2$$

where n is the number of data points.

To find the values for A, B and C that minimize the previous expression, the derivative with respect to each unknown must be taken and set equal to zero. This can be expressed as

$$\frac{\delta \Sigma r_i^2}{\delta A} = 0, \qquad \frac{\delta \Sigma r_i^2}{\delta B} = 0 \text{ and } \frac{\delta \Sigma r_i^2}{\delta C} = 0 .$$

Substituting the expression for r_i and taking derivatives yields

$$\begin{aligned}\frac{\delta \Sigma r_i^2}{\delta A} &= \frac{2\,\Sigma(A+Bx_i+Cx_i^2-y_i)\delta\,\Sigma(A+Bx_i+Cx_i^2-y_i)}{\delta A}\\ &= 2\,\Sigma(A+Bx_i+Cx_i^2-y_i)\\ &= 0\end{aligned}$$

and

$$\begin{aligned}\frac{\delta \Sigma r_i^2}{\delta B} &= \frac{2\,\Sigma(A+Bx_i+Cx_i^2-y_i)\delta\,\Sigma(A+Bx_i+Cx_i^2-y_i)}{\delta B}\\ &= 2\,\Sigma(A+Bx_i+Cx_i^2-y_i)\,\Sigma(x_i)\\ &= 0\end{aligned}$$

and

$$\begin{aligned}\frac{\delta \Sigma r_i^2}{\delta C} &= \frac{2\,\Sigma(A+Bx_i+Cx_i^2-y_i)\delta\,\Sigma(A+Bx_i+Cx_i^2-y_i)}{\delta C}\\ &= 2\,\Sigma(A+Bx_i+Cx_i^2-y_i)\,\Sigma(x_i^2)\\ &= 0 .\end{aligned}$$

The set of three equations can be expressed as

$$A\sum n + B\sum x_i + C\sum x_i^2 = \sum y_i$$

$$A\sum x_i + B\sum x_i^2 + C\sum x_i^3 = \sum x_i y_i$$

$$A\sum x_i^2 + B\sum x_i^3 + C\sum x_i^4 = \sum x_i^2 y_i .$$

The solution of these three simultaneous equations gives the values for A, B and C which minimize the sum of the residuals squared and, hence, the best least-squares curve fit.

The Gauss-Jordan method of solving simultaneous equations is used for reasons of execution time and expandability. For the second order polynomial the coefficient matrix [K] and constant matrix [G] become

$$[K] = \begin{bmatrix} \sum n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \quad \text{and} \quad [G] = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix}$$

where all summations can be determined directly from the input data. The simultaneous equations can then be written as

$$[K] \begin{bmatrix} A \\ B \\ C \end{bmatrix} = [G] .$$

The Gauss-Jordan method reduces $[K]$ to a unity matrix to yield

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = [S]$$

where the matrix $[S]$ is now the solution matrix.

The methodology is the same and the derivation is similar for other order polynomials than the example shown here. However, care must be exercised because the increased number of multiplies, divides, and adds for high order polynomials contributes to losses in accuracy.

APPENDIX C

COMPUTATION OF THE CORRELATION COEFFICIENT

The correlation coefficient computed by the histogram modeling subroutine provides a measure of the accuracy of the polynomial curve used to model the histogram valley. The correlation coefficient compares the variance of the computed curve values (about the mean of the true curve) to the variance of the true curve (about its own mean). This can be expressed as

$$\rho = \left[\frac{\frac{1}{N}\sum(\hat{y}_i - \bar{y})^2}{\frac{1}{N}\sum(y_i - \bar{y})^2}\right]^{\frac{1}{2}}$$

where ρ = correlation coefficient

y_i = true curve values

$\bar{y}$ = mean value of true curve

$\hat{y}_i$ = calculated curve values and

N = number of data points.

All summations in this appendix are over the interval $i = 1$ to N.

The variance of the true curve can be expressed as

$$\frac{1}{N}\sum(y_i-\bar{y})^2 = \frac{1}{N}\sum(y_i-\hat{y}_i)^2+\frac{1}{N}\sum(\hat{y}_i-\bar{y})^2+\frac{2}{N}\sum(y_i-\hat{y}_i)(\hat{y}_i-\bar{y}).$$

The last summation in the preceding equation is zero because all terms in that sum have factors of the form

$$y - a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

which are zero by virtue of the solution of the simultaneous equations. Substituting the second equation into the first equation yields

$$\rho = \left[\frac{\frac{1}{N}\sum(y_i - \bar{y})^2 - \frac{1}{N}\sum(y_i - \hat{y}_i)^2}{\frac{1}{N}\sum(y_i - \bar{y})^2} \right]^{\frac{1}{2}} .$$

It follows directly that

$$\rho = \left[1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \right]^{\frac{1}{2}} .$$

As the correlation between the true curve and the calculated curve improves, the correlation coefficient approaches the value 1. Very dissimilar curves have a low correlation coefficient.

The histogram modeling subroutine uses this final equation to compute the correlation coefficient.

LIST OF REFERENCES

Castleman, Kenneth R. *Digital Image Processing*. Englewood Cliffs, N.J.: Prentice-Hall, 1979.

Chin, Roland T., and Harlow, William A. "Automated Visual Inspection: A Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-4 (November 1982): 557-562.

Gonzalez, Rafael C., and Wintz, Paul. *Digital Image Processing*. London: Addison-Wesley, 1977.

Kaplan, Gadi. "Industrial Electronics." *IEEE Spectrum* 21 (January 1984): 68-71.

Miller, Alan R. *Basic Programs for Scientists and Engineers*. Berkeley: Sybex, 1981.

Papoulis, Athanasios. *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1965.

Peleg, Shmuel. *Iterative Histogram Modification, 2*. College Park, Maryland: University of Maryland, 1977.

Pratt, William K. *Digital Image Processing*. New York: Wiley and Sons, 1978.

Rosenfeld, Azriel, and Kak, Avinash C. *Digital Picture Processing*, 2 vols. New York: Academic Press, 1982.

Weszka, Joan S. *Threshold Evaluation Techniques*. College Park, Maryland: University of Maryland, 1977.