COMPUTER-AIDED PROCESSING TECHNIQUES FOR USEAGE IN REAL-TIME IMAGE EVALUATION

A Thesis Presented to

the Faculty of the Department of Electrical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

L. K. Bromley

May, 1977

ABSTRACT

The capture and conviction of criminals is often hampered by the present cumbersome method of suspect identification. A witness to a crime may have to view thousands of photographs in order to find the one that corresponds to the suspect. This process can be speeded up by utilizing a computer-aided system.

This thesis provides a computer program that takes as an input a digitized version of the suspect's photograph or sketch. This information is processed by the computer and provides as output, the location of the major features(eyes, nose, mouth, etc.) and also a facial outline. Using this information an algorithm can be developed that can be used to process the photographs contained in a police department's mug file by producing an array containing information that is unique to each photograph. Then when a crime is committed, the digitized version of the suspect's sketch can be given to the computer to construct this array. The array can then be compared to those on file and those most similar can be chosen. Then the witness would only have to view these few photographs.

Before the development of this automatic feature locating algorithm, facial measurements were taken by hand at a rate of 5 minutes per case. Using this algorithm, the major facial features were accurately located within 10 seconds in over 88% of the test cases. The facial outline that was produced by the algorithm was acceptable in 66% of the cases. When implemented in a user interactive system, the algorithm presented in this thesis should significantly speed up the process of suspect identification.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

Chapter 1

INTRODUCTION

Statement of the Problem

One of the major problems of densely populated cities, such as Houston, is crime. In far too many cities, hundreds of large and small crimes, ranging from shoplifting to murder, are committed each day. Some criminals are caught by the police, but many are not caught. The reason is not through any lack of effort by the police but is simply due to the sheer cumbersomeness of the present system of suspect identification. The Houston Police Department has access to thousands of files of known criminals from all over the continent. When a crime has occured, witnesses are often brought in to help identify the perpetrator by giving the police a verbal description. In many cases a police artist is asked to construct a sketch of the suspect from this verbal description. In addition to giving a description or sketch of the suspect, the witness may be asked to review the department's "mug file", which contains photographs of most known criminals. These photographs may be contained in large books or, at best, as slides for projection. In either case, the witness is required to scan hundreds of photographs in hopes of finding the one that corresponds to the suspect. This process can be very time consumming and tedious, causing the witness to give up before scanning all of the photographs. So, unless there are other clues as to the identification of the suspect, a criminal's chances of escaping justice are fairly good.

Problem Solution

In order to resolve this problem, the University of Houston has been given a grant from the Law Enforcement Assistance Administration for development of a computer-aided system that would help in the identification of suspects. An algorithm is to be developed to at least reduce the number of photographs that the witness would have to look through.

Using the algorithm developed under the grant, a computer would digitally scan and process all the photographs currently on file at the police department. This information would be stored for later reference by the computer. Then, when required, the computer could digitally scan a photograph or sketch of a suspect and, using the algorithm, select from the preprocessed images, those photographs most resembling the suspect. The witness would then only have to review these few photographs. In this manner, a criminal's chances of escaping justice will be reduced. And, perhaps, as more criminals are caught, fewer new crimes will be committed.

In response to this grant, the University of Houston has set up an Image Analysis Laboratory. A HP2100A minicomputer with 32k of memory, magnetic tape and dual disk drives has been provided in the laboratory for the development of the algorithm. A microfiche display card system is available for projection of microfiched mug shots that have been provided by the police department. A black and white TV camera and monitor is available for use in the digitazation of photographs. The scanning, digitizing and storing of images is conducted

by an SDS92 computer. In addition to the TV monitor, a storage scope and a plotter are available for displaying the digitized images. A block diagram of the Image Analysis Laboratory equipment and the system procedures for digitizing and displaying images are presented in detail in Chapter 3.

General Description of Thesis

The program developed for this thesis is to be a tool for use in the realization of the final complex algorithm to be used by the police department. The program, henceforth referred to as the CAPTURE (Computer-Aided Processing Techniques for Useage in Real-time image Evaluation) Algorithm, provides two major functions: 1) Locates the major facial features and 2) produces a general outline of the face.

There are a few assumptions that were made in the development of the CAPTURE Algorithm. It is assumed that the photograph to be digitized is of the face and shoulder region only, not full length. It is also assumed that the image is reasonably centered, i.e., that the top of the head is in the upper quarter of the picture and that the chin is in the lower quarter. The algorithm has also been designed to work on Caucasians with no beard or eyeglasses. An extension of the algorithm is possible such that other races or images with beards can be processed. But, such an extension is outside the scope of this thesis.

Uses

The CAPTURE Algorithm is to be used as a tool in the development and implementation of the final complex suspect identification package that will be implemented in the Oakland Police Department. The

technique that is to be used in the final package, is one that uses the relative distances between facial features. For example, the distance between the tip of the nose and the mouth, or the distance from the nose tip to the side of the face would be useful measurements. These and other measurements of the face would be taken and stored for each of the mug shots the police department has on file. Then when an artist has made a sketch from a witness's verbal description, the same type of measurements can be taken from the sketch after it had been digitally scanned. The computer would then compare the measurements from the sketch with those stored in its memory, chosing those similar.

Prior to this work, such measurements had to be taken by hand. But, taking manual measurements of the thousands of photographs contained in the police department's files would be far too time consuming and tedious. There is also a certain amount of measurement inaccuracies due to human error. The program developed for this thesis automatically locates the major features with horizontal or vertical lines. It is then a simple matter to find these measurements by having the computer add or subtract the appropriate line locations.

Figure 1-1 represents a typical CAPTURE Algorithm output. Table 1-1 gives examples of the types of measurements that might be useful, and the type of calculation that would be involved. An array of the resulting distances that were calculated, indexed according to measurement number, would be saved. It would then be this array which is used for comparison in the determination of which photographs are to be viewed.

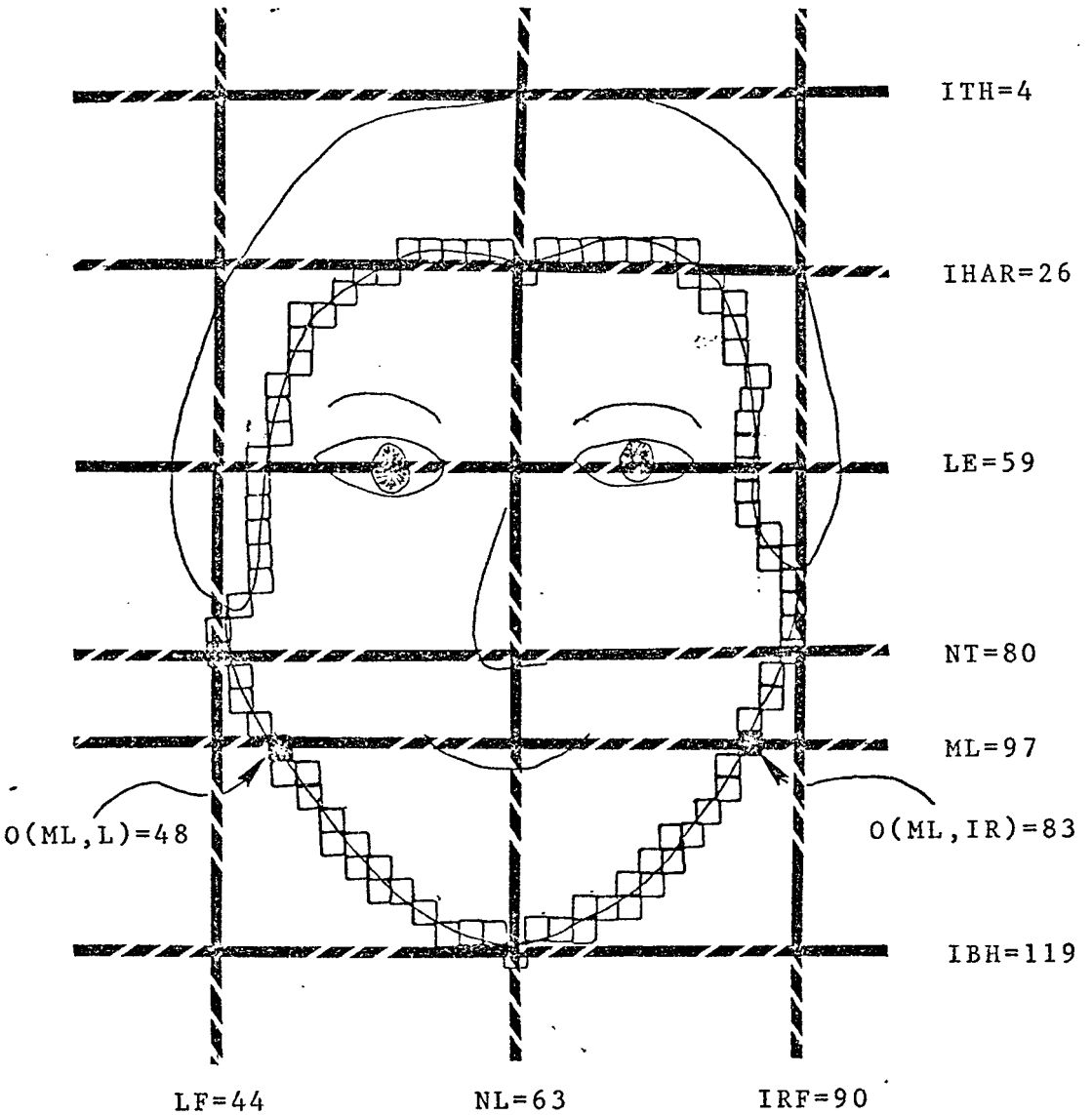


Figure 1-1 Typical Output of CAPTURE Algorithm

Table 1-1

Tabulation of Possible Measurements

Measurement number	Description	Calculation required	Resulting Distance
1	Distance between nose tip and mouth	ML-NT=97-80	17
2	Distance between eyes and nose tip	NT-LE=80-59	21
3	Distance between nose line and right side of face at the mouth line	O(ML,IR)-NL =83-63	20
4	Distance between nose line and left side of face at the mouth line	NL-O(ML,L) =63-48	15
5	Distance between mouth and chin tip	IBH-ML=119-97	22
6	Distance between eyes and mouth	ML-LE=97-59	38
7	Distance between eyes and chin tip	IBH-LE=119-59	60

Another method that could have been used in the final package is based on the use of the Identikit. The Identikit is a collection of various shapes of mouths, chins, noses, and so forth. Using this system, a witness would be asked to construct a sort of sketch of the suspect by choosing the appropriate shapes of the facial features. Each of the shapes in the kit have a corresponding number. The array of numbers of the selected shapes put together by the witness would then be given to the computer. The computer would compare this array to those of the preprocessed images, each having a similar array of Identikit numbers. Those with arrays similar to that provided by the witness would be selected for viewing.

If this method was chosen for the final algorithm, an outline of the face and each of its major features would have to be determined. The facial outline provided by the CAPTURE Algorithm would probably only require that some sort of smoothing be performed. The feature location provided by the CAPTURE Algorithm will give the exact location of each facial feature that needs to be outlined. Therefore, only a program that would provide a smoothed outline for the features would be required.

As can be seen, the use of the automated feature location and facial outline provided by the CAPTURE Algorithm makes it a useful tool for the development of the final complex algorithm. It is flexible enough to be used in many of the various concepts of how the final algorithm is to be implemented.

Chapter 2

LITERATURE REVIEW

A library search was conducted in an attempt to gather information on existing methods of automatic facial identification. It was found that much of the work being done was by qualitative methods. In A. J. Goldstein's paper[1], for example, a method is discussed in which a face feature questionnaire, as given in Figure 2-1, was filled out. Under this system, a witness to a crime would be asked to fill out this questionnaire , rating various facial features from one to five. But, the questionnaire is difficult to fill out on even someone that is well known to the subject, such as a spouse or relative, let alone on a stranger that was perhaps only seen a few seconds. It also relies too much on the individual's concept of what constitutes, for example, a big nose or wide eyes. These methods were therefore discarded.

Other papers found in the literature review described various methods in which certain parts of an anatomical x-ray were being outlined. These papers involved such anatomical regions as the lungs[2,3],

---

[1] A. J. Goldstein, L. D. Harmon, and A. B. Lisk, "Identification of Human Faces", Procedings of the IEEE, Vol. 59, No. 5, (May 1971)

[2] E. L. Hall, R. P. Kruger, and A. F. Turner, "An Optical-Digital System for Automatic Processing of Chest X-Rays", Optical Engineering, Vol. 13, No. 3, (May/June, 1074), pp. 250-257.

[3] E. L. Hall, R. P. Kruger, and A. F. Turner, "Automated Measurements from Chest X-rays for Lung Disease Classification", Computer Graphics, Pattern Recognition and Data Structures Conference, (Beverly Hills, May 1975)

6. Mouth

a. Lip Thickness

UPPER

1	2	3	4	5
Thin	-	Medium	-	Thick

LOWER

1	2	3	4	5
Thin	-	Medium	-	Thick

b. Lip Overlap

1	2	3
Upper	Neither	Lower

c. Width

1	2	3	4	5
Small	-	Medium	-	Large

d. Nose-to-Mouth Distance

1	2	3	4	5
Short	-	Medium	-	Large

e. Mouth-to-Chin Distance

1	2	3	4	5
Short	-	Medium	-	Long

8. Ears

a. Length

1	2	3	4	5
Short	-	Medium	-	Long

b. Lobes

1	2	3	4	5
Attached	-	Medium	-	Not Attached

Figure 2-1 Excerpt from a Face-Feature Questionnaire

heart[1,2] and knee[3]. It was decided that some of these methods could be adapted for use in outlining the face. Some of the methods that were tried but subsequently discarded are described in the following paragraphs.

Weighted Derivative

The first method that was tried was a weighted derivative method simular to the weighted gaussian technique used in D. A. Ausherman's[3] paper. An edge was first assumed to be present at the intersection of the eyeline and the left side of the face, as determined by the feature locating algorithm. Since the search for the edge structure moved upward, the pixel immediately above this assumed point and the two values on either side were addressed. Horizontal and vertical derivatives were calculated for each of these three points using the smoothed derivative method described in Appendix A. These derivatives were then multiplied by a weighting function, as described in Appendix B, that weights the horizontal and vertical derivatives for each point as to which contains the most information. The edge was taken to be at the point of maximum weighted derivative. The entire process was repeated for subsequent rows.

---

[1] E. L. Hall and others, "A Survey of Preprocessing and Feature Extraction Techniques for Radiographic Images", IEEE Transactions on Computers, Vol. C-20, No. 9, (September, 1971), pp. 1032-1044.

[2] R. P. Kruger, "Computer Processing of Radiographic Images", Ph. D. Thesis, (May, 1971), University of Missouri, Columbia Missouri.

[3] D. A. Ausherman, S. J. Dwyer and G. S. Lodwick, "Extraction of Connected Edges for Knee Radiographs", IEEE Transactions on Computers, Vol. C-21, No. 7, (July, 1972), pp. 753-758.

The results of this method was found to be too erratic. It requires a sharply defined edge structure, as were available in the knee radiographs, to work well. The photographs had areas of shadow along the sides of the face that seemed to confuse the algorithm. Once a mistake was made, it was propagated down to subsequent rows with little hope for recovery.

Bit Removal and Bit Plane Methods

These two methods are concerned with looking at one or more of the six binary bits used to indicate the intensity level of each pixel. If a pixel has an intensity level of 57, for example, the corresponding binary word would be 111001.

The bit removal method removes bits one through N from the intensity level word where,

$$1 \leq N < 6$$

For example, if N=4 then the first four bits of the intensity word is to be set to zero while leaving the remaining two untouched. Then for a pixel with intensity 111001 (a decimal 57), the rescaled value would be 000001 or a decimal 1.

This rescaling process is performed for all the pixels in the image. The resulting scaled pixel values are then displayed as an entire picture.

In contrast, the bit plane method masks out all but the desired bit. If, for example, bit plane four is desired, then all the bits in the intensity level word are reset to zero except for bit four which is left untouched. For an intensity level of 111001 (57), the rescaled

level would be zero, since bit four is a zero. In this case the pixel would not be illuminated in the resulting picture. If, however, the level was 110111 (55), bit four is a one, resulting in an illuminated pixel when the picture was drawn.

Of the two, the bit plane method was the more promising. Bit three did give the general outline of the edge structure, including eyes, eyebrows and hair. This method however, also had a great deal of "noise", i.e., spots illuminated where no edge existed on the original photograph. The noise rendered the method unusable for the purpose of feature measurement.

High Pass Filtered Image

This method was employed in hope that it would help enhance the edges that were within the shadows. A high pass digital filter was selected by the process described in Appendix C. A high pass filter was chosen since it has the property of being a differentiator and therefore should enhance edge structure. The value of cutoff frequency was varied in an attempt to achieve the best image.

A cutoff frequency of $f_r = .75\ f_n$, where f_n represents the Nyquist frequency, was found to be the optimum for the image being tested. However, this number varied substantially for each photograph. A search method was attempted in order to determine the proper number for a given photograph. The time, however, to conduct the search was too long to be practical. So this method was also discarded.

Chapter 3

BASIC PROGRAM TECHNIQUES

This chapter will discuss some of the basic techniques used in the development and implementation of the CAPTURE Algorithm. These techniques will include a description of how the photographs are digitized and displayed, and some of the techniques used in the CAPTURE Algorithm programming.

System Procedures

The equipment and software necessary for digitizing and storing photographs was already developed and ready for use when research for this thesis began. Figure 3-1 gives a block diagram of the overall system. The next few paragraphs briefly describe how this system operates.

Digitizing photographs. The digitizing process involves use of the black and white TV camera and monitor as well as the SDS computer. A photograph is displayed either from slides or on the microfich display system. The camera is positioned so that the head of the image fills the camera's viewing range as displayed by the monitor. The image system program is read into the SDS computer which then transfers control to the teletype for command imput. The command to scan the image is typed into the teletype. When the return button is pressed the SDS scans the image which is broken up into a matrix having 128 rows and 128 columns. Each element of the matrix being call a pixel. When the SDS scans

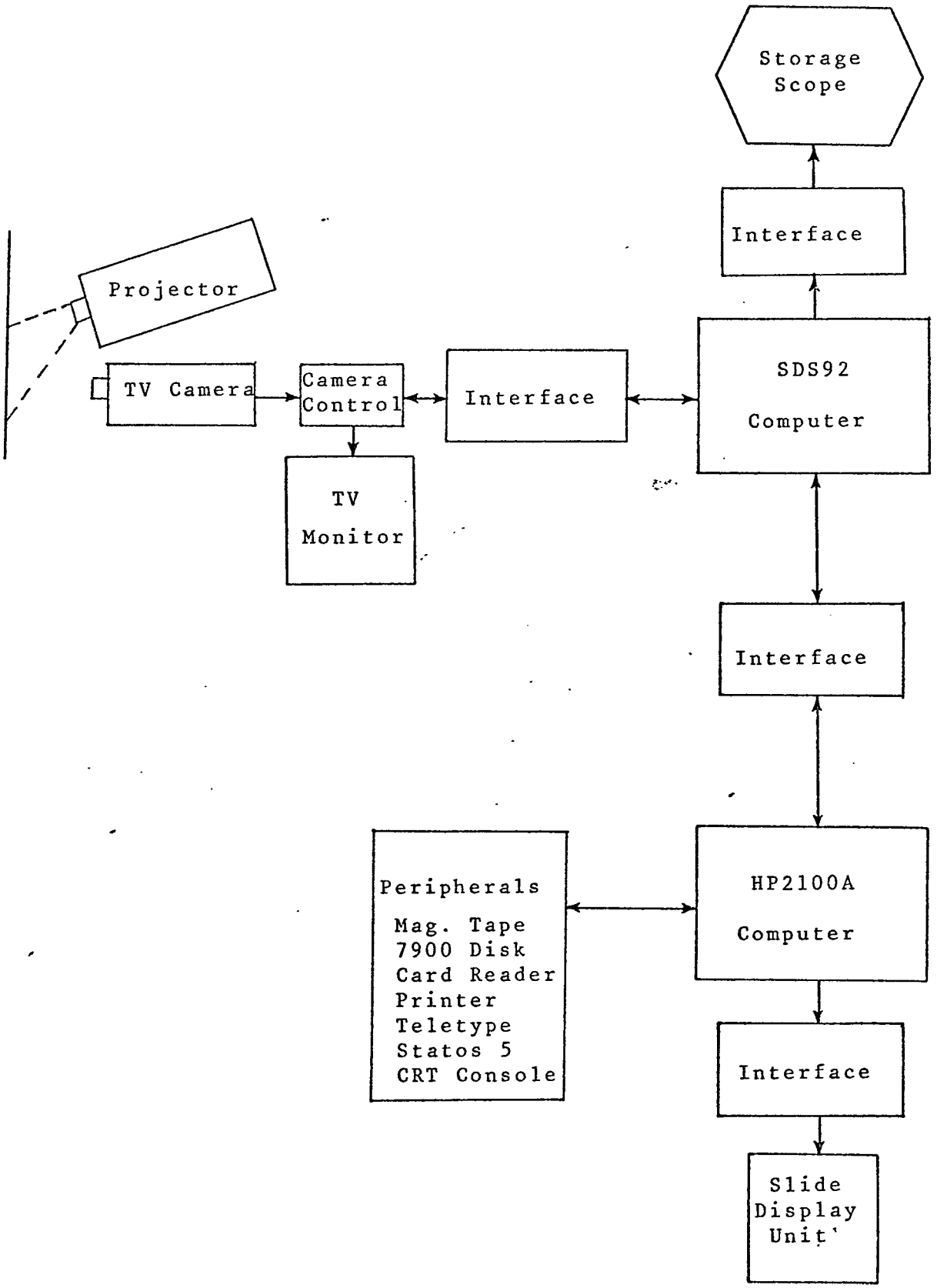


Figure 3-1 Image Processing Laboratory System Block Diagram

an image it assigns a value of 0 to 63 to each pixel according to the intensity level. If the area is very dark, the SDS will assign a pixel value of zero. If very light, a value of 63 will be assigned. After the scanning process is complete, requiring less than a second of time, the digitized version of the image, i.e., the matrix of pixel values, is stored in the SDS.

The digitized image can now be sent to the HP2100A for storage on the disk. First, a 128 by 128 file must be opened on the disk under the desired file name. Then, after the appropriate file name has been entered, a second command is given to the SDS that sends the digitized version of the image to the disk for storage. It can now be readily accessed by the HP2100A minicomputer.

Displaying Images. Images can be displayed in several ways. The actual pixel values can be printed on the CRT console or on the teletype by successive calls to EXEC. (See Chapter D.) This type of display is good for developing or debugging programs that work with these pixel values. This method was used in the early stages of research for the CAPTURE Algorithm to develope the signature technique. (See Chapter 4)

The image can be displayed in visual form on the storage scope, TV monitor or the electrostatic printer. Most of the research for the CAPTURE Algorithm was conducted by displaying the image on the storage scope through successive calls to subroutine SCOPE. Chapter 4 will describe this procedure in detail.

The examples presented in this thesis were produced by the electrostatic printer. The technique used to produce these plots was to first fill a 128 by 128 matrix with the desired information. If the original image is desired to be outputted, the array contains the pixel values. Lines are drawn by filling the appropriate row or column in the array with zeroes. After the array has been filled with the appropriate data a call to Subroutine DRAW will output the array onto the electrostatic printer.

Techniques Developed for CAPTURE Algorithm

The following paragraphs will describe some of the basic concepts used in the CAPTURE Algorithm. The understanding of the concepts of signatures, derivatives and digital filters is vital to the understanding of the algorithm.

Signatures. A signature is obtained by summing columns (column signature) or rows(row signature) of pixel values of a digitized picture. After plotting these sums, the signature can be viewed for the general regions of light or dark. For example, if a row signature is taken over the entire picture, a signature as shown in Figure 3-2 is obtained. The hair tends to bring the signature down while the brighter portions of the face, such as the nose tip and forehead, tend to peak the signature. A vertical line depicting the left side of the face, LF, can be found by determining the leftmost minimum of the signature. Likewise, the right side of the face, IRF, can be found by determining the rightmost minimum of the signature. After IRF and LF are found, a vertical line depicting the nose line can be found by determining the maximum between LF and IRF.

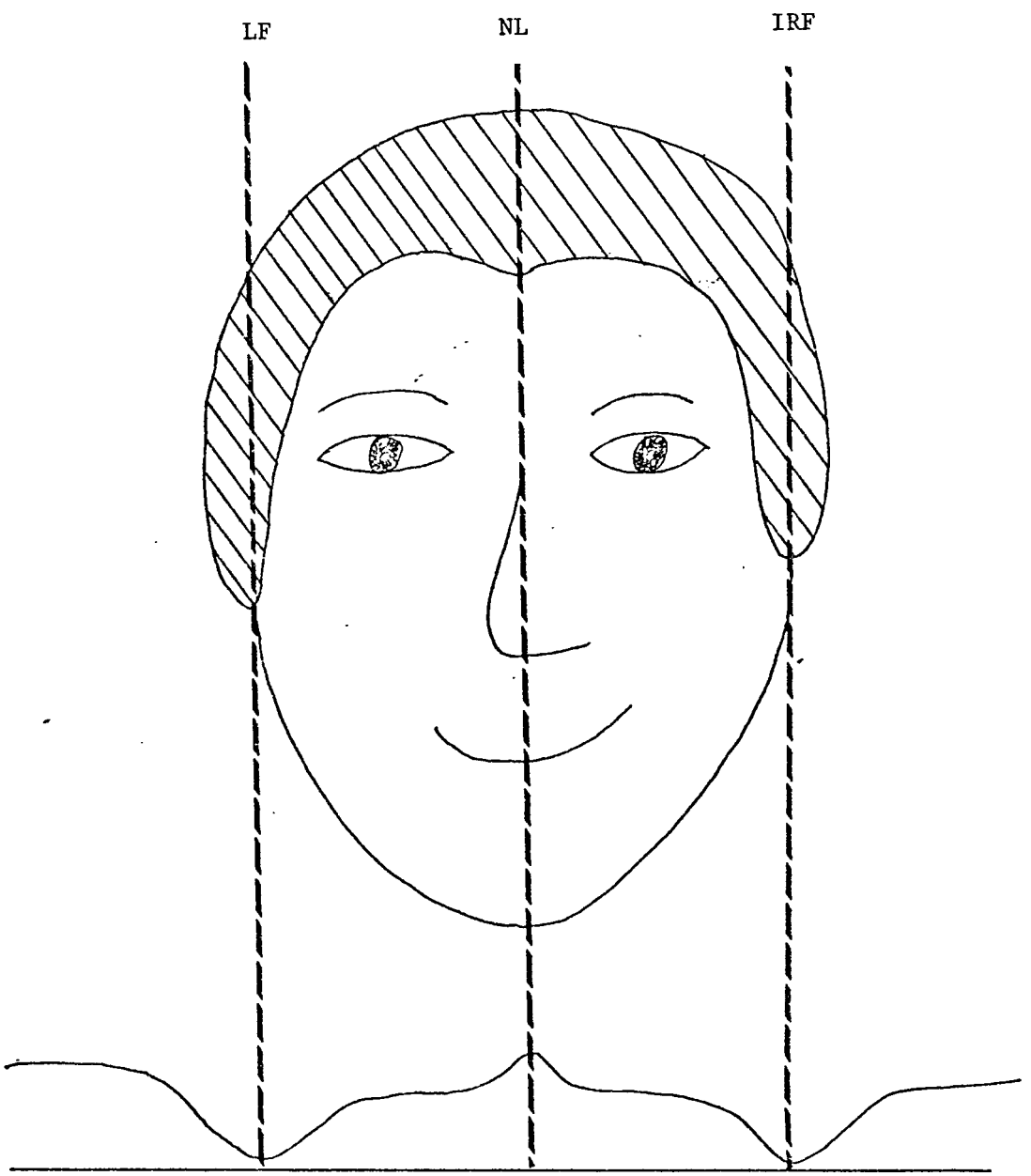


Figure 3-2 Example of a row signature

Derivatives. In some cases, the signature method was not adequate to define certain features of the face. In these cases the derivative of either the signature or the pixel values (from single row or column) was determined. Two types of derivatives were used. The first method was a smoothed derivative, which smoothes the data by passing a least squared parabola through an odd number of points centered around the point of interest and then computing the derivative of this parabola. Evaluating this derivative at zero will provide the derivative for the point of interest. The development of this theory is contined in Appendix A. As shown in this appendix, the first derivative is given by:

$$\dot{y}_k = .0357 \sum_{k=-3}^{3} k \, \overline{y_k}$$

where $\overline{y_k}$ is the pixel value and when using three pixels on either side of the point to be differentiated.

The second method was simply a finite difference derivative. The finite difference at point k is defined by

$$\dot{y}_k = \frac{\overline{y_{k-1}} - \overline{y_{k+1}}}{2\Delta t}$$

where Δt represents the distance between elements.

This derivative can be simplified to

$$\dot{y}_k = \overline{y_{k-1}} - \overline{y_{k+1}}$$

since only the points of maximum and minimum will be of interest and not the value of the derivative.

The direction of the subtraction, that is, whether $\overline{y_{k-1}}$ is subtracted from $\overline{y_{k+1}}$ or whether $\overline{y_{k+1}}$ is subtracted from $\overline{y_{k-1}}$ is

dependent upon the feature being located and the direction of the search. For example, say the intersection of the hairline, IHAR, and the nose line, NL, is to be found. The search starts by taking the finite difference moving down the nose line. $\overline{y_{k-1}}$ will therefore be the point directly above the point of derivative, $\overline{y_k}$, and $\overline{y_{k+1}}$ will be directly below $\overline{y_k}$. As the algorithm progresses down the noseline one would expect to see a drastic change in the derivative at the hairline where the darker hair changes into a lighter forehead. At this point $\overline{y_{k-1}}$ will lie on the dark hair and will have a very low pixel value. $\overline{y_{k+1}}$ will lie on the forehead and have a very large pixel value. Therefore if a positive derivative is desired, the finite difference derivative would be found by

$$\dot{y}_k = \overline{y_{k+1}} - \overline{y_{k-1}}$$

If, however, the search had been from the bottom up, then a change from light to dark is encountered and the positive derivative would be found by

$$\dot{y}_k = \overline{y_{k-1}} - \overline{y_{k+1}}$$

Figure 3-3 shows an example of a finite difference derivative taken down the nose line and searching from the top down.

Digital Filtering. Appendix C describes the basics of digital filtering. Therefore, only the results of the development of the digital filter used in the CAPTURE Algorithm is given here.

A paper published by J. W. Modestino and R. W. Fries[1]

---

[1] J. W. Modestino and R. W. Fries, "Edge Detection in Noisy Images Using Recursive Digital Filtering", unpublished R. P. I. report.

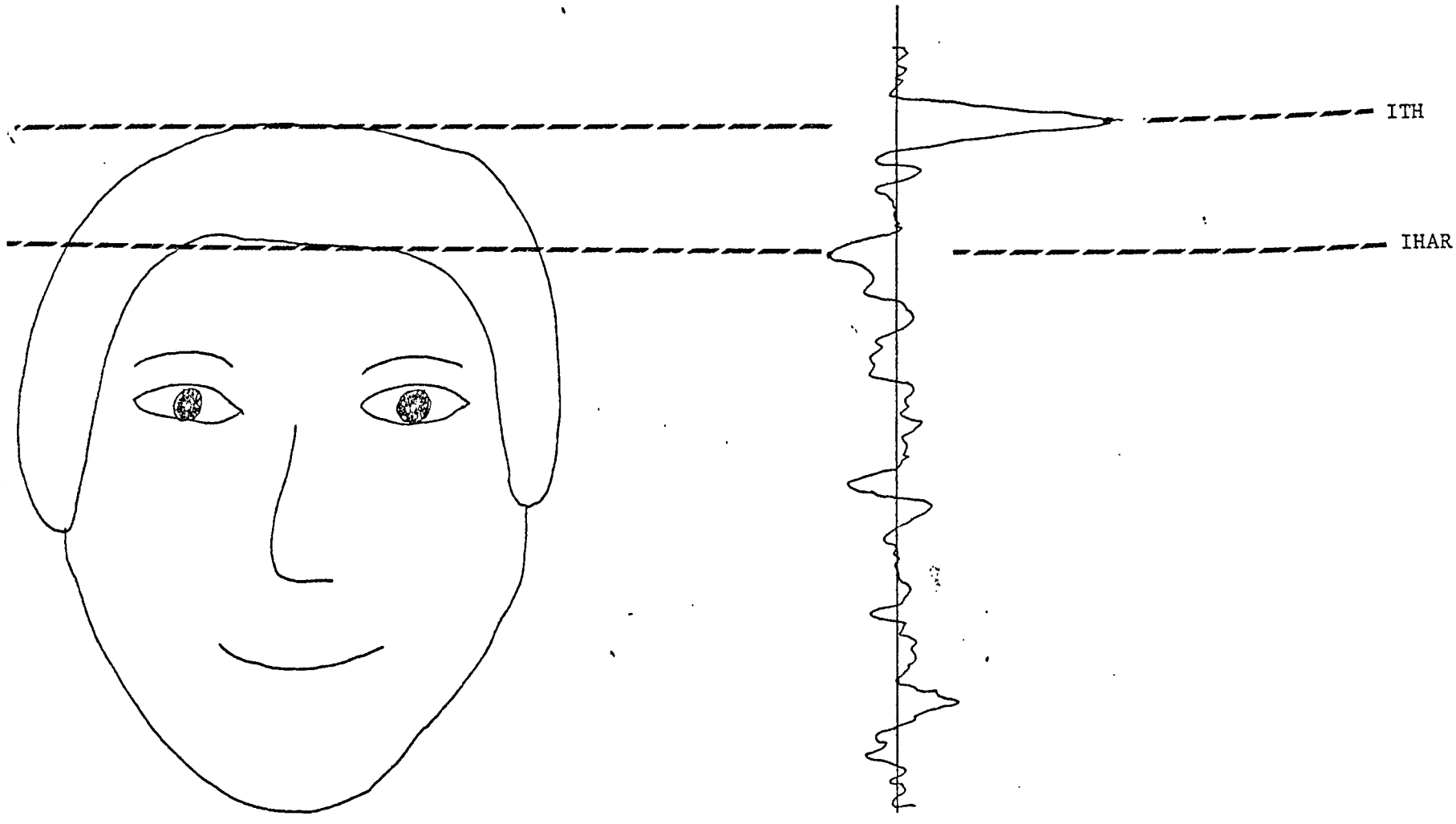


Figure 3-3 Finite difference derivative example taken down noseline

develops an optimum filter that can be used in detecting edges in noisy digitized images. This filter is of the form

$$H_o(r) = \frac{r^2 e^{-r^2/2}}{\frac{\{r^2+2(1-\rho)^2\lambda^2\}\sqrt{r^2+(1-\rho)^2\lambda^2}}{8(1-\rho)\lambda\zeta}} \quad ; r \geq 0$$

where the gray level assumed throughout any elementary rectangle is zero-mean Gaussian with variance σ and correlation coefficient ρ with the gray levels in contiguous rectangles. $\zeta \triangleq \sigma^2/\sigma_n^2$. The parameter ρ represents the degree of correlation across an edge, λ represents the average number of edges per unit distance and ζ represents the SNR of the edge structure.

The digital approximation to $H_o(r)$ was given to be

$$H_o(z_1,z_2) = A\left[\frac{1-\frac{1}{2}(b_{11}+1)\ (z_1^{-1}+z_2^{-1}) + b_{11}z_1^{-1}z_2^{-1}}{1 + a_{10}(z_1^{-1}+z_2^{-1}) + a_{11}z_1^{-1}z_2^{-1}}\right]$$

The gain A and the tree coefficients a_{10}, b_{11}, and a_{11} were determined by using an iterative gradient procedure and the results were given in tabular form. This result is repeated in Table 3-1.

It was determined by trial that the coefficients such that ζ = 3 dB, λ = 0.0125 and ρ = 0.5 worked the best in mose cases. It was expected that a high value of ρ would be required. Since the images that are being processed are of the human face, a high correlation between successive pixel values is likely.

Table 3-1

Summary of Filter Parameters
For Selected Values of λ, ρ and ζ

		$\zeta = \infty$	ζ=10dB			ζ=3dB		
	Filter Coeff.	$-1<\rho<1$	ρ=-0.9	ρ=0.0	ρ=0.5	ρ=-0.9	ρ=0.0	ρ=0.5
λ=0.0125	b_{11}	-0.8256	-0.8924	-0.8252	-0.5736	-0.8264	-0.8256	-0.7556
	a_{10}	-0.2149	-0.2565	-0.3217	-0.4654	-0.3602	-0.3840	-0.4815
	a_{11}	0.0098	-0.2189	-0.1579	0.0449	-0.1540	-0.1461	0.0102
	A	0.1150	0.0530	0.0380	0.0250	0.0220	0.0150	0.0088
λ=0.0250	b_{11}	-0.8256	-0.9396	-0.8460	-0.8228	-0.8664	-0.8120	-0.8260
	a_{10}	-0.2149	-0.1989	-0.2849	-0.3106	-0.3075	-0.3686	-0.3782
	a_{11}	0.0098	-0.2490	-0.1687	-0.1761	-0.2040	-0.1350	-0.1534
	A	0.1150	0.0700	0.0530	0.0400	0.0340	0.0230	0.0150
λ=0.050	b_{11}	-0.8256	-0.9244	-0.7980	-0.8480	-0.8896	-0.8228	-0.7636
	a_{10}	-0.2149	-0.1944	-0.2925	-0.2821	-0.2665	-0.3326	-0.3929
	a_{11}	0.0098	-0.1929	-0.0884	-0.1696	-0.2213	-0.1575	-0.0895
	A	0.1150	0.0840	0.0670	0.0540	0.0480	0.0340	0.0230

Notes: 1. For $\zeta=\infty$ filter coefficients are independent of λ and ρ.

2. Additional results are available from the paper by Modestino and Fries.

This filter has an interpretation as the cascade of the Laplacian operator with a lowpass spatial filter. The Laplacian operator for a two dimensional image is given by

$$\nabla^2 I(X) = \frac{\partial^2 I(X)}{\partial X_1^2} + \frac{\partial^2 I(X)}{\partial X_2^2}$$

where I(X) is the input image.

The application of this differential operator on an image is particularly useful in detecting edges in no-noise conditions. The images that will be experienced, however, contain random noise due to the displaying system and also due to shadows on the face. But, by cascading this operator with a low pass filter, much of this noise can be removed. Notice that although a lowpass filter may blurr the edge structure, as shown in Appendix C, the Laplacian is a second order derivative and so will lessen the effects of the blurring.

Chapter 4

CAPTURE ALGORITHM DESCRIPTION

This chapter will describe the manner in which the CAPTURE Algorithm locates the major features and produces the facial outline. A detailed description of the computer program routines used to implement the algorithm is contained in Appendix D for reference.

Feature Location

The facial features that are located by the CAPTURE algorithm are as follows:

1. Top of head--horizontal line
2. Tip of chin--horizontal line
3. Eyes--horizontal line
4. Mouth--horizontal line
5. Nose tip--horizontal line
6. Left side of face--vertical line
7. Nose line--vertical line
8. Right side of face--vertical line

These features are located by either horizontal or vertical lines as depicted in Figure 4-1.

The algorithm begins by determining the three vertical lines indicating the left and right sides of the face and the nose line. These lines can be found by first forming a row signature of 10 lines near the center of the image. Figure 4-2 is an example of this signature.

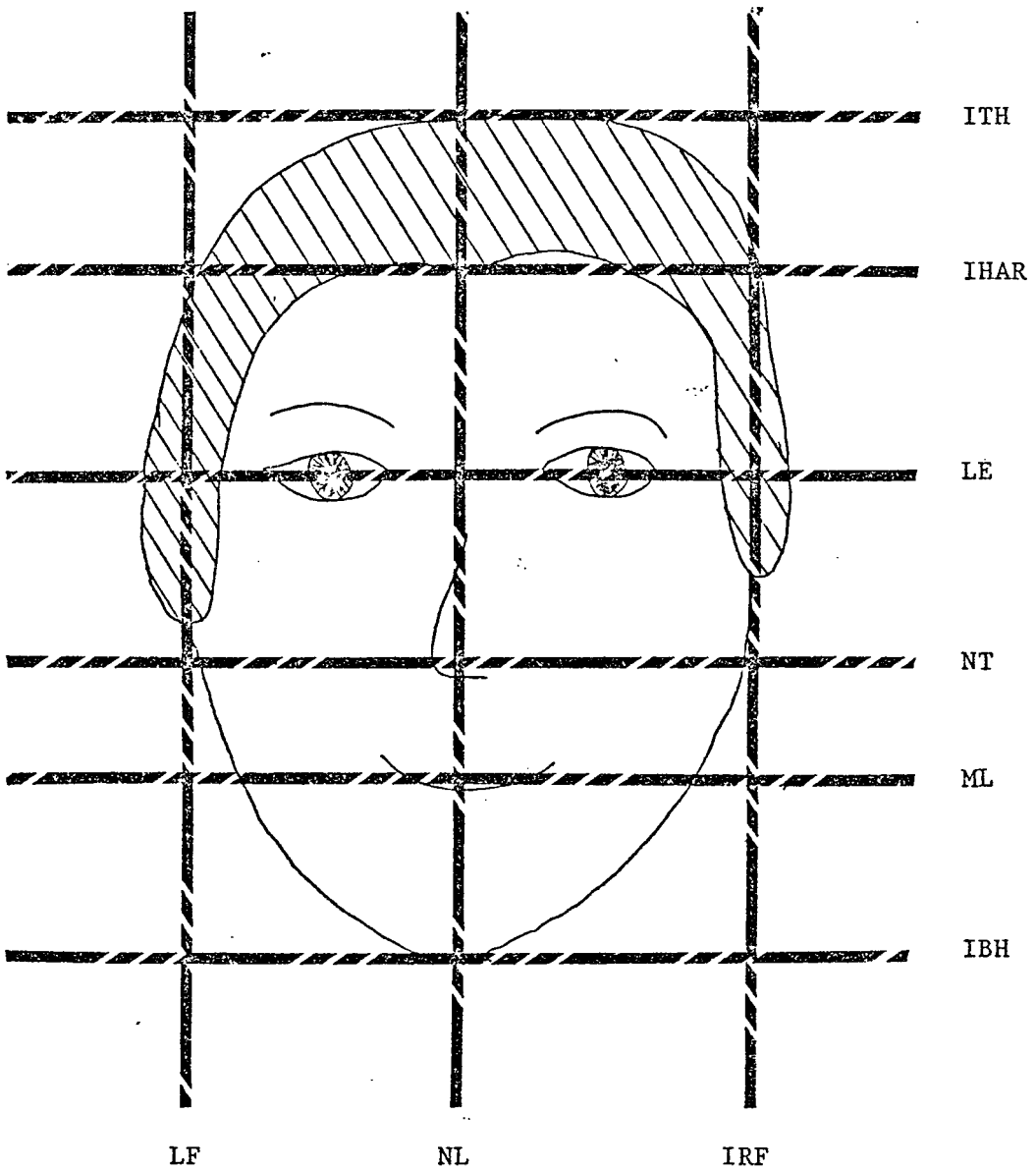


Figure 4-1 Feature Location

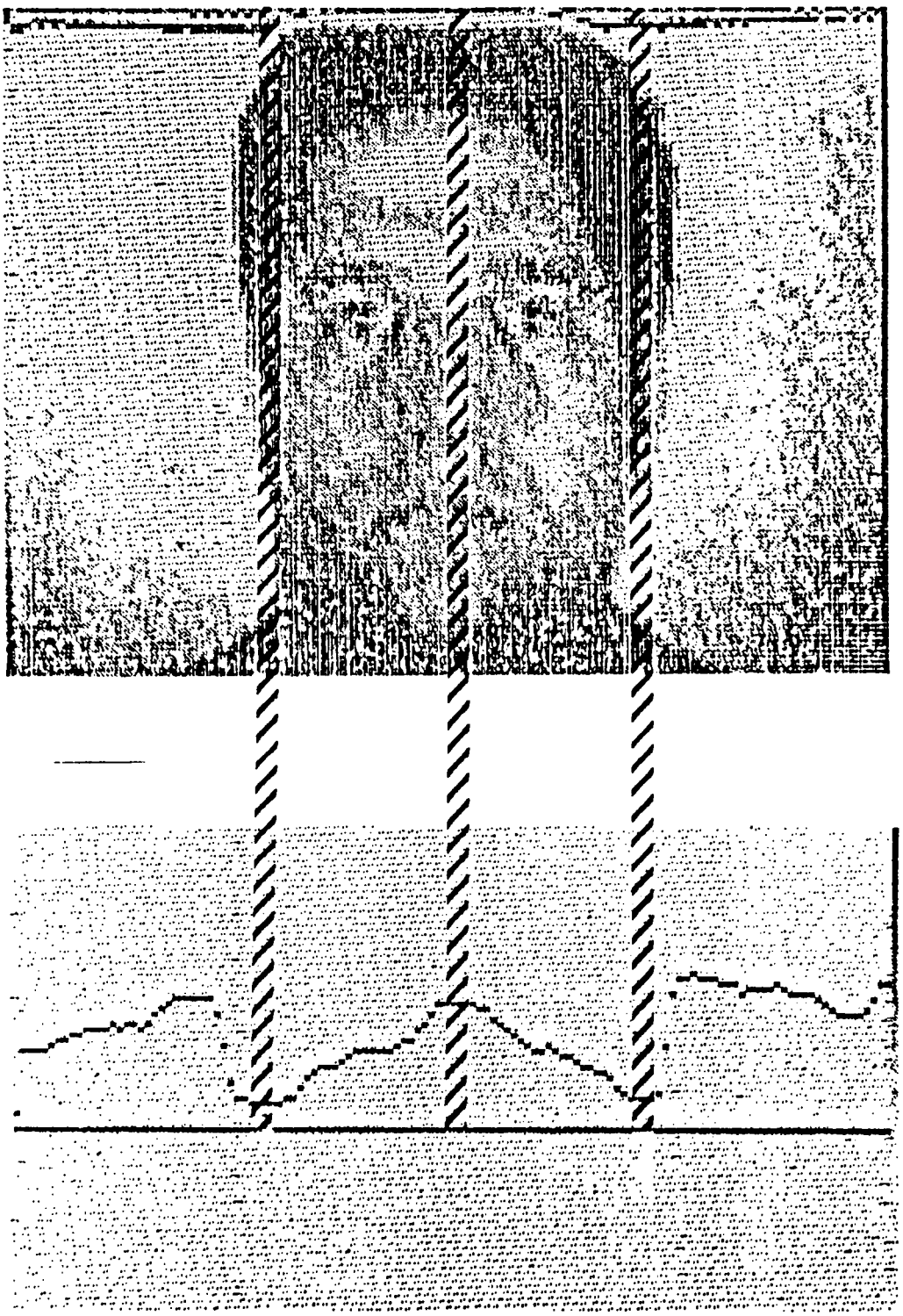


Figure 4-2 Row Signature Example

The estimate for the left side of the face is obtained by finding the minimum of the first half of the signature. The estimate for the right side of the face is likewise found by determining the minimum of the last half of the signature. Using these estimates for LF and IRF, the nose line, NL, can be found by determining the maximum of the signature between LF and IRF.

The next task, then, is to find the top of the head, ITH. This is accomplished by taking a finite difference derivative along the nose line, NL. The top of the head should then be at the point of the maximum negative derivative when searching the top quarter of the derivative array. The derivative should be negative since at the top of the head point we will have a low pixel value, due to the dark hair, minus a large pixel value due to the light background. Figure 4-3 shows an example of the derivative taken down the nose line.

The hairline, IHAR, can similarly be found by determining the maximum positive derivative in the top half of the image working from ITH down. We would expect the derivative to be positive since at the point of the hairline we will have a high pixel value due to the forehead, minus a lower pixel value due to the hair.

Estimates of the bottom of the head, IBH, and the mouth line, ML, are then formed. In general most heads are no more than three times as long as they are wide. Therefore, this criterion is used to form an estimate for the bottom of the head. Since this value is used only as a limit for searches, an exact determination is not necessary at this point. An estimate for the mouth line is formed by assuming it to be 30 rows above the IBH line. This value will be determined exactly at a later time.

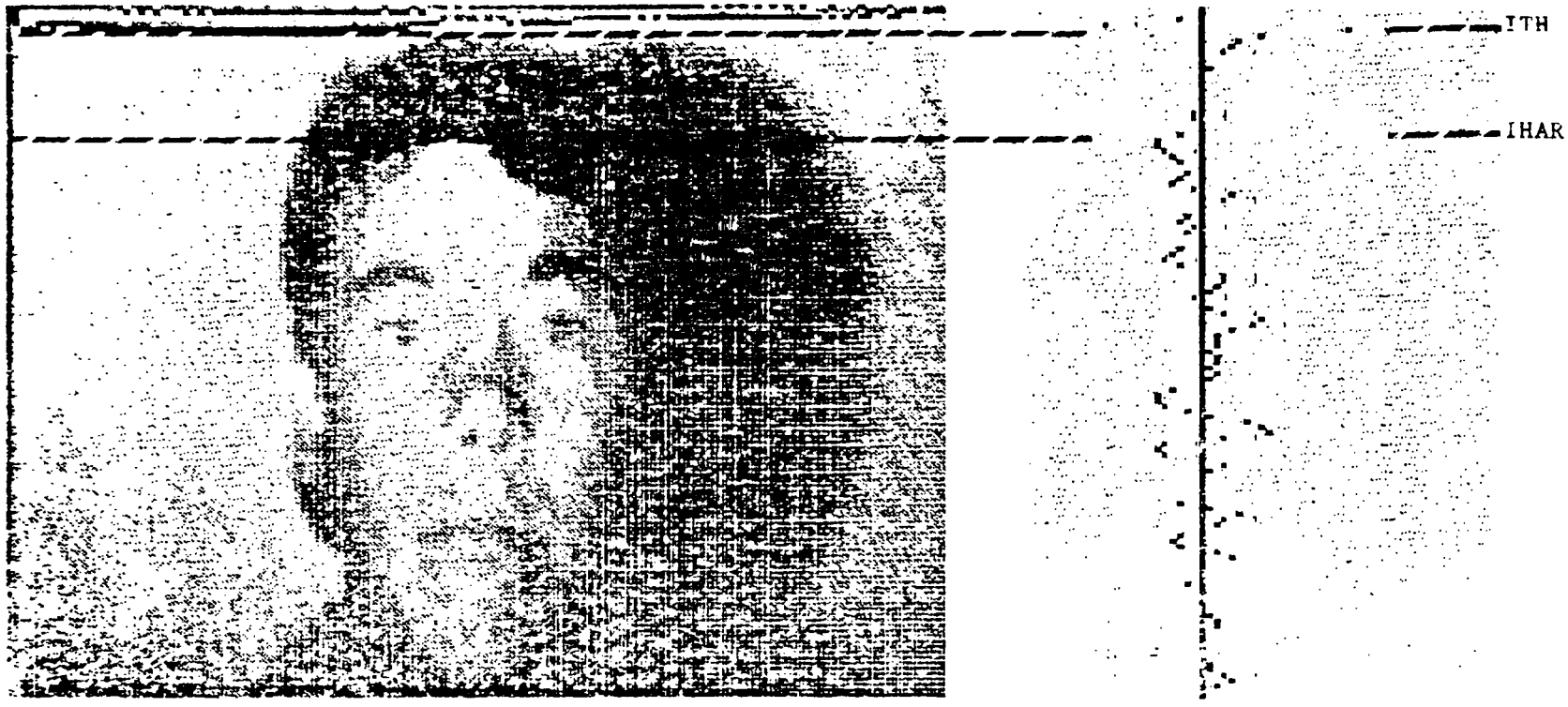


Figure 4-3 Finite Difference Derivative along Noseline

Next, the eyebrow, LEB, and the eyeline, LE, can be found. This is accomplished by first forming a column signature between columns starting at five columns to the right of the nose line and ending at five columns to the left of the right of the face. The eyebrow, containing many low value pixels, should then be the minimum value of this signature.

The eyeline, LE, is next desired to be found. Using the same signature as determined for finding the eyebrow, and working from LEB down, we would expect to find next a peak corresponding to the space between the eyebrow and the eye, and then a minimum corresponding to the eye itself. So these are the next conditions for which the algorithm searches.

The tip of the nose, NT, can now be determined. Since the nose protrudes out from the face, it is usually very bright in the images. Therefore, by forming a column sum just around the nose line, NL, and by determining the maximum, the nose tip can be found. So, to this end, a column signature is formed over the five rows on either side of the nose line. The maximum is searched for between rows starting at 15 rows below the eyeline, and ending 15 rows before the mouthline. This maximum is taken to be NT.

The next task is to define the mouth line ML, more accurately. The mouth line can be found using the same signature used to determine the nose tip. This signature is searched between rows beginning at 10 rows above the old estimate for the mouth line and ending at 20 rows from the bottom of the head. Since the mouth will contain many low pixel values, the mouth line, ML, is taken to be the minimum of the search.

The algorithm determines the right and left extremities of the mouth. These values will not be made available as a program output since these values are too variable. These values would widely vary, for example, in a picture if the suspect was smiling than if he were not smiling. These values will be used however, in the search strategy used to determine the facial outline. These values are determined by first forming a row signature involving ten rows centered around ML. The corners of the mouth, being recessed into the face more than at the center, should contain more low pixel values. Therefore the left corner of the mouth, MLL, can be found by searching for the minimum of the row signature between the center of the left part of the face to the nose line. The right corner of the mouth, MLR, is similarly found by searching for the minimum signature between the nose line and the center of the right side of the face. At this point all the major features have been located. The remaining task, then is to produce the facial outline.

Facial Outline

An example of the facial outline that is to be determined is shown in Figure 4-4. The feature location lines as previously determined are used as an aid in the determination of the facial outline. The outline is formed by first passing the rows of pixel values through a digital filter to remove some of the effects of the facial shadows. The particular filter that is used is described in Chapter 3. After the image is passed through the digital filter, the filtered pixels are compared to the threshold. All pixels having a filtered value less than the threshold are set to a value of one, while pixels over the threshold are set to zero.

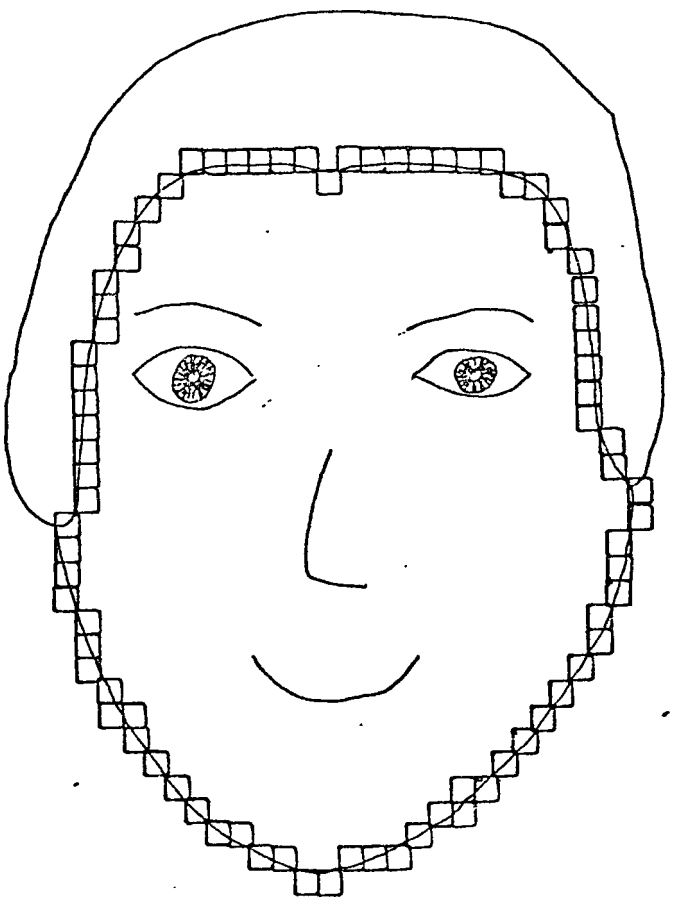


Figure 4-4 Facial Outline

The image matrix at this point contains only pixels having a value of ones or zeros. They are arranged, however, so that most of the pixel values in the face area are now zero and those in the hair and along the chin line are ones. Figure 4-5 is an example of an output from the filter. The process is then to search from the center of the face outward and select the first column whose pixel value is a one to be the face edge. However, some of the darker areas in the face, such as the eyes and mouth, may also contain a few scattered ones that may interfere with the edge detection scheme. But, these areas have already been located by the feature location algorithm. This information can be used to skip over these areas. Therefore the areas that are searched are the six regions as given in Figure 4-6.

A search is begun over regions 1 and 2 from a point begining at five rows above the eyebrow line, up to the hairline. Starting at the left side of the face, a search is conducted up the column until a row is found that contains a value of 1. When a 1 is found, it is replaced with a value of 64 to indicate that it was the first value of 1 encountered. The next column is then addressed and the process is repeated. After searching regions 1 and 2 upwards in this manner, the regions are searched in a horizontal direction. Searching each region in two directions will provide a smoother appearing outline. Region 1 is first searched starting at the nose line and moving left to the left side of the face. Again, when the first value other than a zero is encountered, which could be a value of 1 or a value of 64 from the previous search, the pixel value is reset to be a value of 64. In a similar manner region 2 is searched starting at the nose line and moving right.

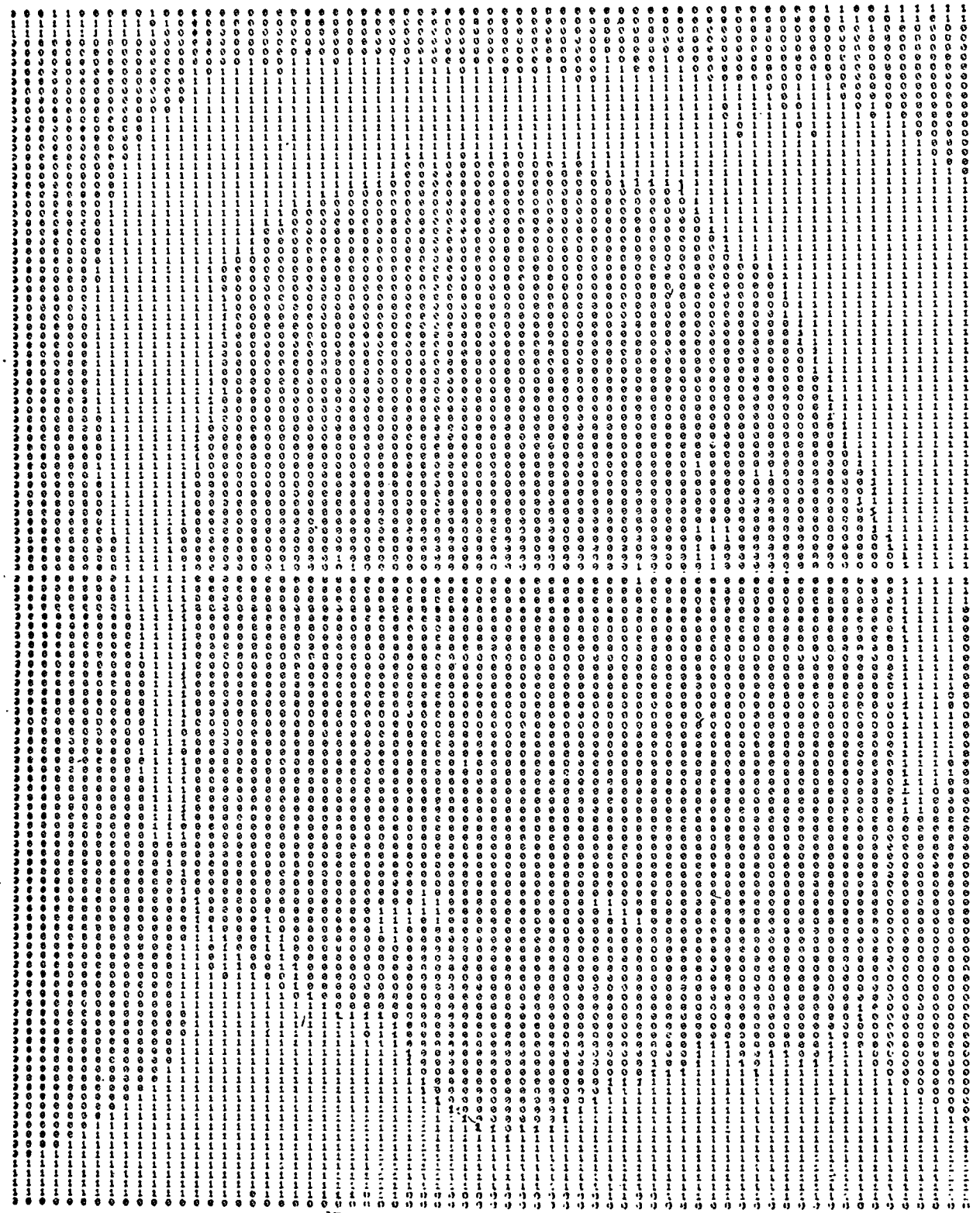


Figure 4-5 Filter Output Example

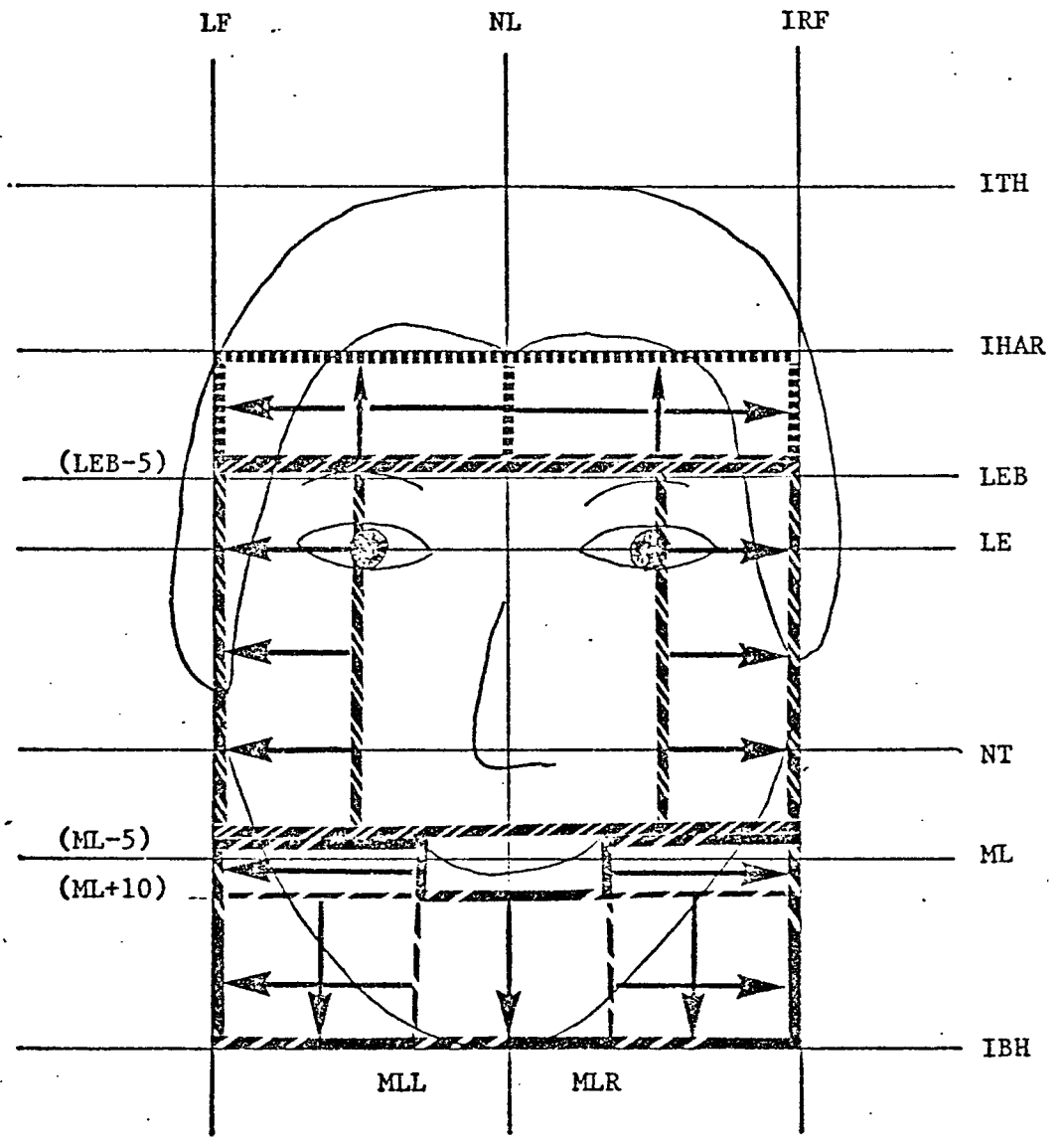


Figure 4-6 Facial outline search regions

After the edges have been defined for regions 1 and 2, regions 3 and 4 are searched. Since these two regions are at the sides of the head, they need only to be searched in the horizontal directions. Region 3 is searched starting from a point midway between the NLF and NNL points and moving left until the left side of the face is reached. In some cases the search may not encounter a pixel of value 1 on a given row. In this case there would be a gap in the facial outline. To prevent this, a running account is kept to indicate the column at which the edge was detected in the previous row. If no edge is detected in a subsequent row, this value is taken to be the edge. Region 4 is searched in a similar manner starting at a point midway between the NLF and IRF points and moving right to the right side of the face.

After all edges have been detected in regions 3 and 4, regions 5 and 6 are searched. Region 5 and 6 are first searched starting at a point ten lines below the mouth and moving down to the bottom of the head. Region 5 is then searched starting at the left edge of the mouth, as determined by the feature locating algorithm and moving to the left side of the face. Region 6 is similarly searched from the right edge of the mouth to the right side of the face. In both regions 5 and 6 a running account is kept to determine the edge column for the previous row in case no edge is found.

After all six regions been searched, the complete facial outline has been formed. This information, including the feature locations can then be output to the user for any further processing that may be required.

Chapter 5

EXAMPLES OF CAPTURE ALGORITHM RESULTS

This chapter will present some examples of the CAPTURE Algorithm results. These results will cover examples of not only when the algorithm worked well but also examples of when it did not work. In these latter examples, reasons for the algorithm's failure will be given and possible corrections for future use will be proposed.

The images in these examples were taken from a magnetic tape that was already available. Although the tape contained 44 images, only the first 16 were accessible. As will be seen in the examples, information is slowly lost on the left side of the images while information on the next image begins overlapping into the right. There probably exists some incongruity between the program that wrote the images onto the magnetic tape and that which reads from the magnetic tape onto disk.

CAPTURE Algorithm Output Time

Before looking at the examples, a few comments are needed on the time required to process a picture using the CAPTURE Algorithm. All the images take approximately the same time to process. The only time that may vary slightly is that required for the determination of the facial outline.

In general, the CAPTURE Algorithm requires about 18 seconds to read the image from the disk, process the information, locate the features and determine the outline. It then requires another 12 seconds just to write this information back onto the disk in order that the

electrostatic printer may be used as an output device. As can be seen most of the time is required reading from or writing onto the disk. But all the information is available after 12 seconds. The total time required to output this information to the user is dependent upon the output device.

CAPTURE Algorithm Examples

Figures 5-1 through 5-8 are examples of CAPTURE Algorithm results. The feature location and facial outline of each of the images were rated under a four point system:

GOOD-All information is determined to be accurate

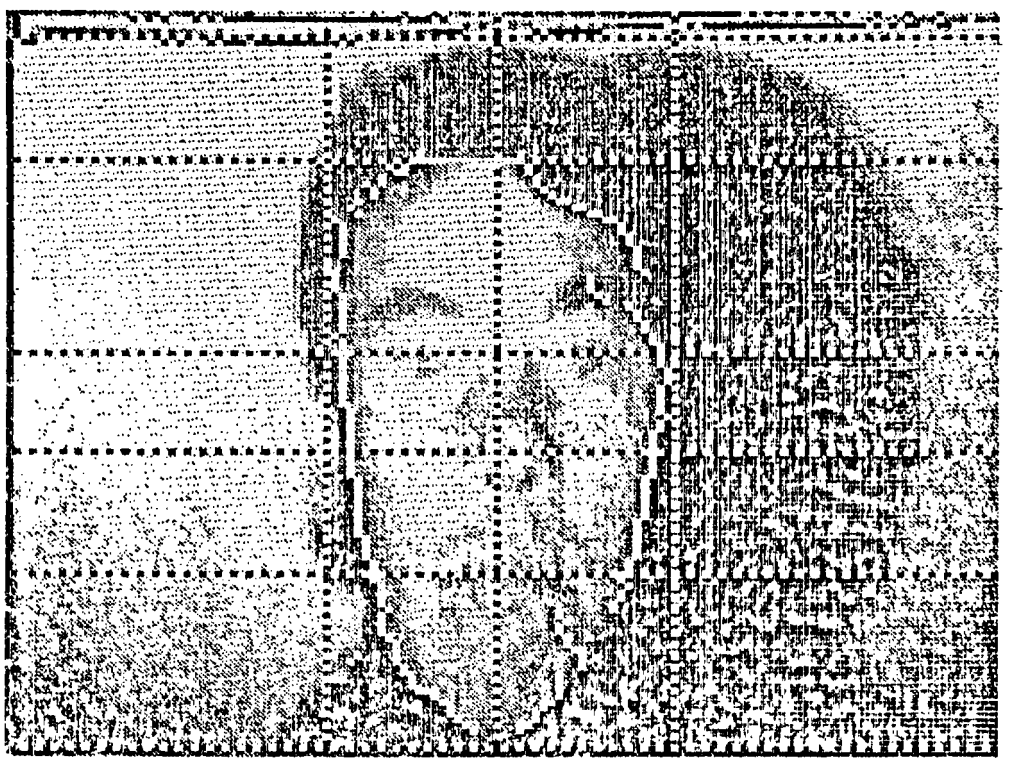
FAIR-Most of the information is accurate

POOR-Some of the information is useable

BAD -No useable information

The feature location technique used by the CAPTURE Algorithm seemed to work very well. Indeed all the images except for Figure 5-8 were given a GOOD rating. The line that was to depict the eyeline, LE, in Figure 5-8 is too high. The algorithm mistook the hair on the right side of the face which falls down nearly to his eyes, to be his right eyebrow. The algorithm then looks for the next peak, which is supposed to be the space between the eyebrow and the eye, and then assumes the next minimum to be the eye. In this case however, it was his eyebrow. If this type of case becomes a problem in the future, the eyeline could be verified by the computer to be accurate by searching upwards from the nost tip line, NT, after it has been determined.

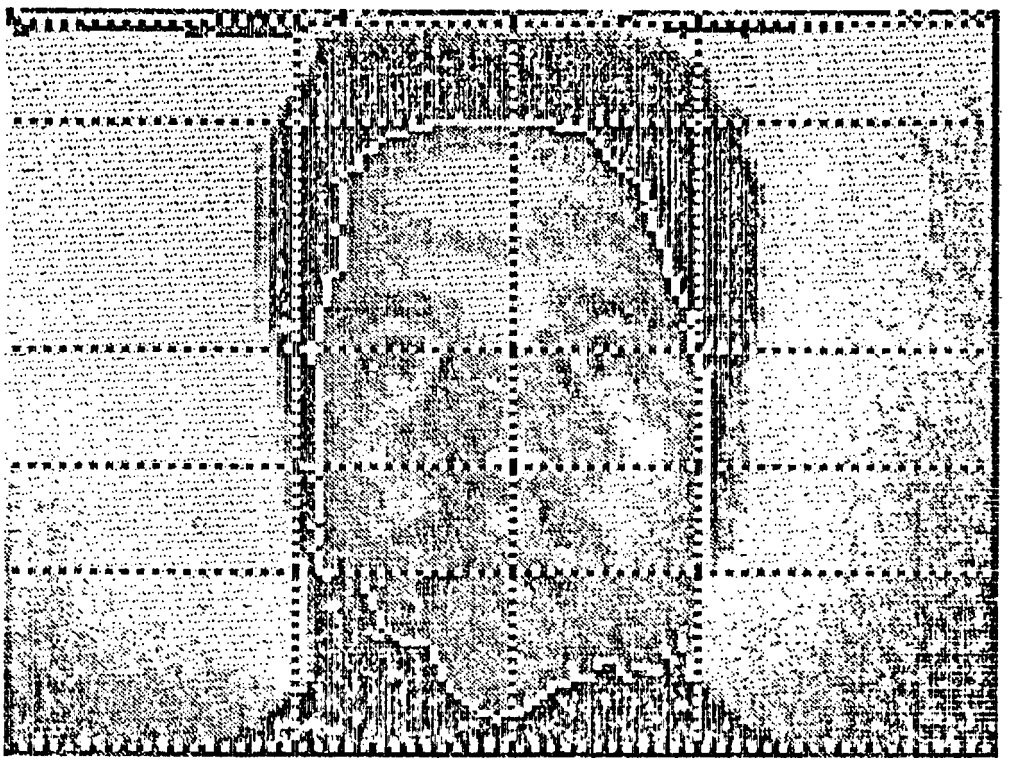
The facial outline algorithm however, did not function as well. Figures 5-1, 5-2, 5-3, and 5-4 were given a GOOD rating since all the information can be used if a smoothing technique is used to soften the



THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

HORIZONTAL LINES		VERTICAL LINES	
TOP OF HEAD=	5	LEFT SIDE OF FACE=	41
HAIRLINE=	26	NOSELINE=	63
EYELINE=	59	RIGHT SIDE OF FACE=	86
NOSETIP=	76		
MOUTHLINE=	97		
BOTTOM OF HEAD=	127		

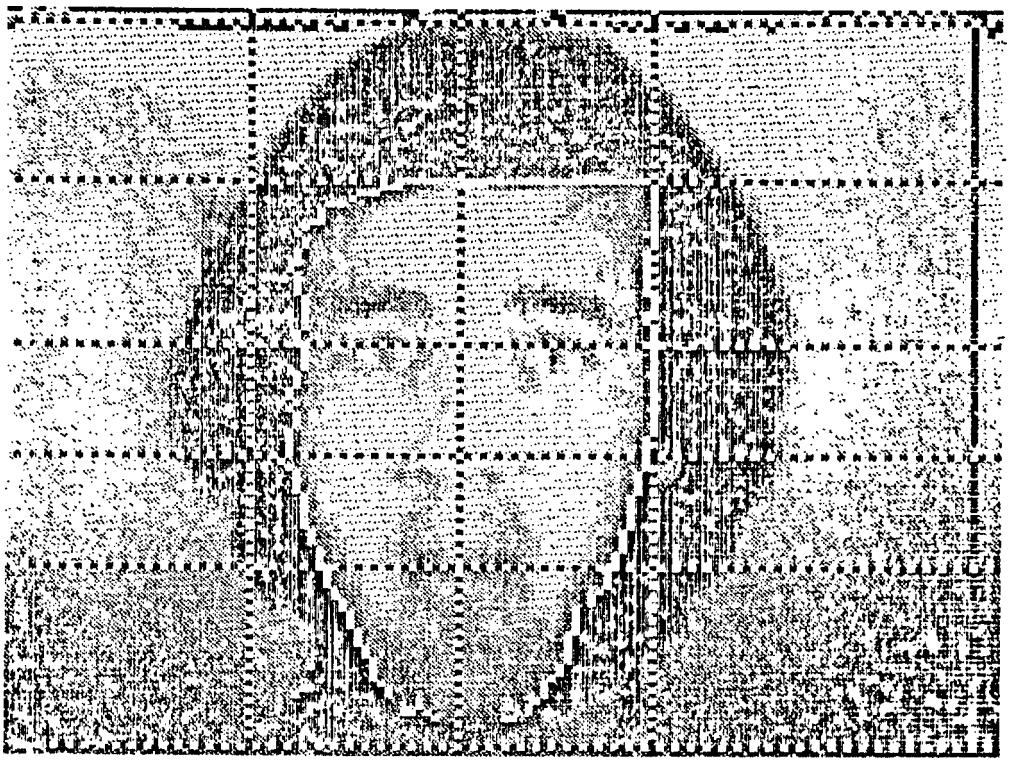
Figure 5-1 CAPTURE Algorithm example



THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

HORIZONTAL LINES		VERTICAL LINES	
TOP OF HEAD=	3	LEFT SIDE OF FACE=	37
HAIRLINE=	20	NOSELINE=	65
EYELINE=	59	RIGHT SIDE OF FACE=	89
NOSETIP=	79		
MOUTHLINE=	97		
BOTTOM OF HEAD=	127		

Figure 5-2 CAPTURE Algorithm Example



THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

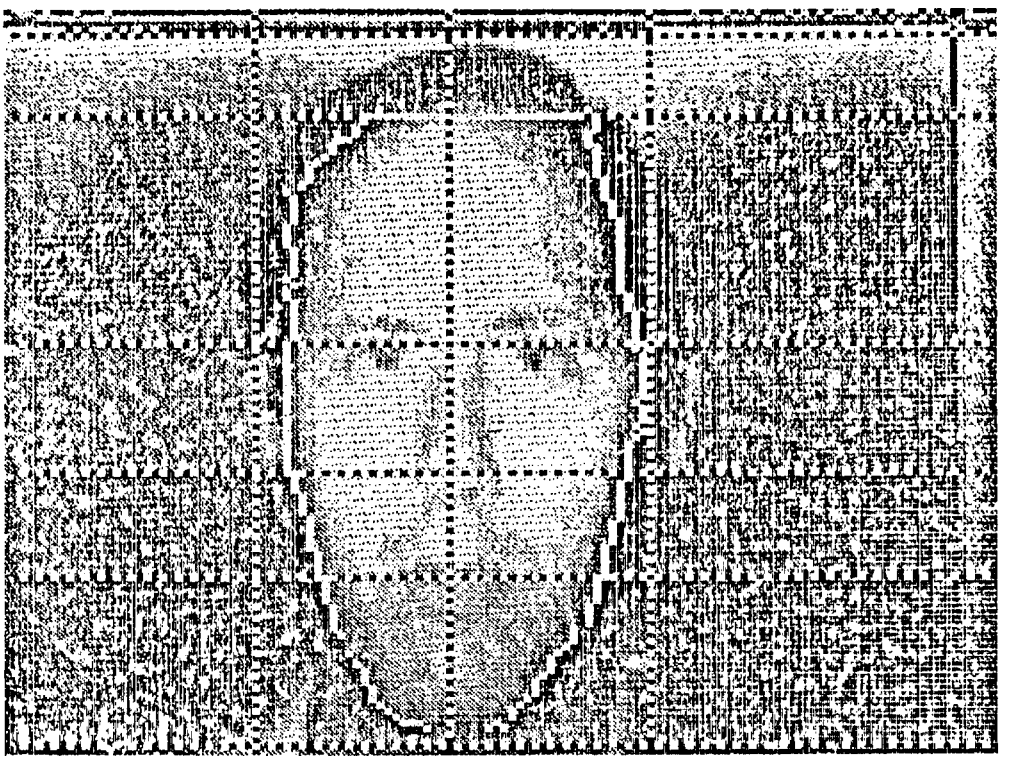
HORIZONTAL LINES
****

TOP OF HEAD= 3
HAIRLINE= 30
EYELINE= 58
NOSETIP= 77
MOUTHLINE= 96
BOTTOM OF HEAD= 127

VERTICAL LINES
****

LEFT SIDE OF FACE= 31
NOSELINE= 58
RIGHT SIDE OF FACE= 83

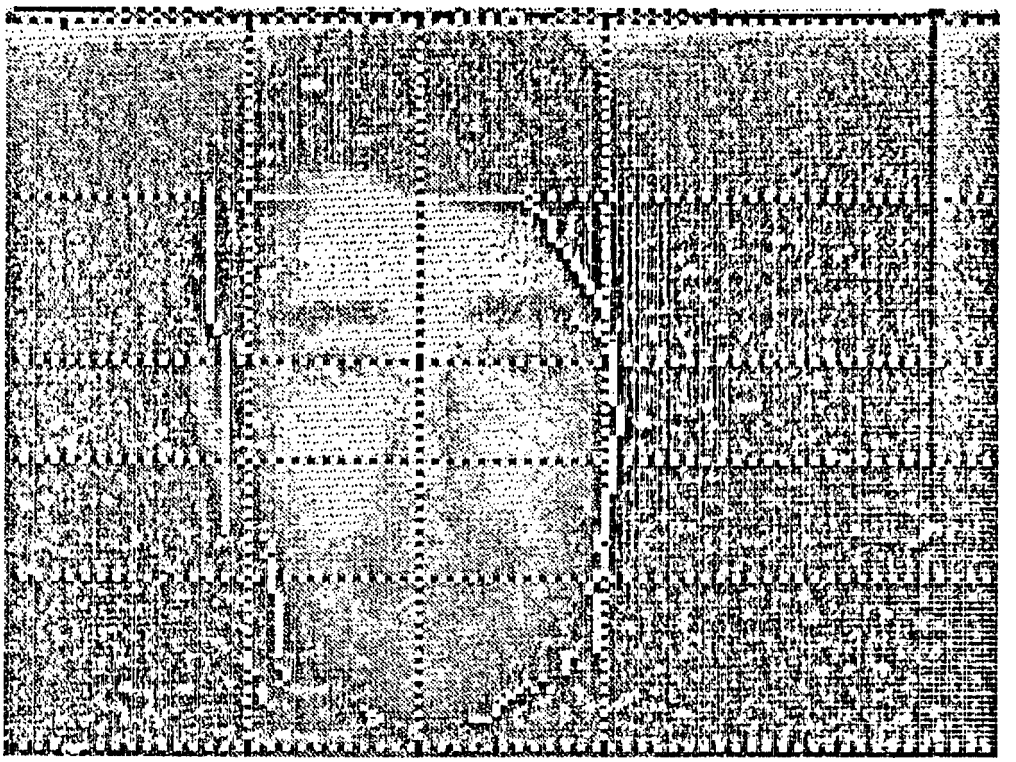
Figure 5-3 CAPTURE Algorithm Example



THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

HORIZONTAL LINES	VERTICAL LINES
**********	**********
TOP OF HEAD= 5	LEFT SIDE OF FACE= 32
HAIRLINE= 19	NOSELINE= 57
EYELINE= 58	RIGHT SIDE OF FACE= 83
NOSETIP= 80	
MOUTHLINE= 98	
BOTTOM OF HEAD= 127	

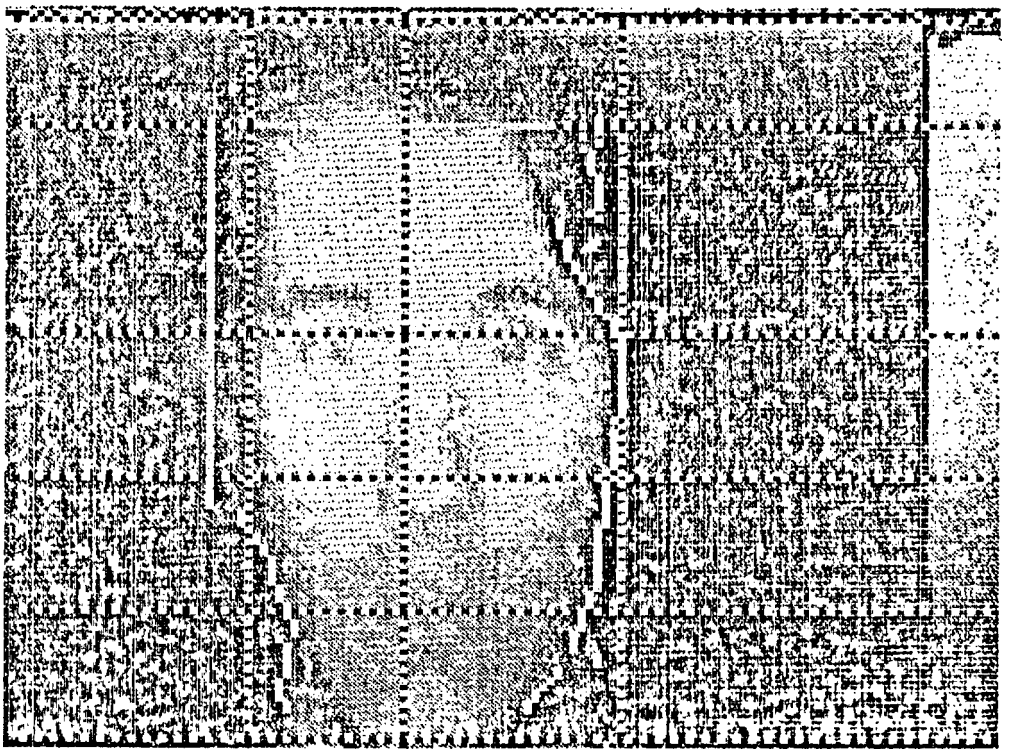
Figure 5-4 CAPTURE Algorithm Example



THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

HORIZONTAL LINES	VERTICAL LINES
TOP OF HEAD= 3	LEFT SIDE OF FACE= 31
HAIRLINE= 33	NOSELINE= 53
EYELINE= 61	RIGHT SIDE OF FACE= 77
NOSETIP= 78	
MOUTHLINE= 98	
BOTTOM OF HEAD= 127	

Figure 5-5 CAPTURE Algorithm Example



THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

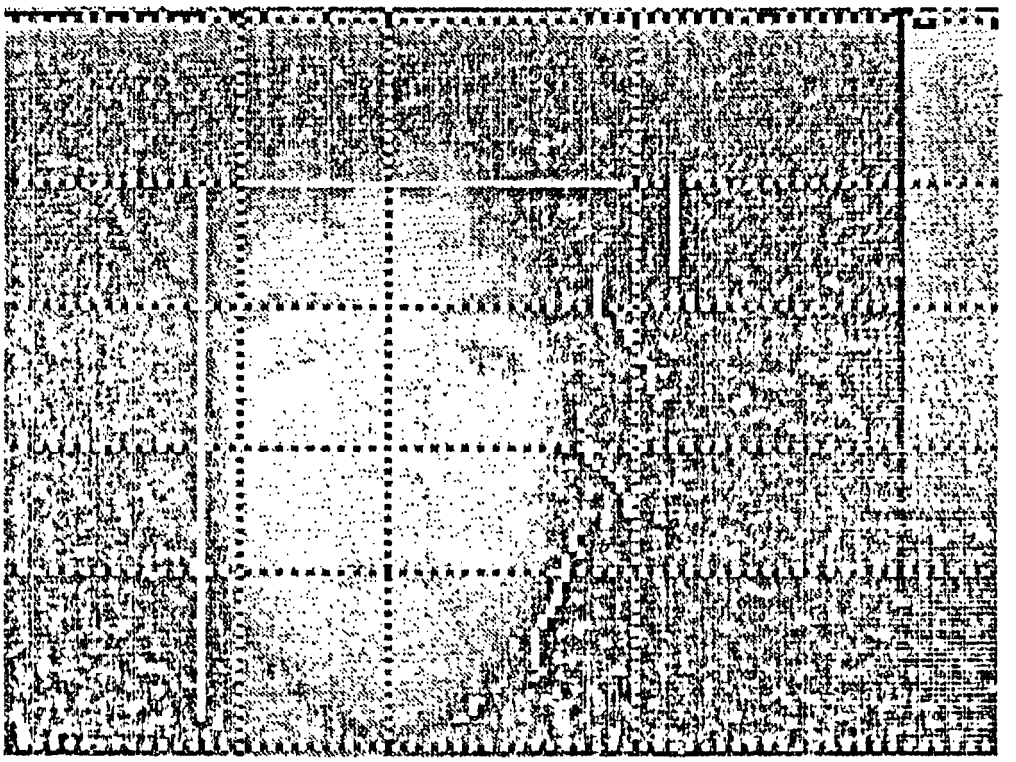
HORIZONTAL LINES		VERTICAL LINES	
TOP OF HEAD=	3	LEFT SIDE OF FACE=	31
HAIRLINE=	21	NOSELINE=	51
EYELINE=	57	RIGHT SIDE OF FACE=	79
NOSETIP=	82		
MOUTHLINE=	105		
BOTTOM OF HEAD=	127		

Figure 5-6 CAPTURE Algorithm Example

THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

HORIZONTAL LINES	VERTICAL LINES
**********	**********
TOP OF HEAD= 2	LEFT SIDE OF FACE= 30
HAIRLINE= 34	NOSELINE= 54
EYELINE= 60	RIGHT SIDE OF FACE= 80
NOSETIP= 82	
MOUTHLINE= 96	
BOTTOM OF HEAD= 127	

Figure 5-7 CAPTURE Algorithm Example



THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
DETERMINED FOR FILE ???

HORIZONTAL LINES	VERTICAL LINES
**********	**********
TOP OF HEAD= 3	LEFT SIDE OF FACE= 30
HAIRLINE= 31	NOSELINE= 49
EYELINE= 52	RIGHT SIDE OF FACE= 81
NOSETIP= 76	
MOUTHLINE= 97	
BOTTOM OF HEAD= 127	

Figure 5-8 CAPTURE Algorithm Example

abrupt changes. Figure 5-7 was given a rating of FAIR since only the outline in the lower right quadrant is inadequate. Figures 5-5 and 5-6 were given ratings of POOR. Although a smoothing technique will greatly enhance the outline's appearance, there is a good deal of information lost particularly in the upper half. It is highly probable that the standard threshold of .11 that was used on all the images was too low. Increasing this threshold will help the facial outline. Perhaps some algorithm can be developed to automatically determine the optimum value of threshold for each particular image, based on its average gray scale value. Notice that these figures are somewhat blurry with few distinct edge structures. Figure 5-8 was given a rating of BAD since there is very little information contained in the facial outline.

Table 5-1 gives a summary of all the images that were accessible on the tape. As given in the table, 56% of the images on the tape qualified (i.e., contained images having no eyeglasses or beards, etc) for use. Of these images, the feature location program received ratings of GOOD in 88.9% and FAIR in 11.1%. If useable information is considered to be ratings of GOOD or FAIR, then the feature location algorithm provided useable information in 100% of the cases. The facial outline algorithm, using a standard threshold of .11 for all images, provided useable information in 66.7% of the cases.

Table 5-1

CAPTURE Algorithm Rating Summary

File	Qualifies?	Feature Location	Outline	Figure
1	Yes	Good	Good	5-1
2	Yes	Good	Good	5-2
3	Yes	Good	Fair	-
4	No	-	-	-
5	Yes	Good	Good	5-3
6	No	-	-	-
7	Yes	Good	Good	5-4
8	No	-	-	-
9	No	-	-	-
10	Yes	Good	Poor	5-5
11	Yes	Good	Poor	5-6
12	Yes	Good	Fair	5-7
13	No	-	-	-
14	Yes	Fair	Bad	5-8
15	No	-	-	-
16	No	-	-	-
Totals	9/16=56%	8/9=88.9% GOOD 1/9=11.1% FAIR	4/9=44.5% GOOD 2/9=22.2% FAIR 2/9=22.2% POOR 1/9=11.1% BAD	

Chapter 6

CONCLUSIONS AND RECOMMENDATIONS

As can be seen from the examples in Chapter 5, the CAPTURE Algorithm can provide accurate feature location and facial outline data. After only 18 seconds this information is available for the user to employ as necessary. Previously, feature measurements were taken by hand, requiring an average of 5 minutes per case for maximum accuracy. The savings in time provided by the CAPTURE Algorithm are substantial.

Future users of the CAPTURE Algorithm may want to investigate further possible techniques for improvement of the algorithm. For example, some attention may be devoted to pre-filtering the image before the features are located. This may remove some of the shadows that could possibly confuse the algorithm Other filters may be tried for the facial outline, or, an automatic search routine could be developed that would find the optimum threshold for each image.

One area for future work on the CAPTURE Algorithm that is strongly recommended, is the adaptation for stereographic images. A companion program should be developed, using similar techniques as in CAPTURE to provide a facial outline and measurement data for a facial profile. In this manner, information can be provided for both the full face, as is now available, and for the profile resulting in a three dimensional description of the suspect. This development is important because a witness may only see the profile of the suspect and not the full face.

There are also many possible options for the actual employment of this algorithm. It may be that an interactive system should be employed. Under this option an operator will decide upon the accuracy of the feature locations and determine whether the facial outline is smooth enough for usage. When, for example, the computer completes its calculations a question could be asked of the operator whether all the information is accurate. If not, the operator can tell the computer what part is not accurate and suggest a possible new direction in which the computer should search. If a result as shown in Figure 5-8 were produced, having the eyeline being located too high, an operator could tell the computer that line LE was inaccurate and to search immediately below this line for the proper location. In this manner both the computer and the human operator will be providing what each does best: rapid and accurate computations as only a computer can provide and human judgement as only a human can provide.

Even in this type of interactive system a significant amount of time savings can be attained. For example, say there are 10,000 mug shots on file that are to be processed. If, at worst case, the computer can only process 70% of these images, leaving 30% to be measured by hand the total time can be calculated:

Number of images by computer = (10,000) (.7) = 7000

Number of images by hand = (10,000) (.3) = 3000

Time for computer = (7000) (.3) = 2100 minutes

Time for hand = (3000) (5.) = 15000 minutes

Total time = 17,100 minutes = 285 hours

Assuming an eight hour work day, this time corresponds to 35.625 man-days. If, however, all 10,000 photographs had to be measured by hand:

$$\text{Time by hand} = (10{,}000)\ (5) = 50{,}000 \text{ minutes}$$

$$= 833.333 \text{ hours}$$

Again assuming an eight hour work day, this time corresponds to 104.16 man-days. This represents a total savings of 68.54 man-days when aided by the CAPTURE Algorithm.

BIBLIOGRAPHY

Ausherman, D. A., Samuel J. Dwyer and Gwilym S. Lodwick, "Extraction of Connected Edges from Knee Radiographs", IEEE Transaction on Computers, Vol C-21, No. 7, July 1972, pp. 753-758.

El'bur, R. E., "Utilization of the Apparatus of Projective Geometry In the Process of the Identification of Individuals by their Photographs", pp. 321-348.

Fries, R. W., "Edge detection in Noisy Images Using Recursive Digital Filtering", M. S. Thesis, Department of Electrical and Systems Engineering, Rensselaer Polytechnic Institution, Troy, N. Y., November 1975.

Goldstein, A. J., L. D. Harmon, and A. B. Lisk, "Identification of Human Faces", Proceedings of the IEEE, Vol. 59, No. 5, May 1971 pp. 748-760.

Hall, Earnest L., Richard P. Kruger, and A. F. Turner,"An Optical-Digital System for Automatic Processing of Chest X-Rays", Optical Engineering, May/June, 1974.

Hall, Earnest L., Richard P. Kruger, A. F. Turner, "Automated Measurements from Chest X-rays for Lung Disease Classification", Computer Graphics, Pattern Recognition and Data Structures Conference, May 1975.

Hall, Earnest L., Richard P. Kruger etaal, "A Survey of Preprocessing and Feature Extraction Technizue for Radiographic Images", IEEE Transactions on Computers, Vol. C-20, No. 9, September, 1971.

Kruger, R. P., "Computer Processing of Radiographic Images", Ph. D. Thesis University of Missouri, Columbia Missouri, May, 1971.

Modestino, J. W., R. W. Fries, "Edge Detection in Noisy Images Using Recursive Digital Filtering", Unpublished R. P. I. report.

Stockham, Thomas G., "Image Processing in the Control of a Visual Model, Proceedings of the IEEE, (July, 1092), pp. 828-842.

APPENDICES

APPENDIX A

SMOOTHED DIFFERENTATION OF SAMPLED DATA

Let N be the total number of data points and n be an odd number representing the number of data points to which a least square smoothing parabola is to be fit. The center value of each interval will be the point to be smoothed and differentiated.

The corresponding parabola is

$$y_k = a_o + a_1 k + a_2 k^2 \qquad -(n-1)/2 \leq k \leq (n-1)/2$$

where y_k here is an estimate of each $\overline{y_k}$ contained in the interval. The error, e_k is then

$$e_k = y_k - \overline{y_k}$$
$$= a_o + a_1 k + a_2 k^2 - \overline{y_k}$$

The squared error over n points is then

$$E^2 = \sum_{-A}^{A} e_k^2 = \sum_{-A}^{A} (a_o + a_1 k + a_2 k^2 - \overline{y_k})^2$$

where

$$A = (n-1)/2$$

The values of a_o, a_1, and a_2 which minimizes the squared error are found by

$$\frac{\partial E^2}{\partial a_o} = 0 = \sum_{k=-A}^{A} 2(a_o + a_1 k + a_2 k^2 - \overline{y_k}) = 0 \qquad \text{(A-1)}$$

$$\frac{\partial E^2}{\partial a_1} = 0 = \sum_{k=-A}^{A} (a_o + a_1 k + a_2 k^2 - \overline{y_k})\,(2k) = 0 \qquad \text{(A-2)}$$

$$\frac{\partial E^2}{\partial a_2} = 0 = \sum_{k=-A}^{A} (a_o + a_1 k + a_2 k^2 - \overline{y_k})\ (2k^2) = 0 \qquad \text{(A-3)}$$

From numerical analysis it can be shown that

$$\sum_{k=1}^{A} k^2 = \frac{n(n^2-1)}{24}$$

$$\sum_{k=1}^{A} k^4 = \frac{n(n^2-1)}{24}\ \frac{(3n^2-7)}{20}$$

Then, eqn (A-1) becomes

$$\sum_{k=-A}^{A} 2a_o + \sum_{k=-A}^{A} 2a_1 k + \sum_{k=-A}^{A} 2a_2 k^2 - \sum_{k=-A}^{A} 2\overline{y_k} = 0$$

$$2a_o n + 2a_2 \frac{n(n^2-1)}{12} - 2\sum_{k=-A}^{A} \overline{y_k} = 0$$

Or,

$$a_o n + a_2 \frac{n(n^2-1)}{12} - \sum_{k=-A}^{A} \overline{y_k} = 0 \qquad \text{(A-4)}$$

Likewise, (A-2) becomes

$$2a_o \sum_{k=-A}^{A} k + 2a_1 \sum_{k=-A}^{A} k^2 + 2a_2 \sum_{k=-A}^{A} k^3 - 2 \sum_{k=-A}^{A} k\overline{y_k} = 0$$

$$2a_1 \frac{n(n^2-1)}{24} - 2 \sum_{k=-A}^{A} k\overline{y_k} = 0$$

Or,

$$a_1 \frac{n(n^2-1)}{12} - \sum_{k=-A}^{A} k\overline{y_k} = 0 \qquad \text{(A-5)}$$

And (A-3) becomes

$$2a_o \sum_{k=-A}^{A} k^2 + 2a_1 \sum_{k=-A}^{A} k^3 + 2a_2 \sum_{k=-A}^{A} k^4 - 2 \sum_{k=-A}^{A} k^2 y_k = 0$$

$$2a_o \frac{n(n^2-1)}{12} + 2a_2 \frac{n(n^2-1)}{12} \frac{(3n^2-1)}{20} - 2 \sum_{k=-A}^{A} k^2 \overline{y_k} = 0$$

Or,

$$a_o \frac{n(n^2-1)}{12} + a_2 \frac{n(n^2-1)}{12} \frac{(3n^2-7)}{20} - \sum_{k=-A}^{A} k^2 \overline{y_k} = 0 \qquad \text{(A-6)}$$

Since the first derivative at k=0 is

$$\dot{y}_o = a_1$$

only the value of a_1 has to be solved. From equation (A-5):

$$a_1 \frac{n(n^2-1)}{12} - \sum_{k=-A}^{A} k\overline{y_k} = 0$$

Or,

$$a_1 = \frac{12}{n(n^2-1)} \sum_{k=-A}^{A} k\overline{y_k}$$

So, by taking n=7

$$a_1 = \frac{12}{7(48)} \sum_{k=-3}^{3} k\overline{y_k}$$

$$a_1 = .0357 \sum_{k=-3}^{3} k\ \overline{y_k}$$

APPENDIX B

CALCULATION OF WEIGHTING FACTORS

A weighting factor is used to weight the derivatives taken to find the outline of the face. Since a two dimensional derivative is being taken, this weighting factor is used to determine whether the horizontal derivative, vertical derivative or some combination of the two carries the most information about the edge structure.

The weighting function is desired to be such that when looking at the top of the head, the vertical derivative only is considered and when at the side of the head, the horizontal derivative is the only one taken. Elsewhere on the face, it should be a combination of the two. This requirement is illustrated in Figure B-1.

A cosine function would account for this type of function. When $\theta=0^{o}$, $\cos(\theta)=1$ so the horizontal component should be multiplied by the $\cos(\theta)$ and the vertical component by $(1-\cos(\theta))$. When at the top of the head $\theta=90^{o}$ and $\cos(\theta)=0$. So the horizontal component would be multiplied by zero and the vertical component by one.

This factor is calculated by first assumming that the head is basically a circle. When looking at the upper half of the face, the circle is centered on the intersection of the nose line, NNL, and the eyebrow, NLEB. Figure B-2 shows this calculation.
Then,

$$\text{Cos}(\theta) = \frac{H}{\sqrt{(\text{NNL}-\text{NLF})^2+(\text{X}-\text{NLEB})^2}}$$

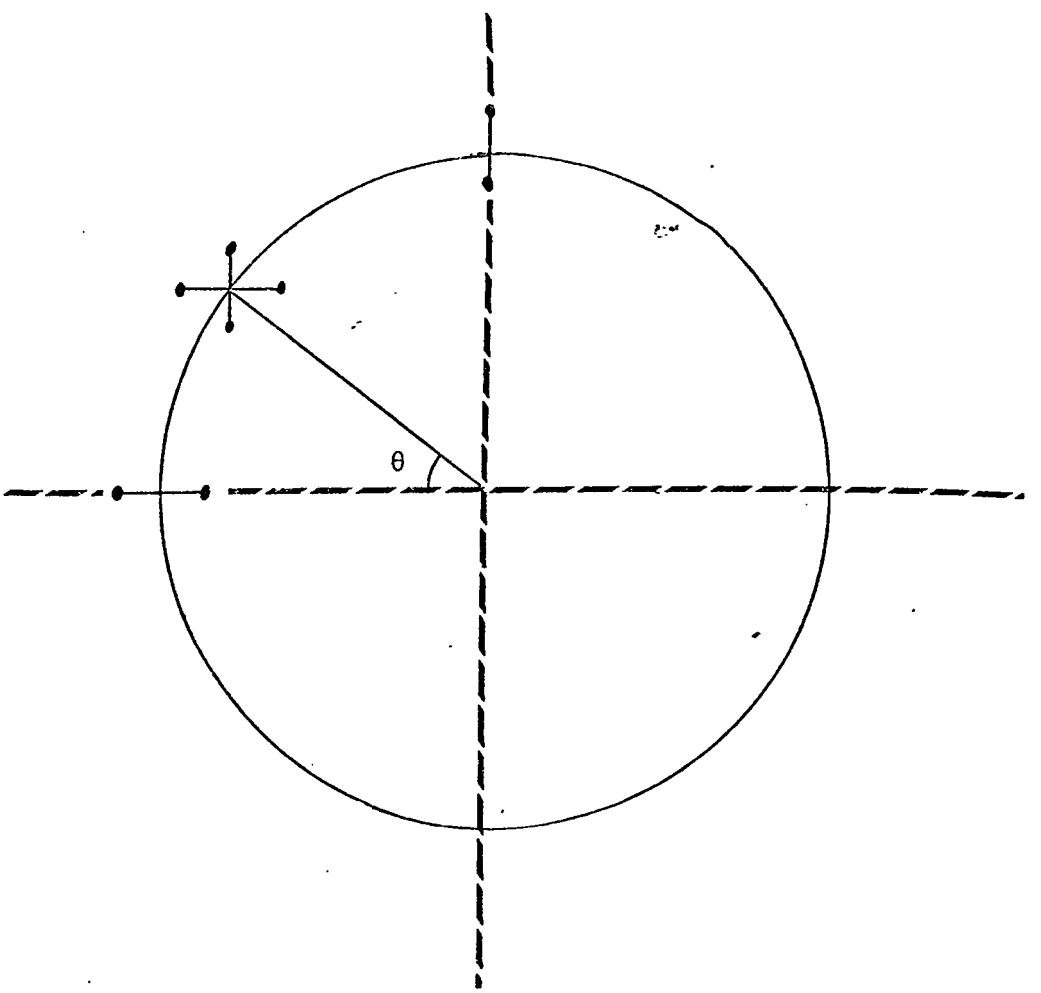
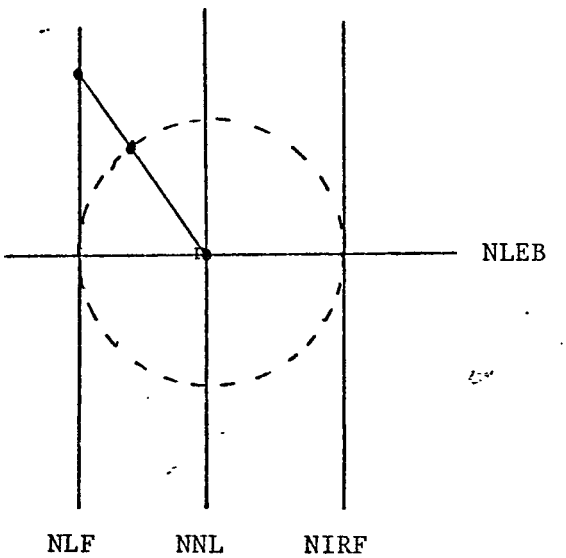


Figure B-1 Two-dimensional derivative requirement



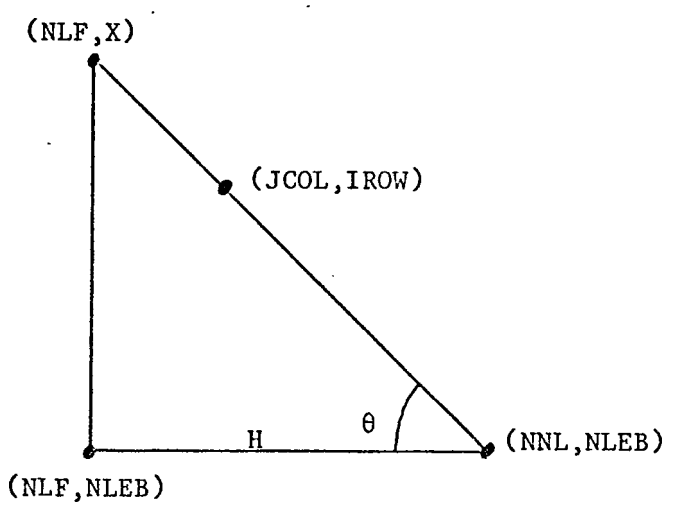


Figure B-2 Calculation of weighting factor

X can be found by assumming that the slope of the line connecting (JCOL,IROW) and (NNL,NLEB) is the same as for (JCOL,IROW) and (NLF,X):

$$\text{Slope} = \frac{(\text{IROW}-\text{NLEB})}{(\text{JCOL}-\text{NNL})} = \frac{(\text{IROW}-\text{X})}{(\text{JCOL}-\text{NLF})}$$

$$\text{X} = \text{IROW} - \frac{(\text{IROW}-\text{NLEB})(\text{JCOL}-\text{NLF})}{(\text{JCOL}-\text{NNL})}$$

APPENDIX C

DIGITAL FILTERS

This discussion of digital filters begins with a basic review of analog filters. First, consider a basic lowpass filter. Figure C-1a shows a simple lowpass filter circuit diagram. Figure C-1b gives the corresponding Laplace transform version of the filter in Figure C-1a. The transfer function, G(s), of the lowpass filter is given by

$$G(s) = \frac{e_{out}(s)}{e_{in}(s)}$$

But, by the voltage-divider law, $e_{out}(s)$ for Figure C-1b is

$$e_{out}(s) = e_{in}(s) \frac{1/Cs}{R + 1/Cs}$$

so that

$$G(s) = \frac{e_{out}(s)}{e_{in}(s)} = \frac{1/Cs}{R + 1/Cs} = \frac{1}{RCs + 1} \qquad \text{(C-1)}$$

The step response of this type of filter can be found by

$$e_{os}(t) = L^{-1}\{e_o(s)\} = L^{-1}\{G(s)e_{in}(s)\}$$

assuming zero intitial conditions and where,

$e_{in}(s) = 1/s$ for a step response

$G(s)$ is as given in eqn. (C-1)

L is the Laplacian operator

So,

$$e_{os}(t) = L^{-1}\left[\frac{1}{RCs + 1}\left(\frac{1}{s}\right)\right]$$

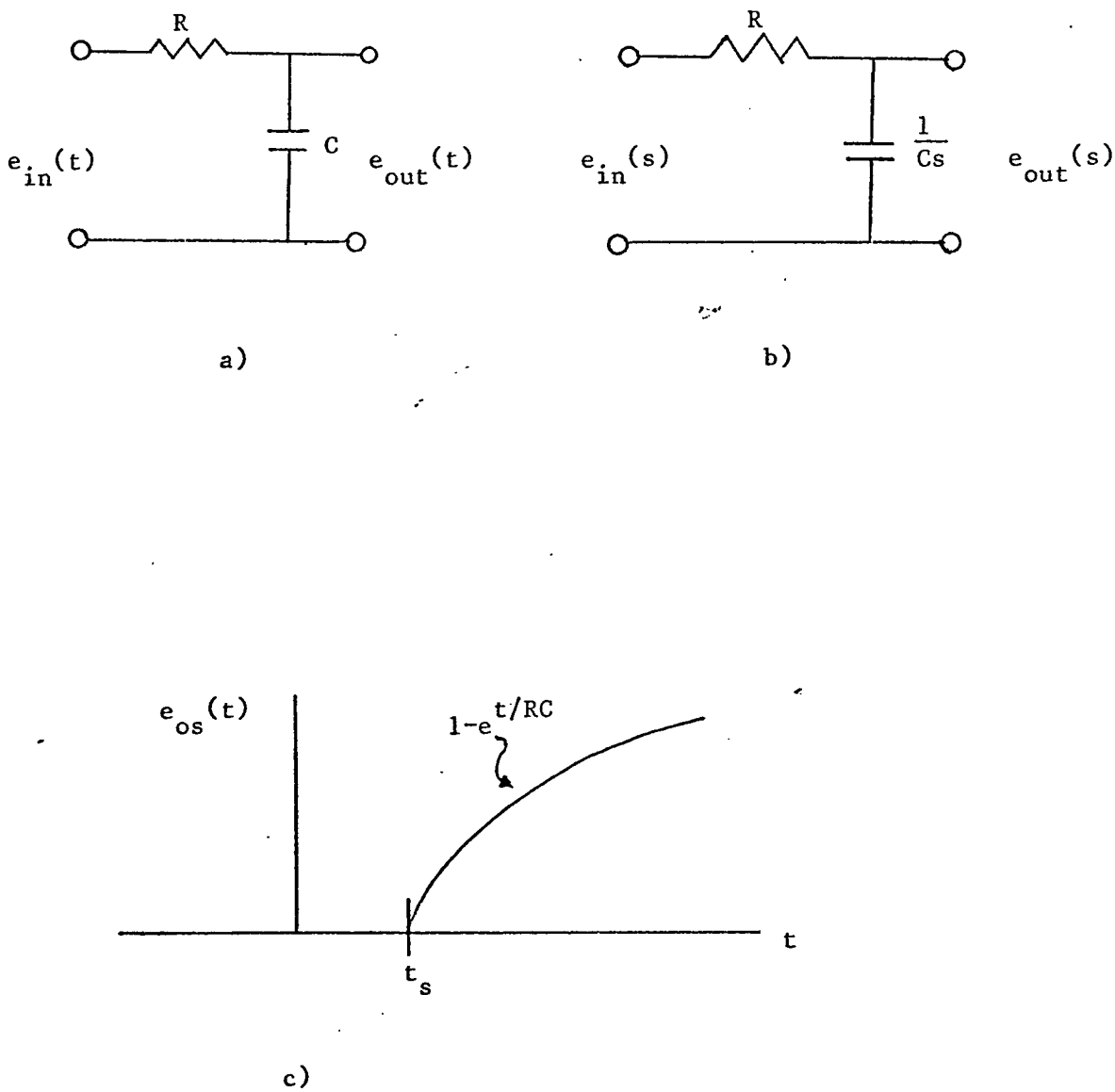


Figure C-1 Circuit characteristics of a lowpass filter. a) circuit diagram b) Laplace transform c) step response

$$e_{os}(t) = L^{-1}\{1/(RCs^2 + s)\} \qquad (C\text{-}2)$$

$$= L^{-1}\{1/(s(RCs + 1))\}$$

which can be evaluated using Partial Fractions;

$$\frac{1}{s(RCs+1)} = \frac{A}{s} + \frac{B}{RCs+1} = \frac{A(RCs+1) + Bs}{s(RCs+1)}$$

Therefore,

$$1 = ARCs + A + Bs$$

$$= s(ARC + B) + A$$

Or, $$(ARC+B) = 0 \qquad (C\text{-}3)$$

$$A = 1 \qquad (C\text{-}4)$$

Solving (C-2) for the B variable

$$B=-ARC$$

and substituting in eqn (C-4) yields

$$B=-RC$$

Therefore, eqn (C-2) can be written

$$e_o(t) = L^{-1}\left[\frac{1}{s} + \frac{-RC}{RCs+1}\right]$$

$$= L^{-1}\left[\frac{1}{s}\right] + L^{-1}\left[\frac{-RC}{RCs+1}\right]$$

$$= L^{-1}\left[\frac{1}{s}\right] + L^{-1}\left[\frac{1}{s+1/RC}\right] \qquad (C\text{-}5)$$

From Laplacian tables it can be found that

$$L^{-1}\left[\frac{1}{s + a}\right] = e^{-at}$$

Then when a=0

$$L^{-1}\left[\frac{1}{s}\right] = 1$$

and when a=1/RC

$$L^{-1}\left[\frac{1}{s+1/RC}\right] = e^{-t/RC}$$

Therefore the unit step response of a simple lowpass filter is of the form

$$e_{os}(t) = 1-e^{-t/RC} \qquad (C-6)$$

Figure C-1c gives a general plot of this response. As can be seen, a lowpass filter acts as an integrator on the input function. If this type of filter were applied to an image the edges, corresponding to a step increase, would be blurred due to the response given in eqn (C-6). It therefore would not serve as an edge detector.

Next, consider a highpass filter. Figure C-2a and C-2b shows a simple highpass filter and its corresponding Laplace transform version, respectively.

The transfer function of this highpass filter is given by

$$G(s) = \frac{s}{s+1/RC} \qquad (C-7)$$

The step response of this filter can be determined in the same manner as for the lowpass filter and is given by

$$e_{os}(t) = e^{-t/RC}$$

Figure C-2c gives a general plot of this response. As shown in Figure C-2c, when applied to an image a highpass filter acts more like a differentiator, assigning the maximum value at the point where the edge is encountered.

The next problem is to change this analog highpass filter into its digital filter version. First, let

G(s) = analog transfer function

H(z) = digital transfer function
(where z^{-1} represents one sample delay)

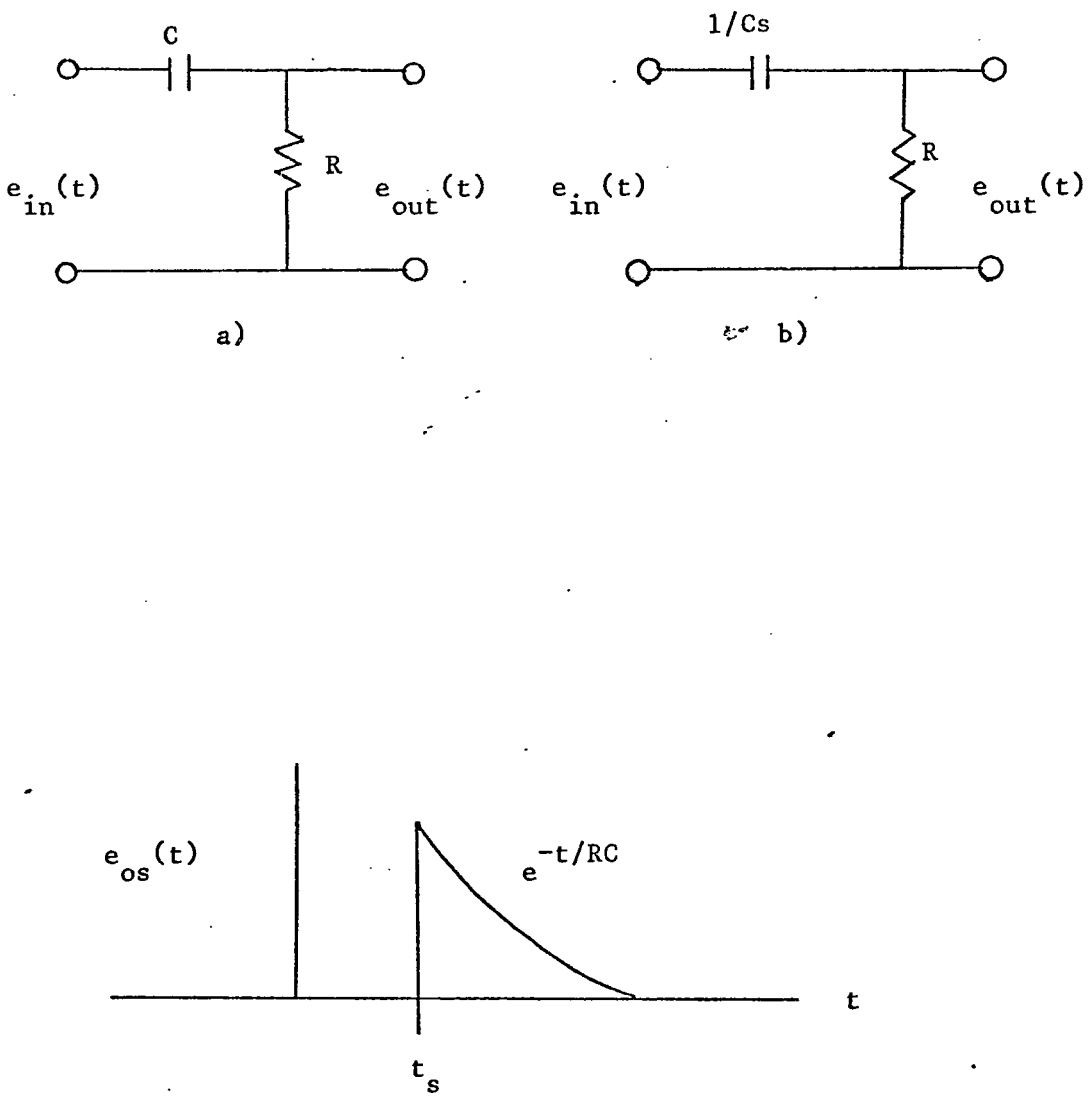


Figure C-2 Circuit characteristics of a highpass filter. a) circuit diagram b) Laplace transform c) step response

and let, λ_r = analog filter reference frequency (rad/sec)

ω_r = digital filter reference frequency (rad/sec)

Then, it can be shown

$$H(z) = G(s)\Big|_{s=\frac{C(1-z^{-1})}{(1+z^{-1})}}$$

where, $C = \lambda_r \cot \frac{\omega_r T}{2}$

For example, assume G(s) is a simple highpass filter of the form

$$G(s) = \frac{s}{s+1}$$

Then, for a digitized photograph with 128 pixels per line

f_s = sampling frequency

= 128 samples/line

$T = 1/f_s$

= 1/128 lines/sample

so, if f_n is the Nyquist frequency

$f_n = f_s/2\pi$

= 64 samples/line

Let $f_r = \omega_r/2$

$= 3/4\ f_n$

$= (3/4)\ (f_s/2)$

$= 3f_s/8$

and let $\lambda_r = 1$

Then,

$$\frac{\omega_r T}{2} = \frac{(3\pi f_s/4)(1/128)}{2} = \frac{3\pi}{8}$$

Therefore, solving for C

$$C = \cot(\omega_r T/2) = .414$$

and

$$H(z) = G(p)\Big|_{p=\frac{C(1-z^{-1})}{(1+z^{-1})}}$$

$$= \frac{\frac{.414(1-z^{-1})}{(1+z^{-1})}}{1 + \frac{.414(1-z^{-1})}{(1+z^{-1})}}$$

After simplifing the above term

$$H(z) = \frac{.2927(1-z^{-1})}{1 + .414z^{-1}}$$

From filter theory it is known that if X(z) is the input to the filter H(z) and Y(z) is the corresponding output

$$Y(z) = H(z)X(z)$$

$$= \frac{.2927(1-z^{-1})}{1+.414z^{-1}} X(z)$$

Now, getting Y(z) in the form of a normal digital filter

$$Y(z) + .414z^{-1}y(z) = .2927(1-z^{-1})X(z)$$

$$Y(z) = .2927(X(z)-z^{-1}X(z))-.414z^{-1}Y(z)$$

Note that $z^{-1}X(z)$ and $z^{-1}Y(z)$ represent the previous input and output respectively. So the algorithm may be written in general:

$$Y(1) = AX(1) \qquad I=1$$

$$Y(I) = A(X(I)+X(I-1))+BY(I-1) \qquad I>1$$

where A=.2927 and B=.414 for this example.

APPENDIX D

DETAILED PROGRAMMING DESCRIPTION

Appendix D will describe in detail, the CAPTURE Algorithm. A block diagram of the program flow is given in Figure D-1. Each of the routines shown in Figure D-1 will be described in the following paragraphs. Routines developed specifically for the CAPTURE Algorithm will be described by first giving a detailed technical description of the programing methods and philosophy. Flow charts and a listing of the routine will also be given for further reference.

The library routines EXEC and SCOPE, will be described by simply giving the program description and useage.

Mainprogram FILT

The purpose of mainprogram FILT is to locate the major facial features, drawing vertical or horizontal lines as necessary for their definition. FILT calls subroutine WRNR to produce the facial outline. All pertinent information is passed to the subroutines through COMMON.

FILT mainprogram technical description. The following major variables are used in mainprogram FILT:

NAME --File name of desired image

IM --A 128 element array for holding a row or column of pixel values

ISUM --A 128 element array used in finding signatures

IHOLD--A 128 by 70 matrix to hold pixel values

LF --Vertical line depicting the left side of the face

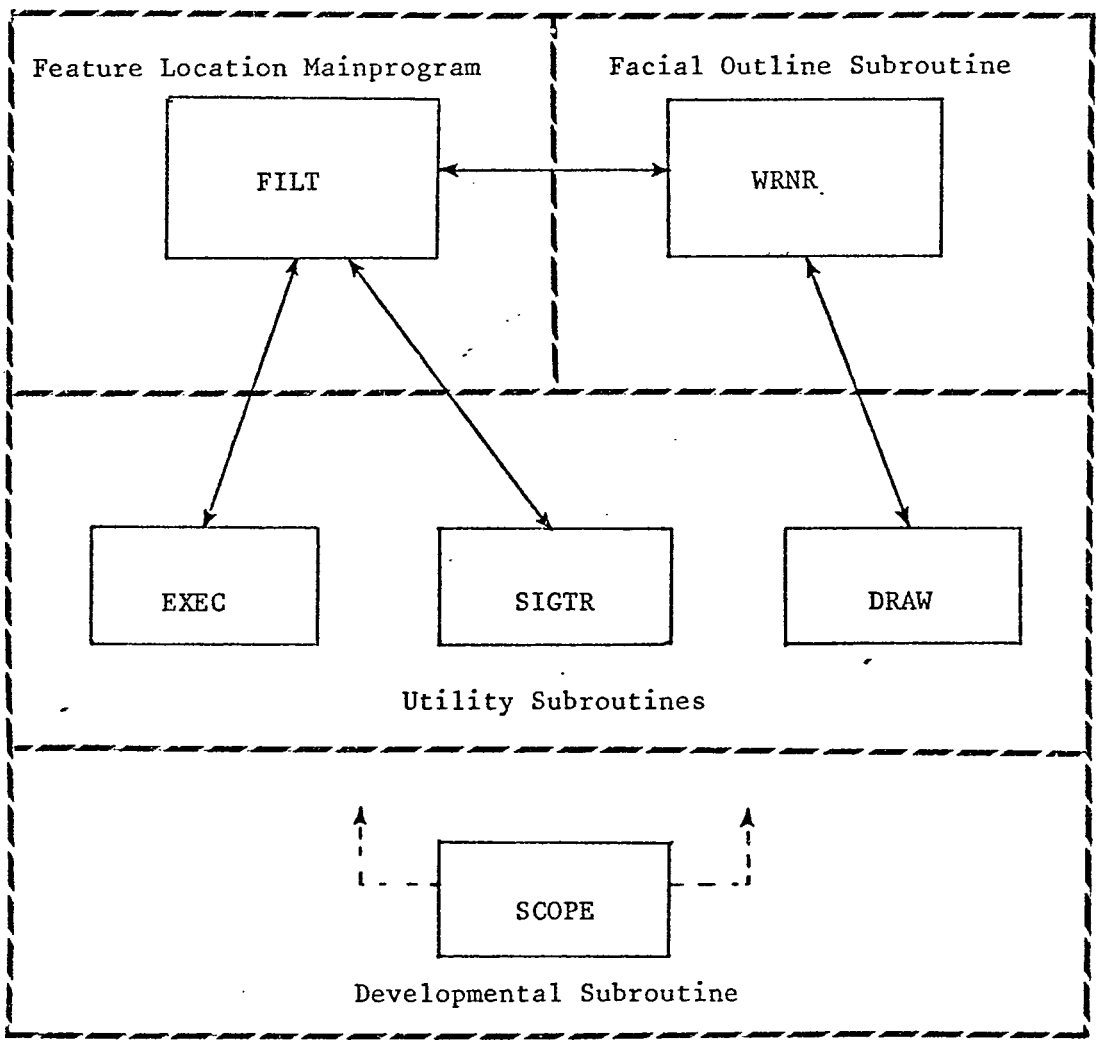


Figure D-1 CAPTURE Algorithm program flow

IRF --Vertical line depicting the right side of the face

NL --Vertical line depicting the nose line

NT --Horizontal line depicting the nose tip

ML --Horizontal line depicting the mouth

ITH --Horizontal line depicting the top of the head

IBH --Horizontal line depicting the bottom of the head

IHAR--Horizontal line depicting the hairline

LE --Horizontal line depicting the eye

LEB --Horizontal line depicting the eyebrow

Figure 1-1 indicates the locations of these lines.

FILT begins by asking for the name of the file containing the digitized version of the desired image. It then fills the IHOLD array with the image's pixel values. In order to conserve core space, only the center 70 columns are read into the IHOLD array. Since most heads are not as wide as they are long, pictures of the head region will have a great deal of wasted space at the left and right extremes of the picture. If the image is centered properly, the center 70 columns should contain all the desired information. However, since column one of IHOLD now corresponds to column 34 of the original image, and since the row and column numbers of the features are desired to be in terms of the original image's coordinates, care must be taken to index the IHOLD array properly.

A call to SIGTR is next made to find the row signature of 10 lines near the center of the image so that an estimate of the left, LF, and right, IRF, side of the face can be determined. Figure D-2 shows an example of the results of this row sum.

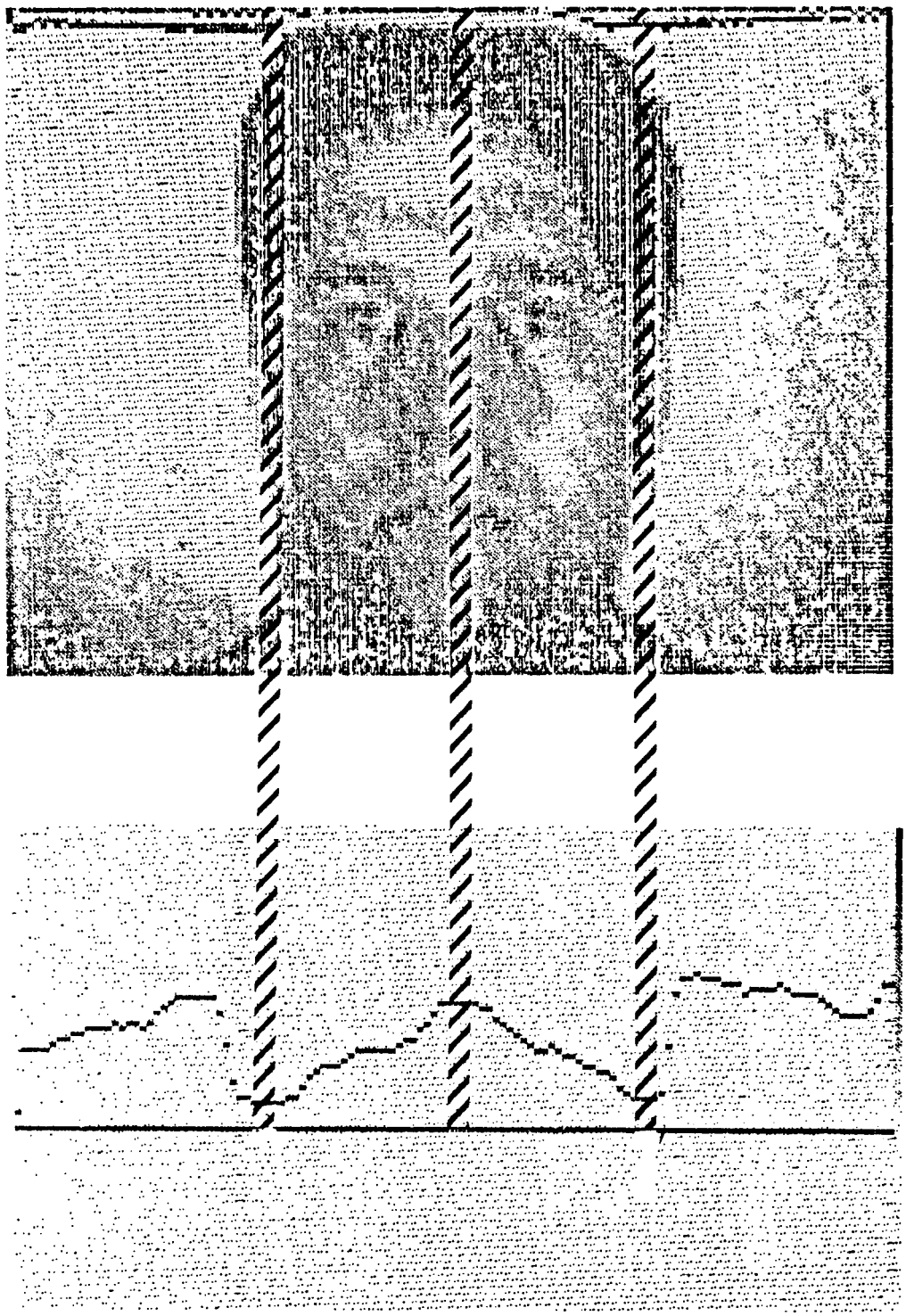


Figure D-2 Row signature example

The estimate for the left side of the face is obtained by finding the minimum of the first half of the signature. The estimate for the right side of the face is likewise found by determining the minimum of the last half of the signature. Using these estimates for LF and IRF, the nose line, NL, can be found by determining the maximum of the signature between LF and IRF.

The next task, then, is to find the top of the head, ITH. This is accomplished by taking a finite difference derivative along the nose line, NL. The top of the head should then be at the point of the maximum negative derivative when searching the top quarter of the derivative array. The derivative should be negative since at the top of the head point we will have a low pixel value, due to the dark hair, minus a large pixel value due to the light background. Figure D-3 shows an example of the derivative taken down the nose line.

The hairline, IHAR, can similarly by found by determining the maximum positive derivative in the top half of the image working from ITH down. We would expect the derivative to be positive since at the point of the hairline we will have a high pixel value due to the forehead, minus a lower pixel value due to the hair.

Estimates of the bottom of the head, IBH, and the mouth line, ML, are then formed. In general most heads are no more than three times as long as they are wide. Therefore, this criterion is used to form an estimate for the bottom of the head. Since this value is used only as a limit for DO loops, an exact determination is not necessary at this point. An estimate for the mouth line is formed by assuming it to be 30 rows above the IBH line. This value will be determined exactly at a later time.

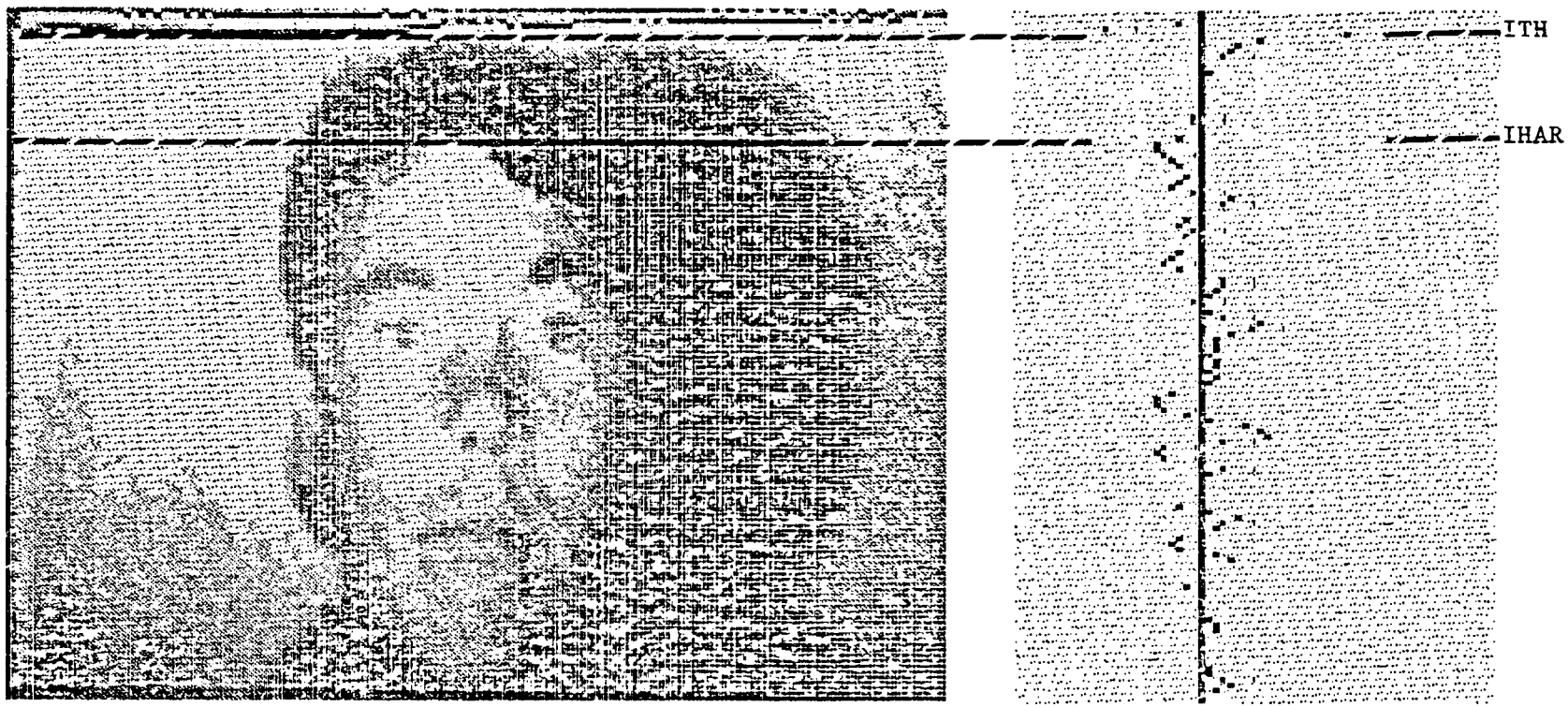


Figure D-3 Finite Difference Derivative along Noseline

Next, the eyebrow, LEB, and the eyeline, LE, can be found. This is accomplished by first forming a column signature between columns starting at five columns to the right of the nose line and ending at five columns to the left of the right side of the face. Notice that in the coding this corresponds to NL-28 to IRF-38 and not to NL+5 to IRF-5 as might be expected. This is due to the 33 column shift between the IHOLD matrix and the original picture. Since NL and IRF are in terms of the original picture we must subtract 33 from these values in order to index the IHOLD array properly. The eyebrow, containing many low value pixels, should then be the minimum value of this signature.

The eyeline, LE, is next desired to be found. Using the same signature as determined for finding the eyebrow, and working from LEB down, we would expect to find next a peak corresponding to the space between the eyebrow and the eye, and then a minimum corresponding to the eye itself. So these are the next conditions for which FILT searches.

The tip of the nose, NT, can now be determined. Since the nose protrudes out from the face, it is usually very bright in the images. Therefore, by forming a column sum just around the nose line, NL, and by determining the maximum, the nose tip can be found. So, to this end, a column signature is formed over the five rows on either side of the nose line. Notice here that the DO loop variable is from NL-5 to NL+5. The column offset between the IHOLD array and the original picture is accounted for by subtracting 33 from the column coefficient in IHOLD. The maximum is searched for between rows starting at 15 rows below the eyeline, and ending 15 rows before the mouthline. This maximum is taken to be NT.

The next task is to define the mouth line ML, more accuratly. The mouth line can be found using the same signature used to determine the nose tip. This signature is searched between rows beginning at 10 rows above the old estimate for the mouth line and ending at 20 rows from the bottom of the head. Since the mouth will contain many low pixel values, the mouth line, ML, is taken to be the minimum of the search.

A section of coding is next entered to determine the right and left extremities of the mouth. These values will not be made available as a program output since these values are too variable. These values would widely vary for example, in a picture if the suspect was smiling than if he were not smiling. These values will be used however in the routine to determine the facial outline as a limiting factor in DO loops. These values are determined by first forming a row signature involving ten rows centered around ML. The corners of the mouth, being recessed into the face more than at the center, should contain more low pixel values. Therefore the left corner of the mouth, MLL, can be found by searching for the minimum of the row signature between the center of the left part of the face to the nose line. The right corner of the mouth, MLR, is similarly found by searching for the minimum signature between the nose line and the center of the right side of the face.

FILT's final function is to call subroutine WRNR to produce the facial outline. All necessary feature location parameters, including MLL and MLR, are passed to WRNR through the COMMON block.

FILT mainprogram flow chart. The next few pages present a flow chart for FILT. This flow chart is very general in nature and is intended to be used only as an aide in understanding mainprogram FILT.

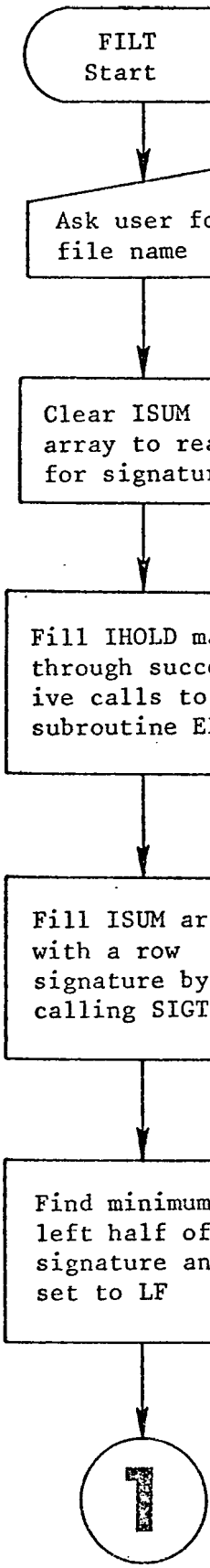
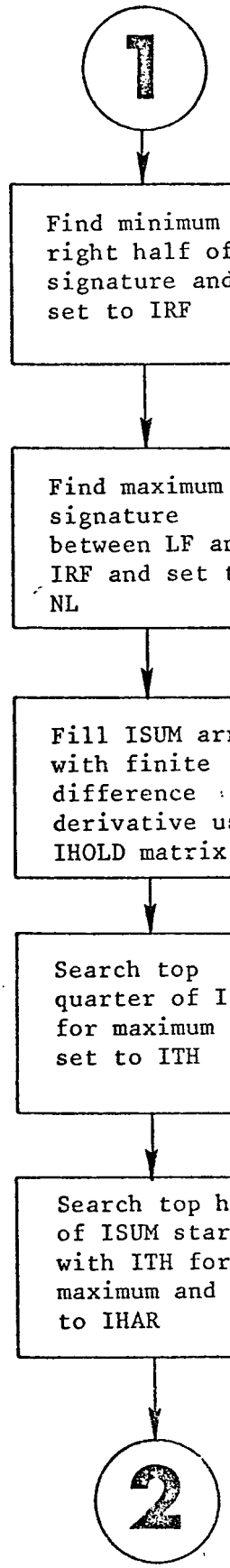
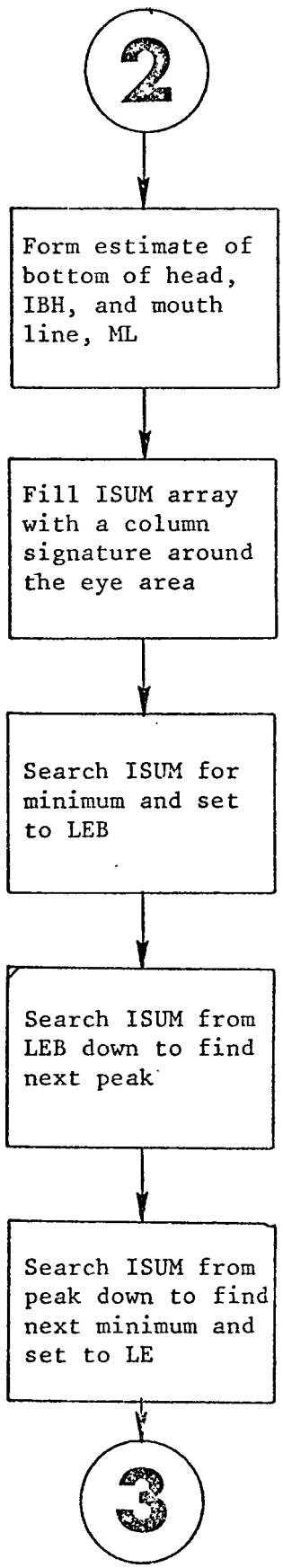
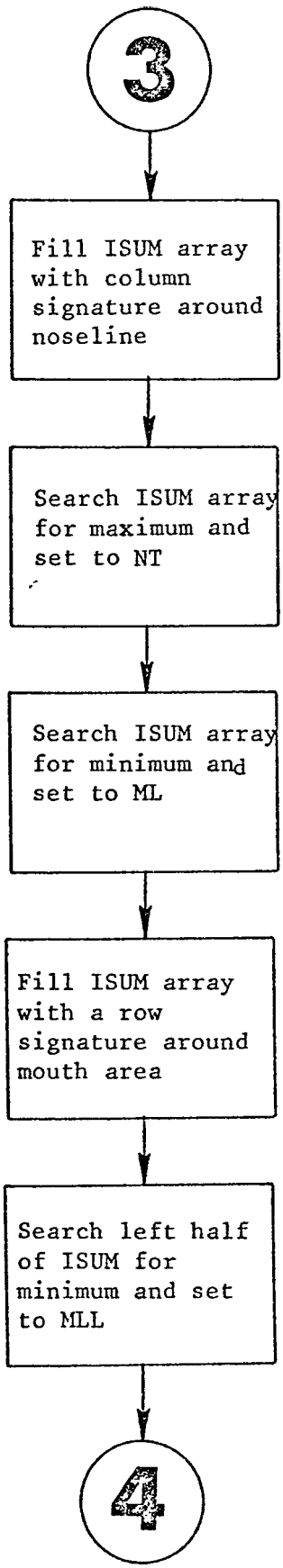
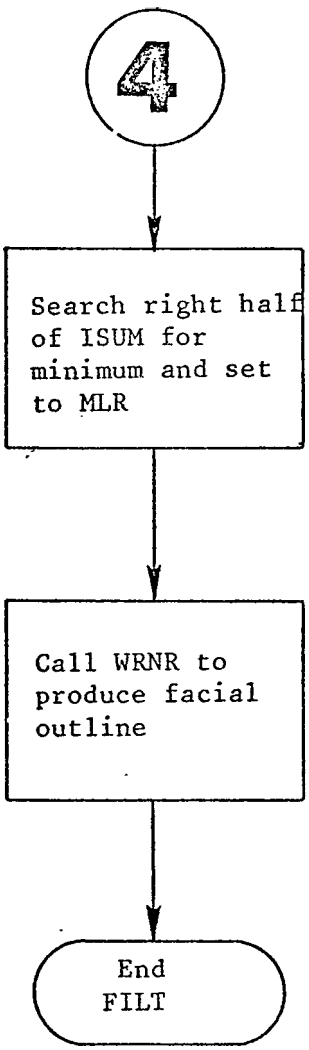


Figure D-4 Mainprogram FILT Flow Chart


1
Find minimum of right half of signature and set to IRF
Find maximum of signature between LF and IRF and set to NL
Fill ISUM array with finite difference derivative using IHOLD matrix
Search top quarter of ISUM for maximum and set to ITH
Search top half of ISUM starting with ITH for maximum and set to IHAR
2


2
Form estimate of bottom of head, IBH, and mouth line, ML
Fill ISUM array with a column signature around the eye area
Search ISUM for minimum and set to LEB
Search ISUM from LEB down to find next peak
Search ISUM from peak down to find next minimum and set to LE
3


3
Fill ISUM array with column signature around noseline
Search ISUM array for maximum and set to NT
Search ISUM array for minimum and set to ML
Fill ISUM array with a row signature around mouth area
Search left half of ISUM for minimum and set to MLL
4


4
Search right half of ISUM for minimum and set to MLR
Call WRNR to produce facial outline
End FILT

FILT mainprogram listing. This discussion of mainprogram FILT is concluded with a listing of the coding, as presented in Figure D-5. Each of the subroutines called by FILT will be described in detail in later sections.

Subroutine WRNR

The purpose of subroutine WRNR is to form the facial outline using the feature location lines as determined by mainprogram FILT as an aid. In general, the outline is formed by first passing the image through a digital filter as described in Chapter 3 and in Appendix C. The next step is to set a threshold for the filter output. Any filtered pixel values that are greater than this value are set to a zero, while any that are less than the threshold are set to one. The face is then divided into six regions as given in Figure D-6. Each of these regions are searched in the directions given in Figure D-6. The facial outline edge then is determined to be at the location where the first "1" is encountered in the matrix.

WRNR subroutine technical description. The feature location information is passed to WRNR through COMMON by mainprogram FILT. These variable names will remain the same in WRNR. The additional major variables used by WRNR and their function are:

HOPT(I) -An array holding the filtered pixel values for one row of IHOLD.

A, B11, A10, A11-Fixed variables used in implememtation of the digital filter. (See Chapter 3)

ICNV, JCNV -Conversion factors reflecting the offset of the IHOLD matrix from the original picture in horizontal and vertical directions respectively

```
001   FTN
002         PROGRAM FILT
003   C           LINDA BROMLEY
004         COMMON IHOLD(128,70),NAME(3),IM(128)
005         COMMON LF,IRF,NL,ITH,IBH,IHAR,ML,LE,NT,LEB,MLL,MLR
006         DIMENSION ISUM(128),IDERV(128)
007         WRITE(1,40)
008      40 FORMAT("FILE-")
009         READ(1,11) NAME
010      11 FORMAT(2A2,A1)
011         DO 9 I=1,128
012       9 ISUM(I)=0
013         DO 20 J=34,103
014         CALL EXEC(14,103B,IM,128,NAME,J)
015         DO 3 I=1,128
016         IHOLD(I,J-33)=IM(I)
017       3 CONTINUE
018      20 CONTINUE
019         CALL SIGTR(NAME,IM,1,128,45,55,1,ISUM)
020         IMIN=0
021         MIN=1000
022         DO 10 I=64,30,-1
023         IF(ISUM(I).GE.MIN) GO TO 10
024         IMIN=I
025         MIN=ISUM(I)
026      10 CONTINUE
027         LF=IMIN
028         IMIN=0
029         MIN=1000
030         DO 15 I=64,103
031         IF(ISUM(I).GE.MIN) GO TO 15
032         IMIN=I
033         MIN=ISUM(I)
034      15 CONTINUE
035         IRF=IMIN
036         MAX=0
037         IMAX=0
038         DO 63 I=LF,IRF
039         IF(ISUM(I).LE.MAX) GO TO 63
040         IMAX=I
041         MAX=ISUM(I)
042      63 CONTINUE
043         NL=IMAX
044   C***TAKE DERIVITIVE DOWN NOSELINE.
045         DO 333 I=2,126
046         ISUM(I)=IHOLD(I+1,NL-33)-IHOLD(I-1,NL-33)
047     333 CONTINUE
048   C***  ISUM CONTAINS THE DERIVATIVE DOWN THE NOSE LINE.
049   C***TOP OF HEAD IN TOP QUARTER.
050         MAX=0
051         IMAX=0
052         DO 335 I=2,32
053         IS=IABS(ISUM(I))
054         IF(IS.LE.MAX) GO TO 335
055         IMAX=I
056         MAX=IS
057     335 CONTINUE
058         ITH=IMAX
```

Figure D-5 FILT mainprogram listing

```
C***LOOK FOR HAIRLINE IN TOP HALF.
      MAX=0
      IMAX=0
      DO 739 I=ITH+5,64
      IF(ISUM(I).LE.MAX) GO TO 739
      MAX=ISUM(I)
      IMAX=I
  739 CONTINUE
      IHAR=IMAX
C***FORM ESTIMATE OF MOUTH LINE AND BOTTOM OF HEAD.
      IBH=ITH+3*(IRF-LF)
      IF(IBH.GT.127) IBH=127
      ML=IBH-30
C***BLOCK OUT EYE.
      DO 801 I=(IHAR+10),(ML-20)
      IISUM=0
      DO 800 J=(NL-28),(IRF-38)
  800 IISUM=IISUM+IHOLD(I,J)
      ISUM(I)=IISUM
  801 CONTINUE
      MIN=1000
      IMIN=0
      DO 1 I=(IHAR+10),(ML-20)
      IF(ISUM(I).GE.MIN) GO TO 1
      IMIN=I
      MIN=ISUM(I)
    1 CONTINUE
      LEB=IMIN
C****FIND NEXT PEAK.  (SPACE BETWEEN EYEBROW AND EYE)
      MAX=0
      DO 2 I=(LEB+1),(LEB+16)
      IF(ISUM(I).LT.MAX) GO TO 4
      MAX=ISUM(I)
    2 CONTINUE
C****FIND NEXT MINIMUM
    4 MIN=ISUM(I)
      DO 5 J=I,(LEB+16)
      IF(ISUM(J).GT.MIN) GO TO 6
      MIN=ISUM(J)
    5 CONTINUE
    6 LE=J-1
      DO 117 I=LE,IBH
      IISUM=0
      DO 116 J=(NL-5),(NL+5)
  116 IISUM=IISUM+IHOLD(I,J-33)
      ISUM(I)=IISUM
  117 CONTINUE
      MAX=0
      IMAX=0
      DO 167 I=(LE+15),(ML-15)
      IF(ISUM(I).LE.MAX) GO TO 167
      IMAX=I
      MAX=ISUM(I)
  167 CONTINUE
      NT=IMAX
      IMIN=0
      MIN=1000
      DO 168 I=(ML-10),(IBH-20)
      IF(ISUM(I).GE.MIN) GO TO 168
      IMIN=I
      MIN=ISUM(I)
  168 CONTINUE
      ML=IMIN
```

```
C**** FIND WIDTH OF MOUTH
      JSTART=NL-(NL-LF)/2
      JSTOP=NL+(IRF-NL)/2
      DO 900 J=JSTART,JSTOP-
      IISUM=0
      DO 901 I=(ML-5),(ML+5)
  901 IISUM=IISUM+IHOLD(I,J-33)
      ISUM(J)=IISUM
  900 CONTINUE-
      MIN=1000
      IMIN=0
      DO 902 J=JSTART,JSTOP
      IF(J.EQ.NL) GO TO 903
      IF(ISUM(J),GT.MIN) GO TO 902-
      MIN=ISUM(J)
      IMIN=J
      GO TO 902
  903 MLL=IMIN-
      MIN=1000-
      IMIN=0
  902 CONTINUE-
      MLR=IMIN-
      CALL WRNR
      STOP
      END
      END$
*** LIST END ****
```

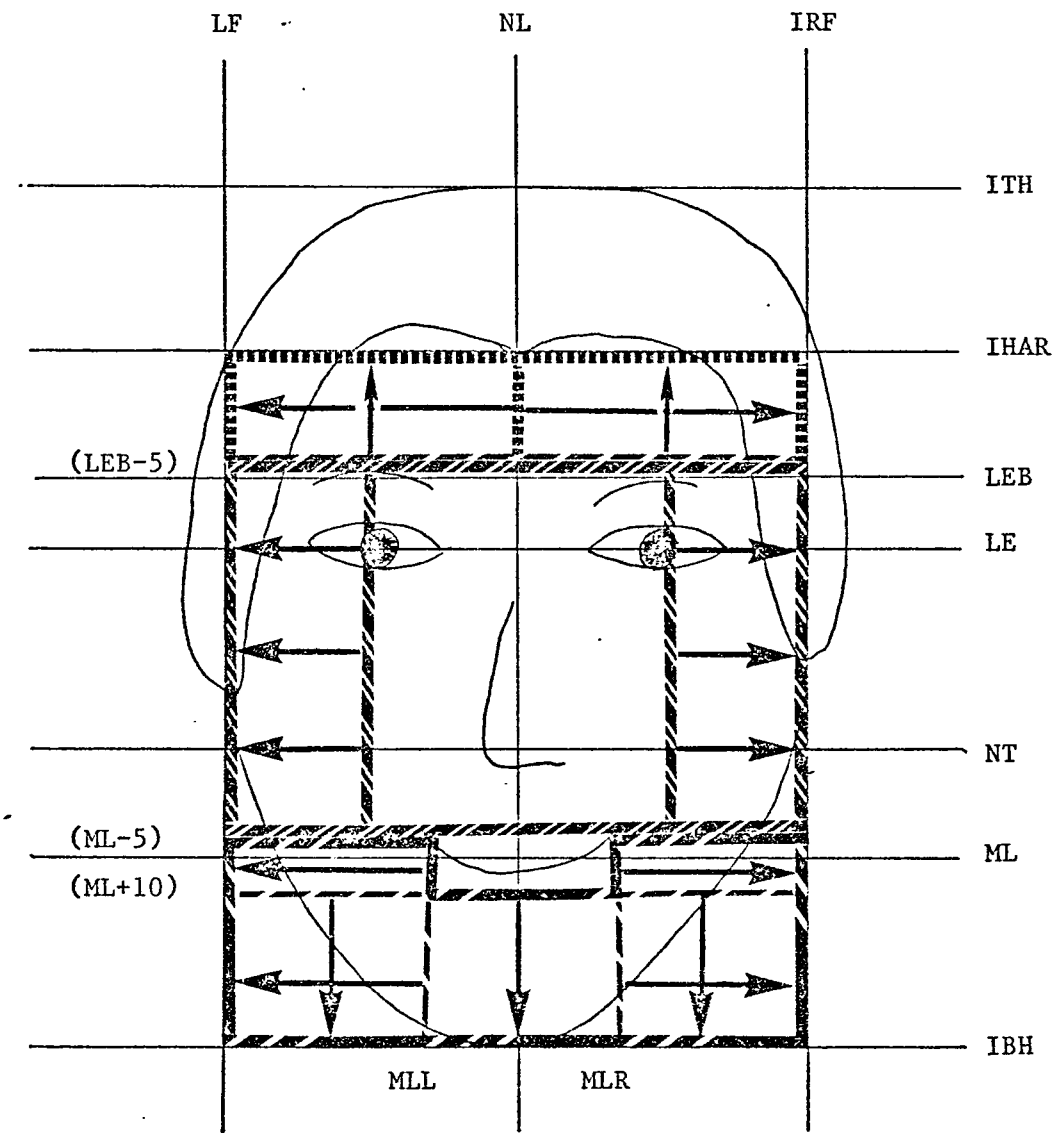


Figure D-6 Facial outline search regions

Z1M,Z2M-Corresponds to the Z_1^{-1} and Z_2^{-1} terms in the digital filter, respectively

Nxxx -Where xxx is a feature location variable (NIBH for example). Represents the converted form of location variable xxx.

Subroutine WRNR begins by defining the filter constants and threshold. These values were determined to be the best for general use by a process of trial and error. A loop is then entered that sets up a matrix, IHOLD, to hold the image values. Since most pictures have a great deal of wasted space on either side of the head in the image, only 70 columns are stored for more efficient use of memory. Which 70 columns are to be stored are determined by where the feature location program found LF to be positioned. The IHOLD matrix is then set up so that column one corresponds to the tenth column to the left of LF. Also, since the top of the head may not be at row one of the image file, the rows of IHOLD were set up so that row one corresponds to ITH. This orientation of the IHOLD matrix is given in Figure D-7.

After the IHOLD matrix has been filled, the digital filter is set up for implementation. For a detailed description of the particular filter being implemented, refer to Chapter 3. A row of IHOLD is filtered and the result is contained in array HOPT. This array is then searched, comparing each filtered pixel value with the threshold value. All pixels having a filtered value less than the threshold are replaced with the value of "1". All others are set to zero. This process is repeated until all rows of IHOLD have been filtered.

The IHOLD matrix at this point contains only ones and zeros. They are arranged, however, so that most of the pixel values in the face area are now zero and those in the hair and along the chin line are ones. So,

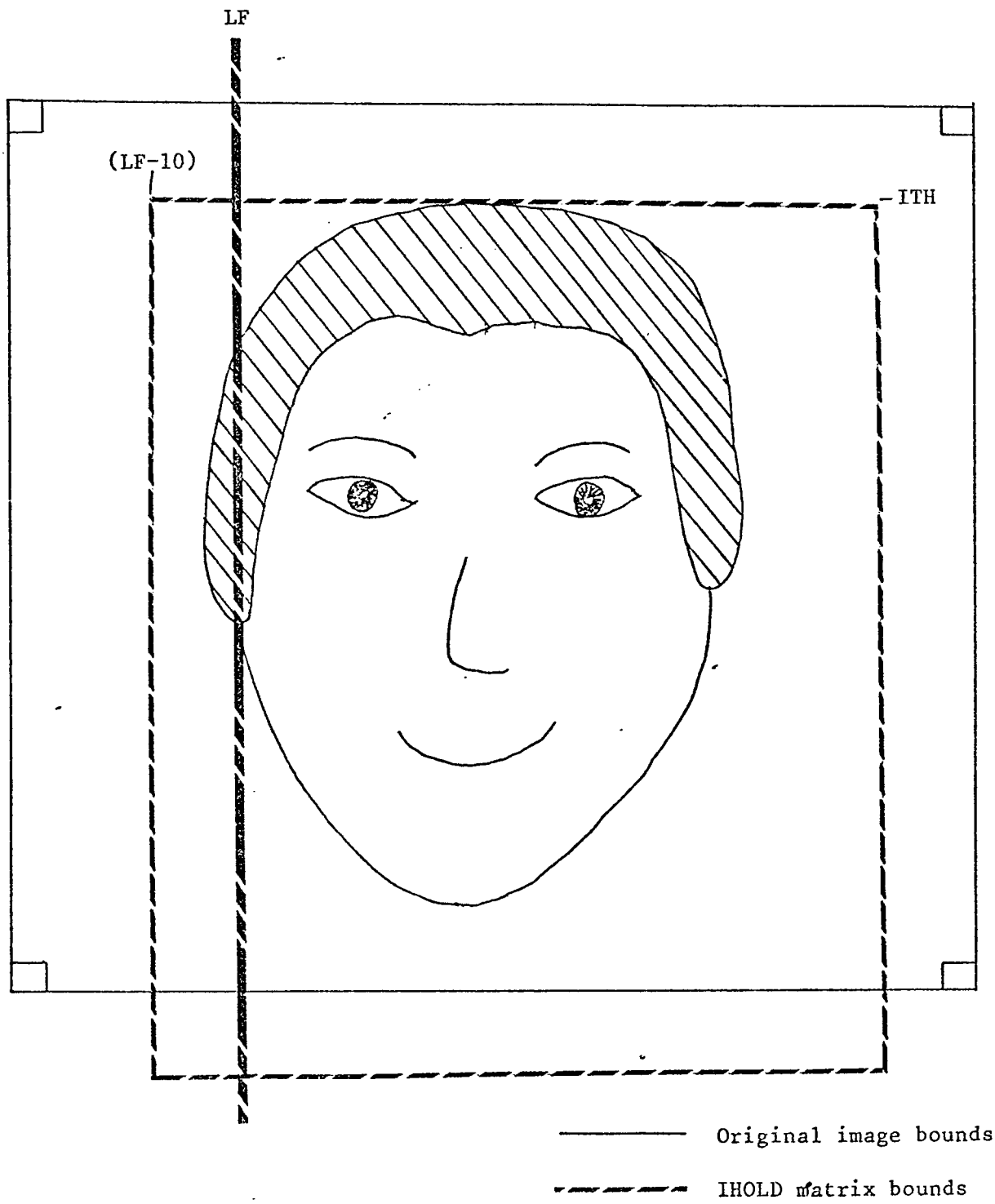


Figure D-7 Orientation of the IHOLD matrix

the process is then to search from the center of the face outward and select the first column encountered whose pixel value has been set to one to be the face edge. However, some of the darker areas in the face, such as the eyes and mouth, may also contain a few scattered ones that may interfere with the edge detection scheme. However, since these areas have already been located by mainprogram FILT, we can use this information to skip over these areas. Therefore the areas that are searched are the six regions as given in Figure D-6.

After determining the converted values for each of the feature location parameters a search is begun over regions 1 and 2 from a point begining at five rows above the eyebrow line, up to the hair-line. Starting at the left side of the face, a search is conducted up the column until a row is found that contains a value of 1. When a 1 is found, it is replaced with a value of 64 to indicate that it was the first value of 1 encountered. The next column is then addressed and the process is repeated. After searching regions 1 and 2 upwards in this manner, the regions are searched in a horizontal direction. Searching each region in two directions will provide a smoother appearing outline. Region 1 is first searched starting at the noseline and moving left to the left side of the face. Again, when the first value other than a zero is encountered, which could be a value of 1 or a value of 64 from the previous search, the pixel value is reset to be a value of 64. In a simular manner region 2 is searched starting at the nose line and moving right to the right side of the face.

After the edges have been defined for regions 1 and 2, region 3 and 4 are searched. Since these two regions are at the sides of the

head, they need only to be searched in the horizontal directions. Region 3 is searched starting from a point midway between the NLF and NNL points and moving left until the left side of the face is reached. In some cases the search may not encounter a pixel of value 1 on a given row. In this case there would be a gap in the facial outline. To prevent this a running account, LSTL, is kept to indicate the column at which the edge was detected in the previous row. If no edge is detected in a subsequent row, this value is taken to the edge. Region 4 is searched in a similar manner starting at a point midway between the NLF and NIRF points and moving right to the right side of the face.

After all edges have been detected in region 3 and 4, regions 5 and 6 are searched. Region 5 and 6 are first searched starting at a point ten lines below the mouth and moving down to the bottom of the head. Region 5 is then searched starting at the left edge of the mouth, NMLL, as determined by FILT and moving left to the left side of the face. Region 6 is similarly searched from NMLR to the right side of the face. In both regions 5 and 6 a running count is kept to determine the edge column for the previous row in case no edge is found.

After all six regions have been searched and the edge structure defined, subroutine DRAW is called to prepare the output file to be displayed on the electrostatic printer. All information is passed to subroutine DRAW through the COMMON block.

WRNR subroutine flow chart. A flow chart for subroutine WRNR is given in Figure D-8. This flow chart is very general in nature and is intended to be used only as an aid in understanding WRNR.

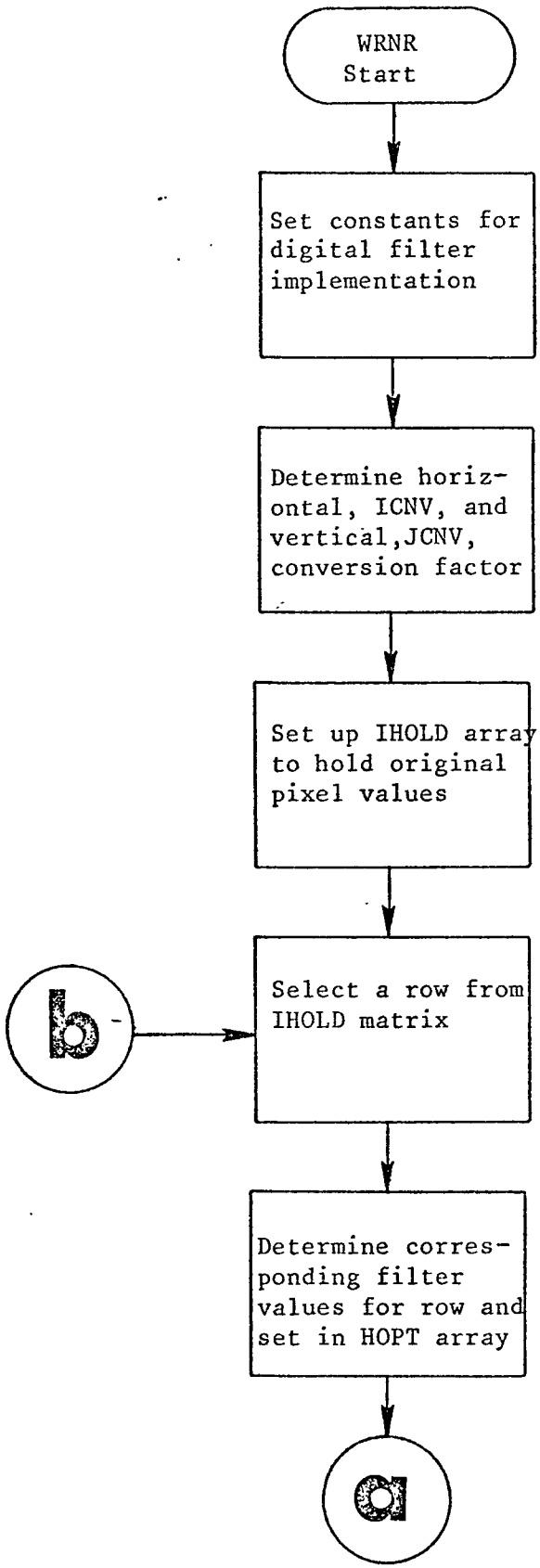
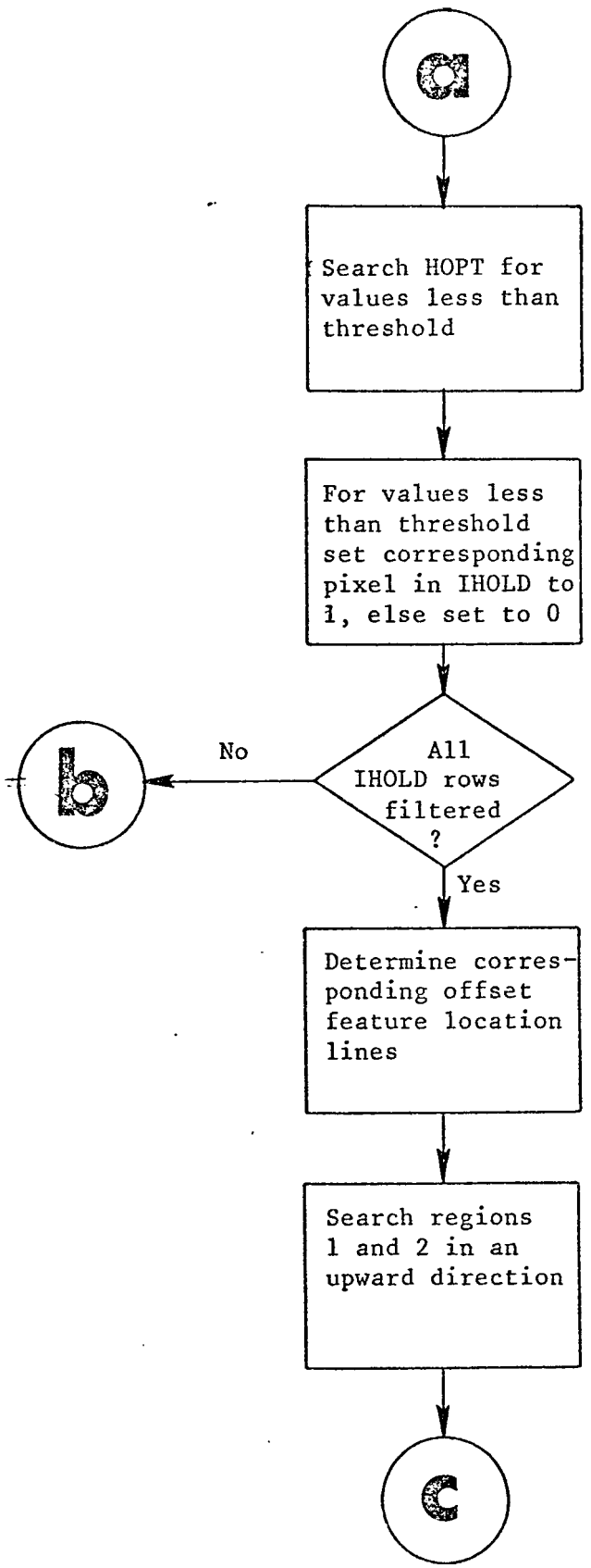
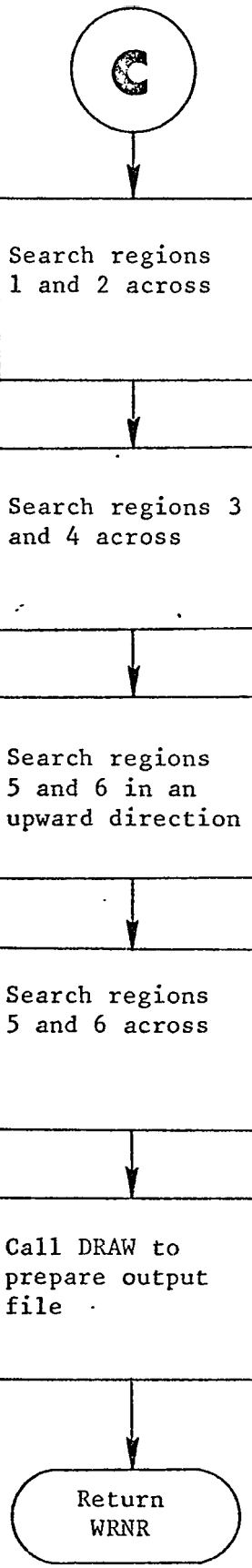


Figure D-8 WRNR Subroutine flow chart


a
Search HOPT for values less than threshold
For values less than threshold set corresponding pixel in IHOLD to 1, else set to 0
All IHOLD rows filtered ?
No
b
Yes
Determine corresponding offset feature location lines
Search regions 1 and 2 in an upward direction
c


C
Search regions 1 and 2 across
Search regions 3 and 4 across
Search regions 5 and 6 in an upward direction
Search regions 5 and 6 across
Call DRAW to prepare output file
Return WRNR

WRNR subroutine listing. This discussion of subroutine WRNR is concluded with a program listing. All subroutines called by WRNR will be discussed in later paragraphs. Figure D-9 gives a listing of subroutine WRNR.

Subroutine DRAW

The purpose of subroutine DRAW is to output the results of the feature location routine and the facial outline routine to a file on the disk. Then, by calling a library routine called IMP4, the file containing the original image, the feature locations and the facial outline can be outputted to an electrostatic printer.

DRAW subroutine technical discussion. All information for the DRAW subroutine is passed in the COMMON block. The variable names remain unchanged from the previous routines and so will not be repeated here. The feature location information is extracted from the variables in the COMMON block and the IHOLD matrix holds the facial outline information.

The first function of subroutine DRAW is to calculate the horizontal and vertical conversion factors, ICNV and JCNV, respectively. These factors will be used later in the program for transferring the outline data contained in IHOLD into the output file.

The next function is to write the vertical lines into the output file. There are three lines that will be written by this section, IRF, NL, and LF, The first time through this DO loop will write the left side of the face line into the J=LF sector. Figure D-10 illustrates this case. The working array IM is first set to all

```
001   FTN
002         SUBROUTINE WRNR
003         COMMON IHOLD(128,70),NAME(3),IM(128)
004         COMMON LF,IRF,NL,ITH,IBH,IHAR,ML,LE,NT,LEB,MLL,MLR
005   C      LINDA BROMLEY
006         DIMENSION HOPT(70)
007   C    SET CONSTANTS FOR WIENER FILTER
008         THRESH=.11
009         A=.0088
010         A11=.0102
011         A10=-.4815
012         B11=-.7556
013         ICNV=-ITH+1
014         JCNV=-LF+11
015   C    SET UP IHOLD ARRAY SO THAT ROW 1 CORRESPONDS TO ITH
016   C       AND COLUMN 1 CORRESPONDS TO LF-10
017         JC=LF-10
018         DO 20 J=JC,JC+69
019         CALL EXEC(14,103B,IM,128,NAME,J)
020         DO 21 I=ITH,ITH+127
021         IHOLD(I-ITH+1,J-JC+1)=IM(I)
022      21 CONTINUE
023      20 CONTINUE
024   C    BEGIN IMPLEMENTATION OF DIGITAL FILTER.
025         DO 1 IROW=1,127
026         HOPT(1)=0.
027         DO 2 JCOL=2,70
028         Z1M=FLOAT(IHOLD(IROW,JCOL-1))
029         Z2M=FLOAT(IHOLD(IROW+1,JCOL))
030         ZPZ=Z1M+Z2M
031         ZZ=Z1M*Z2M
032         HOLD=A*(1.-.5*(B11+1.)*ZPZ+B11*ZZ)
033         HOPT(JCOL)=HOLD/(1.+A10*ZPZ+A11*ZZ)
034      2 CONTINUE
035   C******IF FILTERED VALUE IS ABOVE THRESHOLD REPLACE ZEROED
036   C           ROW ELEMENT IN IHOLD WITH A 1
037         IHOLD(IROW,1)=0
038         DO 3 L=2,70
039         IHOLD(IROW,L)=0
040         IF(HOPT(L).LE.THRESH) IHOLD(IROW,L)=1
041      3 CONTINUE
042      1 CONTINUE
043   C******DETERMINE CORRESPONDING OFFSET FEATURE LOCATION LINES
044         NIHAR=IHAR+ICNV
045         NLEB=LEB+ICNV
046         NML=ML+ICNV
047         NMLL=MLL+JCNV-5
048         NMLR=MLR+JCNV+5
049         NIBH=IBH+ICNV
050         NITH=ITH+ICNV
051         NLF=LF+JCNV
052         NNL=NL+JCNV
053         NIRF=IRF+JCNV
054   C    BEGIN DETERMINATION OF OUTLINE
055   C***DO REGIONS 1 AND 2 UP.
056         DO 901 J=(NLF+1),(NIRF-1)
057         DO 900 I=(NLEB-5),NIHAR,-1
058         IF((I.EQ.(NLEB-5)).AND.(IHOLD(I,J).GE.1)) GO TO 901
059         IF(IHOLD(I,J).NE.1) GO TO 900
060         IHOLD(I,J)=64
061         GO TO 901
062      900 CONTINUE
063         IHOLD(NIHAR,J)=64
064      901 CONTINUE
```

Figure D-9 WRNR subroutine listing

```
C*** DO REGIONS 1 AND 2 ACROSS.
      DO 902 I=(NLEB-5),(NIHAR-1),-1
      DO 904 L=1,2
      IF(L.EQ.2) GO TO 905
      IH=NLF-5
      IDEL=-1
      GO TO 930
  905 IH=NIRF+5
      IDEL=1
  930 DO 903 J=NNL,IH,IDEL
      IF((J.EQ.NNL).AND.(IHOLD(I,J).GE.1)) GO TO 904
      IF(IHOLD(I,J).LE.0) GO TO 903
      IHOLD(I,J)=64
      IF((L.EQ.1).AND.(I.EQ.(NLEB-5))) LSTL=J
      IF((L.EQ.2).AND.(I.EQ.(NLEB-5))) LSTR=J
      GO TO 904
  903 CONTINUE
      IHOLD(I,IH)=64
  904 CONTINUE
  902 CONTINUE
C*** DO REGION 3 AND 4 ACROSS ONLY.
      DO 906 I=(NLEB-5),(NML-5)
      DO 932 L=1,2
      IF(L.EQ.2) GO TO 933
      JSTART=NNL-(NNL-NLF)/2
      JSTOP=NLF-5
      LST=LSTL
      IDEL=-1
      GO TO 934
  933 JSTART=(NIRF-NNL)/2+NNL
      JSTOP=NIRF+5
      LST=LSTR
      IDEL=1
  934 DO 907 J=JSTART,JSTOP,IDEL
      IF((J.EQ.JSTART).AND.(IHOLD(I,J).GE.1)) GO TO 931
      IF(IHOLD(I,J).LE.0) GO TO 907
      IHOLD(I,J)=64
      IF(L.EQ.1) LSTL=J
      IF(L.EQ.2) LSTR=J
      GO TO 932
  907 CONTINUE
  931 IF(L.EQ.1) IHOLD(I,LSTL)=64
      IF(L.EQ.2) IHOLD(I,LSTR)=64
  932 CONTINUE
  906 CONTINUE
```

```
110   C***DO REGIONS 5 AND 6 DOWN.
111         DO 910 J=(NLF+1),(NIRF-1)
112         DO 911 I=(NML+10),NIBH
113         IF((I.EQ.(NML+10)).AND.(IHOLD(I,J).GE.1)) GO TO 910
114         IF(IHOLD(I,J).LE.0) GO TO 911
115         IHOLD(I,J)=64
116         GO TO 910
117     911 CONTINUE
118         IHOLD(NIBH,J)=64
119     910 CONTINUE
120   C*** DO REGIONS 5 AND 6 ACROSS.
121         DO 912 I=(NML-5),NIBH
122         DO 936 L=1,2
123         IF(L.EQ.2) GO TO 937
124         JSTOP=NLF
125         LST=LSTL
126         IDLT=-1
127         GO TO 938
128     937 JSTOP=NIRF
129         LST=LSTR
130         IDLT=1
131     938 DO 913 J=NNL,JSTOP,IDLT
132         IFG=0
133         IF((J.GE.NMLL).AND.(J.LE.NMER).AND.(I.LE.(NML+5))) GO TO 913
134         IF((J.EQ.NNL).AND.(IHOLD(I,J).GE.1)) GO TO 936
135         IF(IHOLD(I,J).LE.0) GO TO 913
136         IHOLD(I,J)=64
137         IF(L.EQ.1) LSTL=J
138         IF(L.EQ.2) LSTR=J
139         GO TO 936
140     913 CONTINUE
141         IF(L.EQ.1) IHOLD(I,LSTL)=64
142         IF(L.EQ.2) IHOLD(I,LSTR)=64
143     936 CONTINUE
144     912 CONTINUE
145   C***DRAW OUTLINE.
146         CALL DRAW
147         RETURN
148         END
149         END$
*** LIST END ****
```

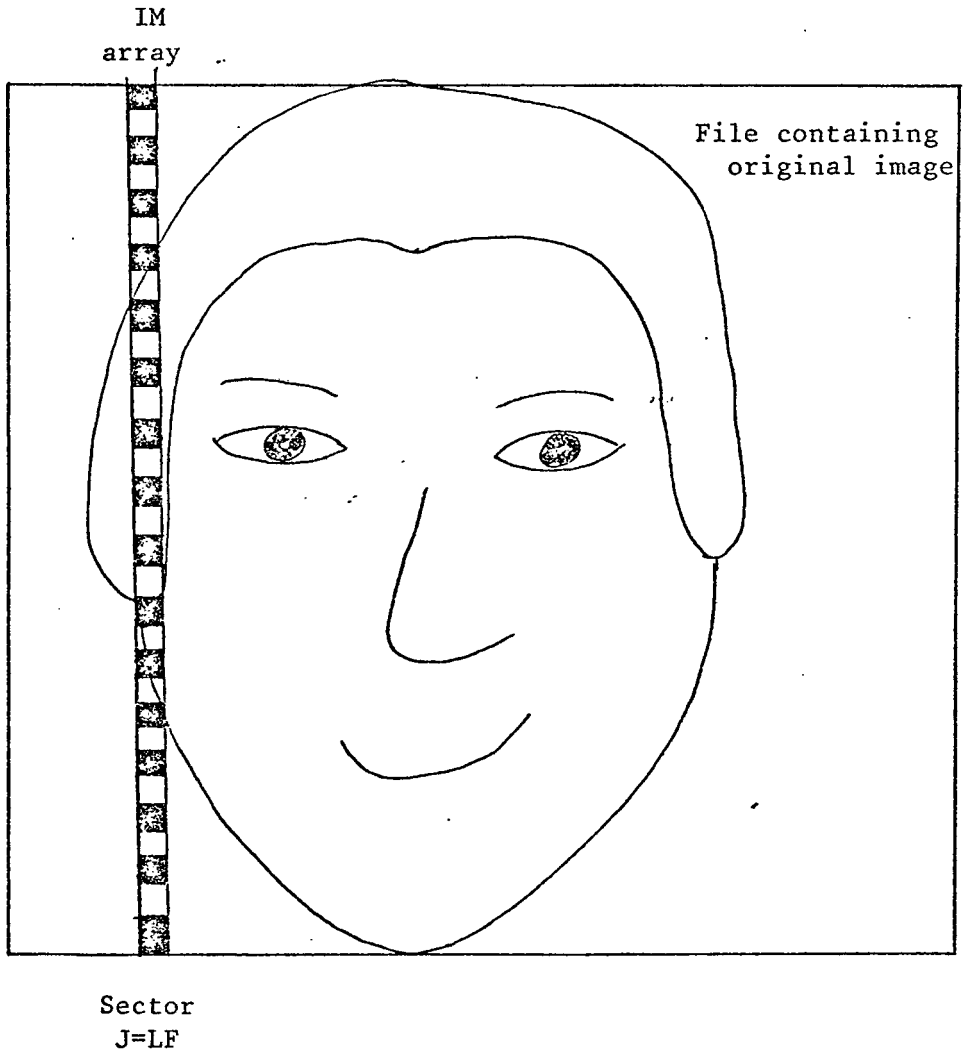


Figure D-10 Example of writing a vertical line using EXEC

zeroes. Then, every other element in the array is filled with a gray scale value of 63 so that when displayed on the electrostatic printer, the line will appear to be dashed resulting in a greater clarity. After filling the IM array, it is written into sector J=LF of the output file by a call to EXEC. The loop is closed by selecting in turn the two remaining vertical lines and writing them into the output file.

The six horizontal lines are next written into the output file. This process is slightly more complex than writing vertical lines due to the writing technique used by EXEC. EXEC writes only columns of data and not rows of data. In the case of horizontal lines, the IM array will have to be filled so that the elements corresponding to the locations of the lines are forced to be either a 0 or 63, while keeping all others to be whatever was read from the original image. Also, since this process will have to be performed when the outline is to be written, the coding to write the outline is contained within this loop. Figure D-11 illustrates this problem, showing the IM array elements for an arbitrary sector J.

Since dashed lines are desired and since all line elements in the IM array will all be either 0 or 63 depending upon whether the last sector output for that line was a 63 or a 0, two holding variable, IF1 and IF2, are defined. A DO loop in J is then entered which will move use sequentially from one sector to the next. A call to EXEC will fill the array IM with the original picture (including the vertical lines previously written). The elements in IM corresponding to the six desired horizontal lines are then set to be the current value of IF1. The values of IF1 and IF2 are then switched so that the next time through the loop the line elments in IM will be the opposite color.

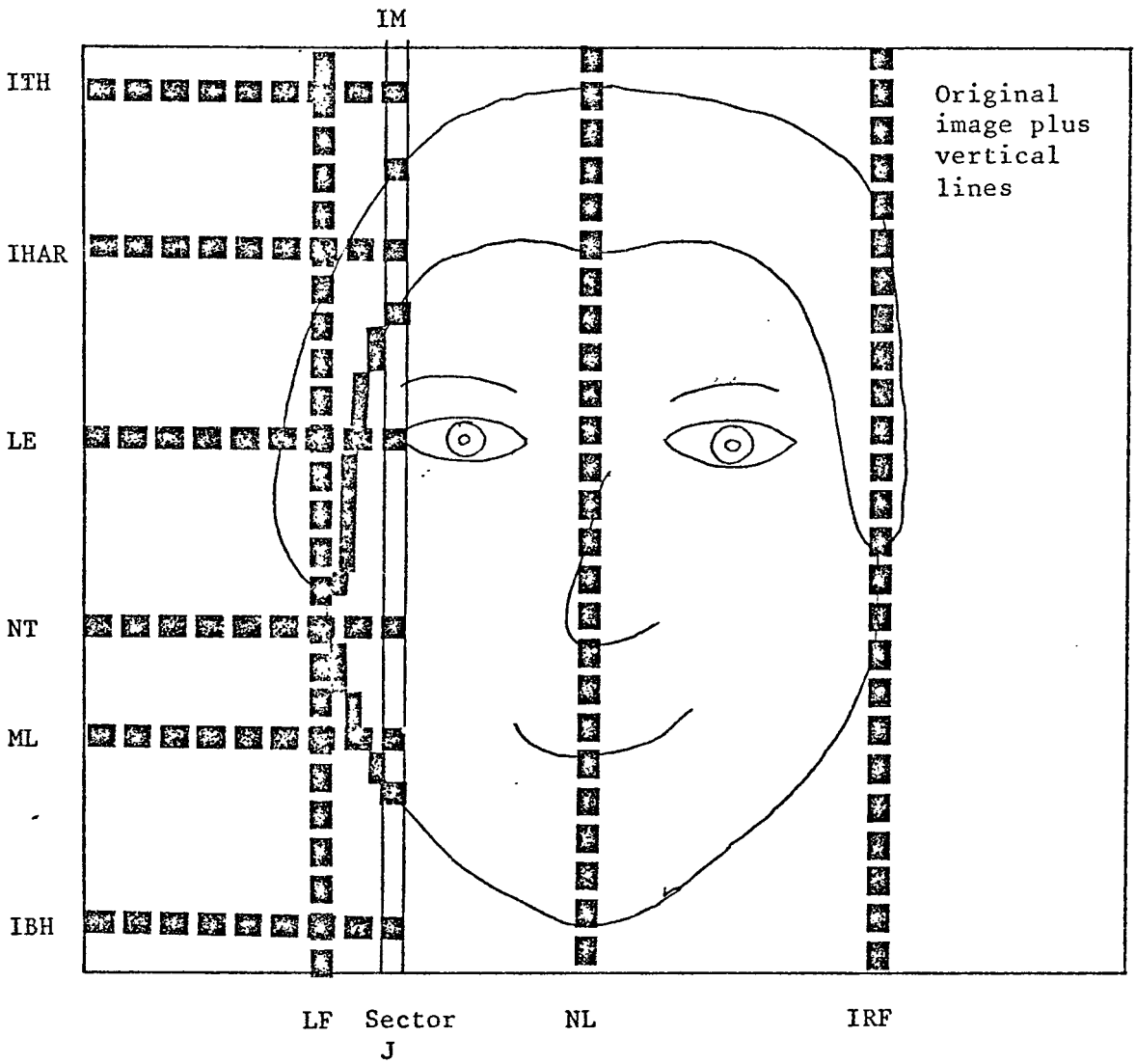


Figure D-11 Example of writing horizontal lines and outline

It is then determined if there are any outline points in this J sector. This is done by looking at the corresponding column in the IHOLD array. Since the columns in the IHOLD array are offset from the column in the original file by the JCNV factor, a new variable JN is defined to be the corresponding offset column in IHOLD. If JN is not between 1 and 70, the IHOLD matrix bounds, then there is no possibility of having an outline point and so control is skipped to the call to EXEC where the column is written into the output file.

If, however, JN is between 1 and 70, there is an outline point contained within the column and so a DO loop is entered in which the 128 column elements of IHOLD is searched. Once again, since the rows of IHOLD may be offset from the rows of the original file, a new variable, IN, defined to be the corresponding offset row in IHOLD. If IN is within the IHOLD row bounds, the matrix element located in the IN row and JN column is checked. If this value is greater than 60, indicating an outline point, the corresponding row element in the IM array is set to 63 to produce a bright dot. The loop is closed by similarly checking each row element in the given column of IHOLD for an outline point. After all elements have been checked and written accordingly into the IM array, a call to EXEC writes the array into the output file. The major loop in J is closed by selecting a new sector.

After the output file has been written, DRAW outputs a message to the CRT screen giving the operator the results of the feature location search. Figure D-12 is an example of this output.

```
THE FOLLOWING FEATURE LOCATIONS HAVE BEEN
     DETERMINED FOR FILE ???

  HORIZONTAL LINES                VERTICAL LINES
********************          **********************

TOP OF HEAD=    5                LEFT SIDE OF FACE= 41
HAIRLINE= 26                     NOSELINE= 63
EYELINE= 59                      RIGHT SIDE OF FACE=  86
NOSETIP= 76
MOUTHLINE=  97
BOTTOM OF HEAD= 127
```

Figure D-12 Example of a typical CRT message

DRAW subroutine flow chart. A flow chart for subroutine DRAW is given in Figure D-13. This flow chart is very general in nature and is intended to be used only as an aid in understanding DRAW.

DRAW subroutine listing. This discussion of subroutine DRAW is concluded with a program listing. Figure D-14 gives a listing of subroutine DRAW.

Subroutine SIGTR

The SIGTR subroutine calculates row or column signatures through calls to EXEC. The number of rows or columns in the signature and the range over which the signature is to be taken are set in the argument list.

SIGTR subroutine technical description. The variables used in the SIGTR argument list and in the subsequent program are defined as follows:

NAME-Image file name as given to mainprogram

IM -A 128 element array used to hold a column of pixel values as provided by EXEC

NCS -Column number where the signature is to start

NCE -Column number where the signature is to end

NRS -Row number where the signature is to start

NRE -Row number where the signature is to end

ISRS-Indicates what type of signature is to be performed. If a one then a row signature is to be taken. Otherwise, a column signature is assummed.

ISIG-An array provided by the mainprogram to hold the resulting signature

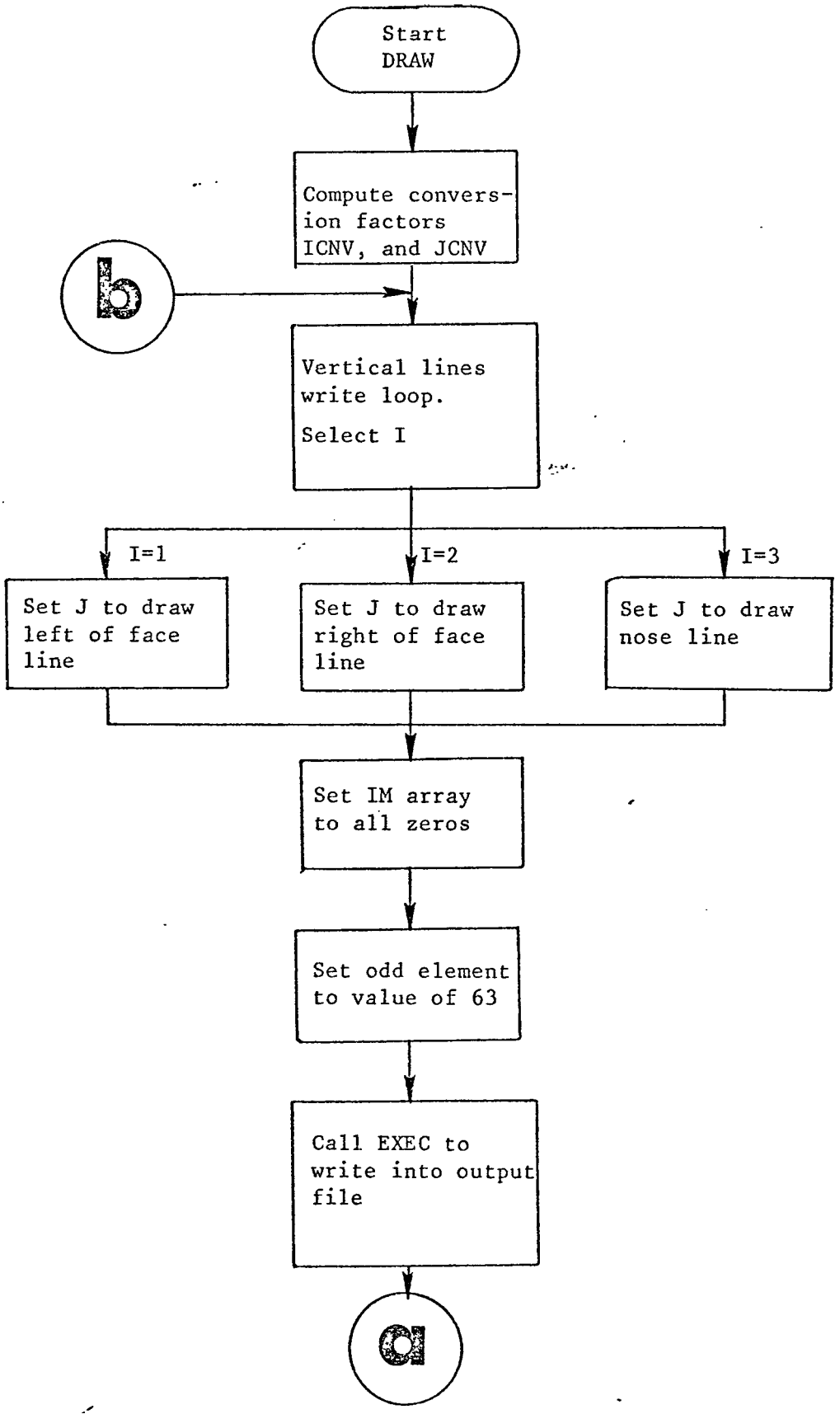
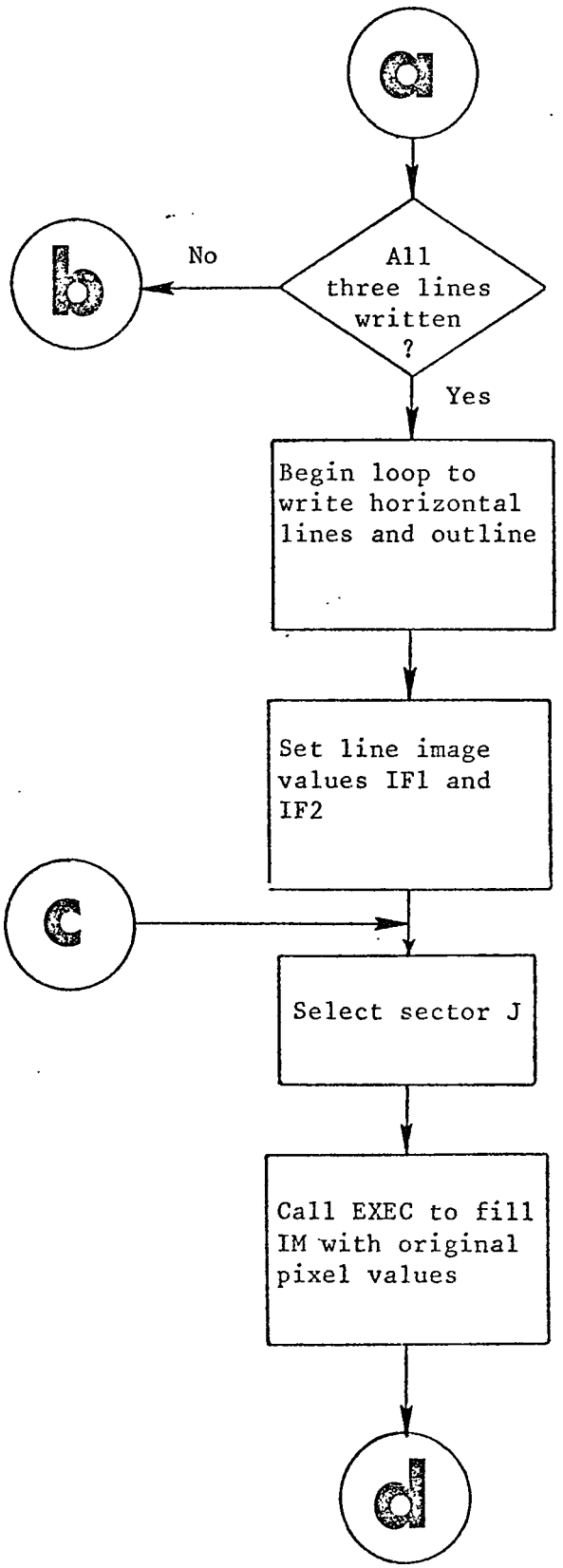
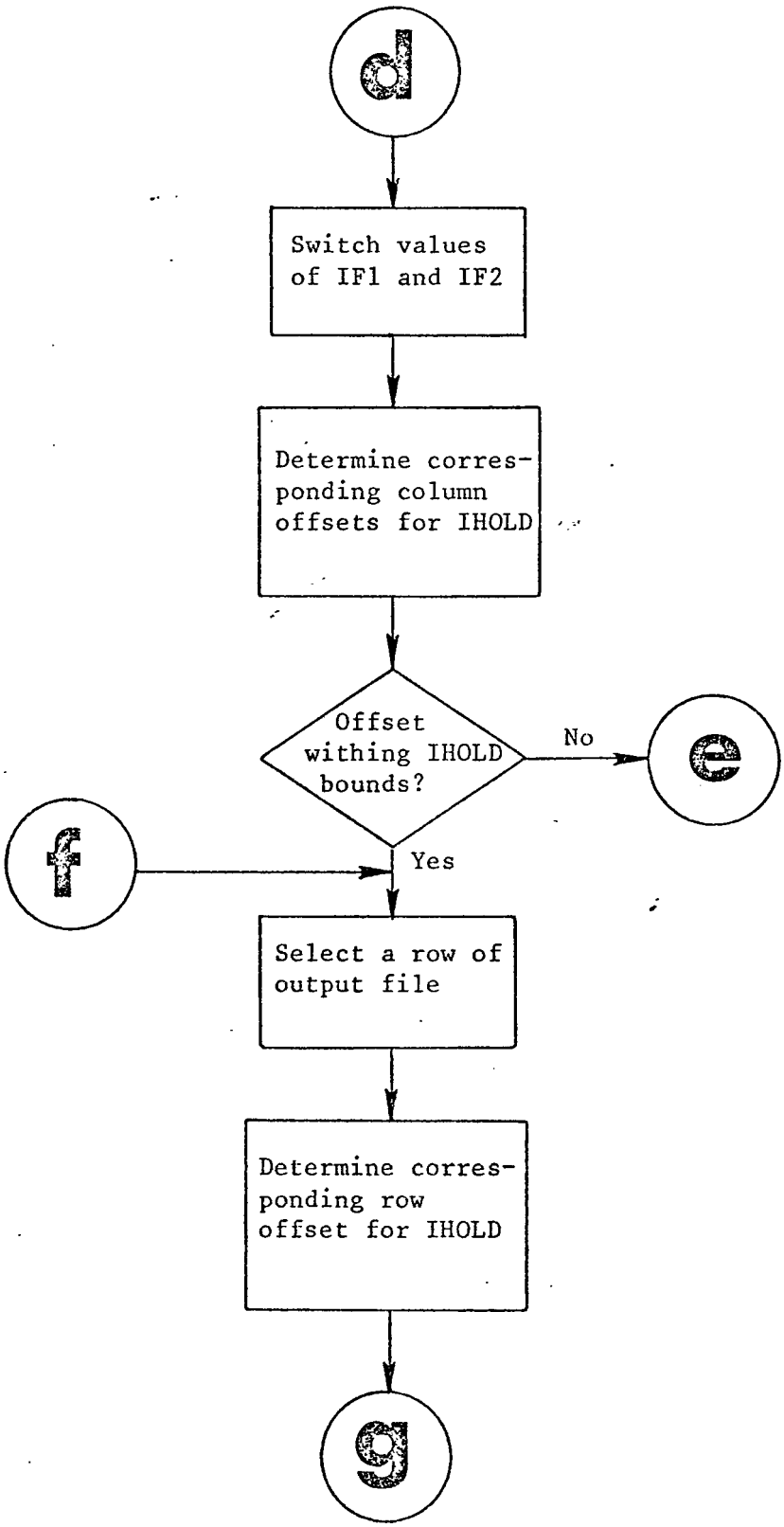
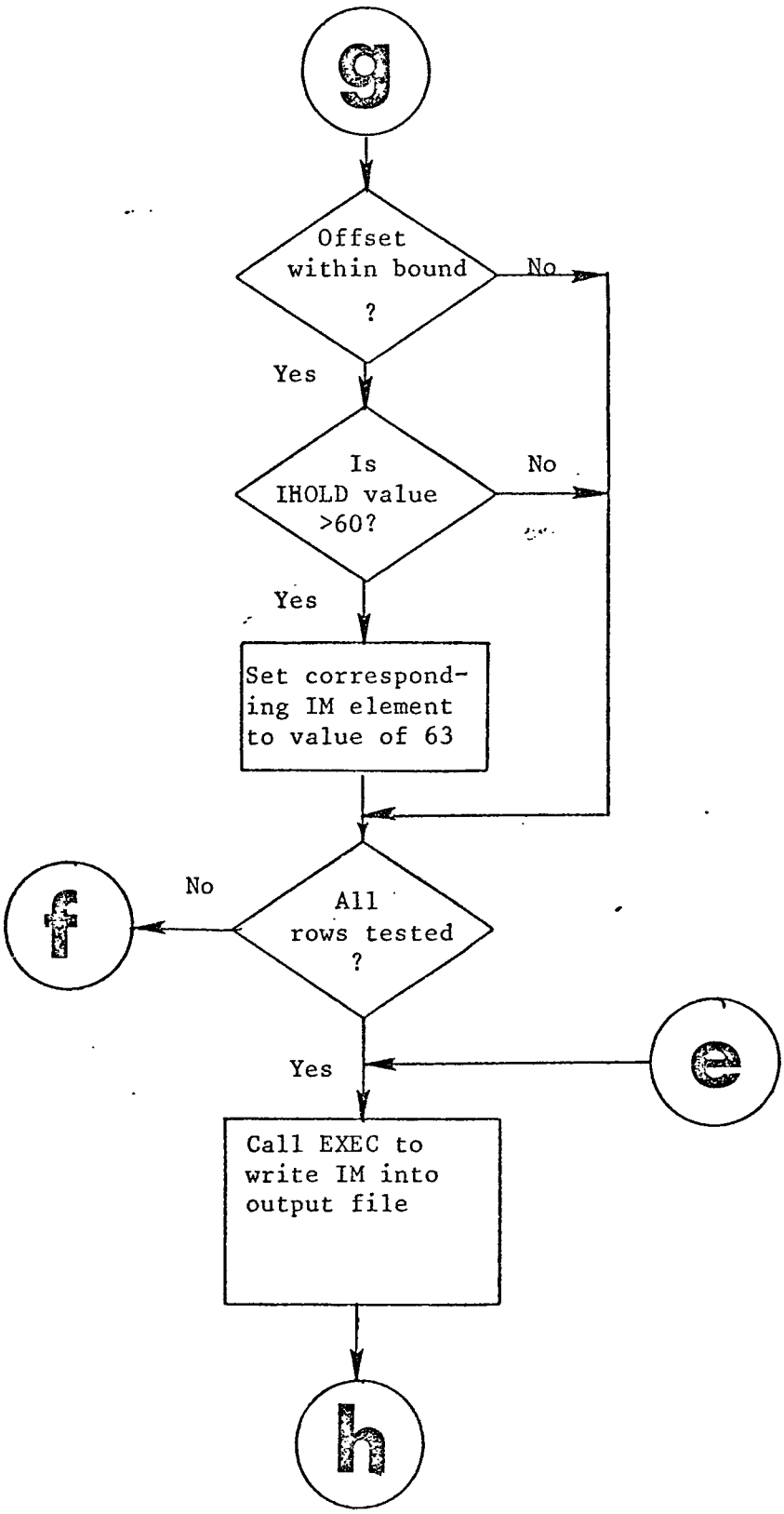
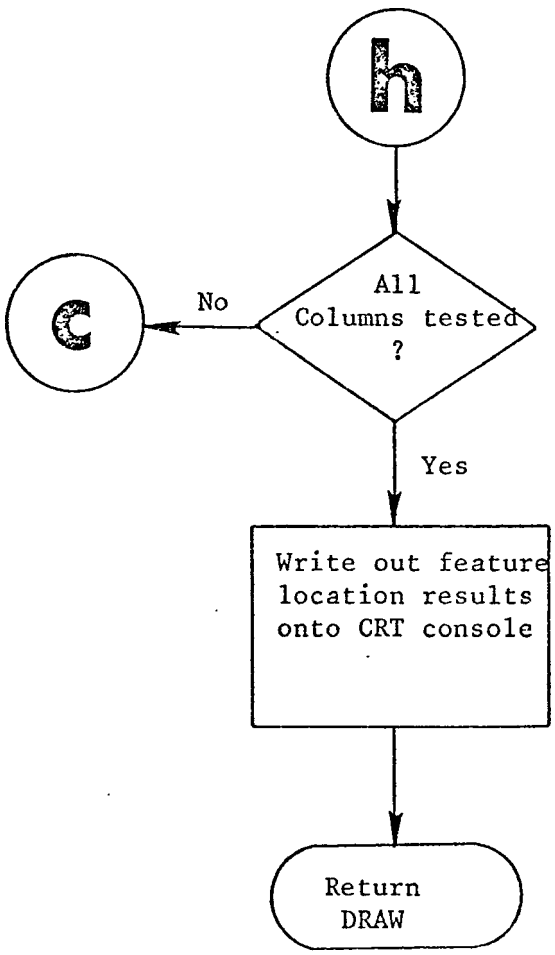


Figure D-13 DRAW subroutine flow chart


a
All three lines written ?
No
b
Yes
Begin loop to write horizontal lines and outline
Set line image values IF1 and IF2
c
Select sector J
Call EXEC to fill IM with original pixel values
d


d
Switch values of IF1 and IF2
Determine corresponding column offsets for IHOLD
Offset withing IHOLD bounds?
No
e
f
Yes
Select a row of output file
Determine corresponding row offset for IHOLD
g


g
Offset within bound ?
No
Yes
Is IHOLD value >60?
No
Yes
Set corresponding IM element to value of 63
f
No
All rows tested ?
e
Yes
Call EXEC to write IM into output file
h


h
All
Columns tested
?
No
c
Yes
Write out feature
location results
onto CRT console
Return
DRAW

```
001   FTN
002         SUBROUTINE DRAW
003   C      L. BROMLEY
004         COMMON IHOLD(128,70),NAME(3),IM(128)
005         COMMON LF,IRF,NL,ITH,IBH,IHAR,ML,LE,NT,LEB,MLL,MLR
006         ICNV=-ITH+1
007         JCNV=-LF+11
008   C*****WRITE VERTICAL LINES INTO NAME.
009         DO 1 I=1,3
010         IF(I.EQ.1) J=LF
011         IF(I.EQ.2) J=IRF
012         IF(I.EQ.3) J=NL
013         DO 2 L=1,128
014         IM(L)=0
015       2 CONTINUE
016         DO 3 L=1,128,2
017         IM(L)=63
018       3 CONTINUE
019         CALL EXEC(15,103B,IM,128,NAME,J)
020       1 CONTINUE
021   C*****WRITE HORIZONTAL LINES INTO NAME.
022         IF1=0
023         IF2=63
024         DO 4 J=1,128
025         CALL EXEC(14,103B,IM,128,NAME,J)
026         IM(ITH)=IF1
027         IM(IBH)=IF1
028         IM(IHAR)=IF1
029         IM(ML)=IF1
030         IM(LE)=IF1
031         IM(NT)=IF1
032         HOLD=IF1
033         IF1=IF2
034         IF2=HOLD
035   C*****WRITE OUTLINE INTO NAME.
036         JN=J+JCNV
037         IF((JN.LT.1).OR.(JN.GT.70)) GO TO 8
038         DO 5 LL=1,128
039         IN=LL+ICNV
040         IF((IN.LT.1).OR.(IN.GT.120)) GO TO 5
041         IF(IHOLD(IN,JN).GT.60) IM(LL)=63
042       5 CONTINUE
043       8 CALL EXEC(15,103B,IM,128,NAME,J)
044       4 CONTINUE
045         WRITE(6,6) (NAME(I),I=1,3)
046       6 FORMAT(///," THE FOLLOWING FEATURE LOCATIONS HAVE BEEN",/,
047        1 "       DETERMINED FOR FILE ",2A2,A1,///)
048         WRITE(6,7) ITH,LF,IHAR,NL,LE,IRF,NT,ML,IBH
049       7 FORMAT(3X,"HORIZONTAL LINES",14X,"VERTICAL LINES",/,1X,
050        1 20("*"),10X,22("*"),//," TOP OF HEAD= ",I3,15X,
051        5 "LEFT SIDE OF FACE= ",I2,
052        2 /," HAIRLINE= ",I2,19X,"NOSELINE= ",I2,/,"EYELINE= ",I2,20X,
053        3 "RIGHT SIDE OF FACE= ",I3,/," NOSETIP= ",I2,/," MOUTHLINE= ",I3,
054        4 /," BOTTOM OF HEAD= ",I3,///)
055         RETURN
056         END$
*** LIST END ****
```

Figure D-14 DRAW subroutine listing

NR-Number of rows in signature

NC-Number of columns in signature

The first action of sumbroutine SIGTR is to calculate the number of rows, NR, and number of columns, NC, being used in the signature. If ISRS is a one, the computer begins calculating a row signature by summing rows NRS to NRE for each column from NCS to NCE.

This is accomplished by first choosing a column within the given range and calling EXEC for that column. The pixel values in the given co column are returned in the IM array. These values are summed from rows NRS to NRE. The normalized signature result for the given row is found by dividing this resulting sum by the number of rows, NR, that were summed. A new column is chosen and the process is repeated until the entire range of columns have been addressed. The ISIG array holding the results of these sums is returned to the mainprogram via the argument list.

If, however, ISRS is not one, the computer skips to the section of the program that calculates column signatures by first clearing out the ISIG array. A column is chosen from the range of NCS to NCE. A call to EXEC places the pixel values for the chosen column into the IM array. Since a summation of columns is desired, the elments of the IM array are summed to corresponding elements in the ISIG array between the rows of NRS to NRE. A new column is chosen and the process is repeated until all desired columns are added to the ISIG array. The ISIG array is normalized by dividing by the number of columns that were summed, NC, before it is passed back to the mainprogram.

SIGTR subroutine flow chart. A flow chart for subroutine SIGTR is presented in Figure D-15. This flow chart is very general in nature and is intended only for an aid in understanding subroutine SIGTR.

SIGTR subroutine listing. This discussion of subroutine SIGTR is concluded with a program listing. Figure D-16 gives a listing of subroutine SIGTR.

Subroutine SCOPE

Subroutine SCOPE is a library subroutine that was already available on the disk. Subroutine SCOPE is used to display images, graphs, lines, or marks on the storage scope for viewing. Although the SCOPE subroutine is not used in the final program, it was used extensively in the development of the program. A very brief discussion of the subroutine is therefore presented.

Useage of subroutine SCOPE. There are three variables in the SCOPE subroutine argument list. The first is the x-axis displacement of the element to be illuminated on the storage scope. The second variable is the y-axis displacement of the element to be illuminated. The third variable represents the intensity to which the element is to be illuminated. Therefore, one call to SCOPE results in the illumination of one element on the storage scope.

For example, to display an image contained in file NAME in the upper left hand quadrant, the program coding would be:

```
      DO 20 J=1,128
      CALL EXEC(14,103B,IM,128,NAME,J)
      DO 3 I=1,128
      CALL SCOPE(J,256-I,4*IM(I))
    3 CONTINUE
   20 CONTINUE
```

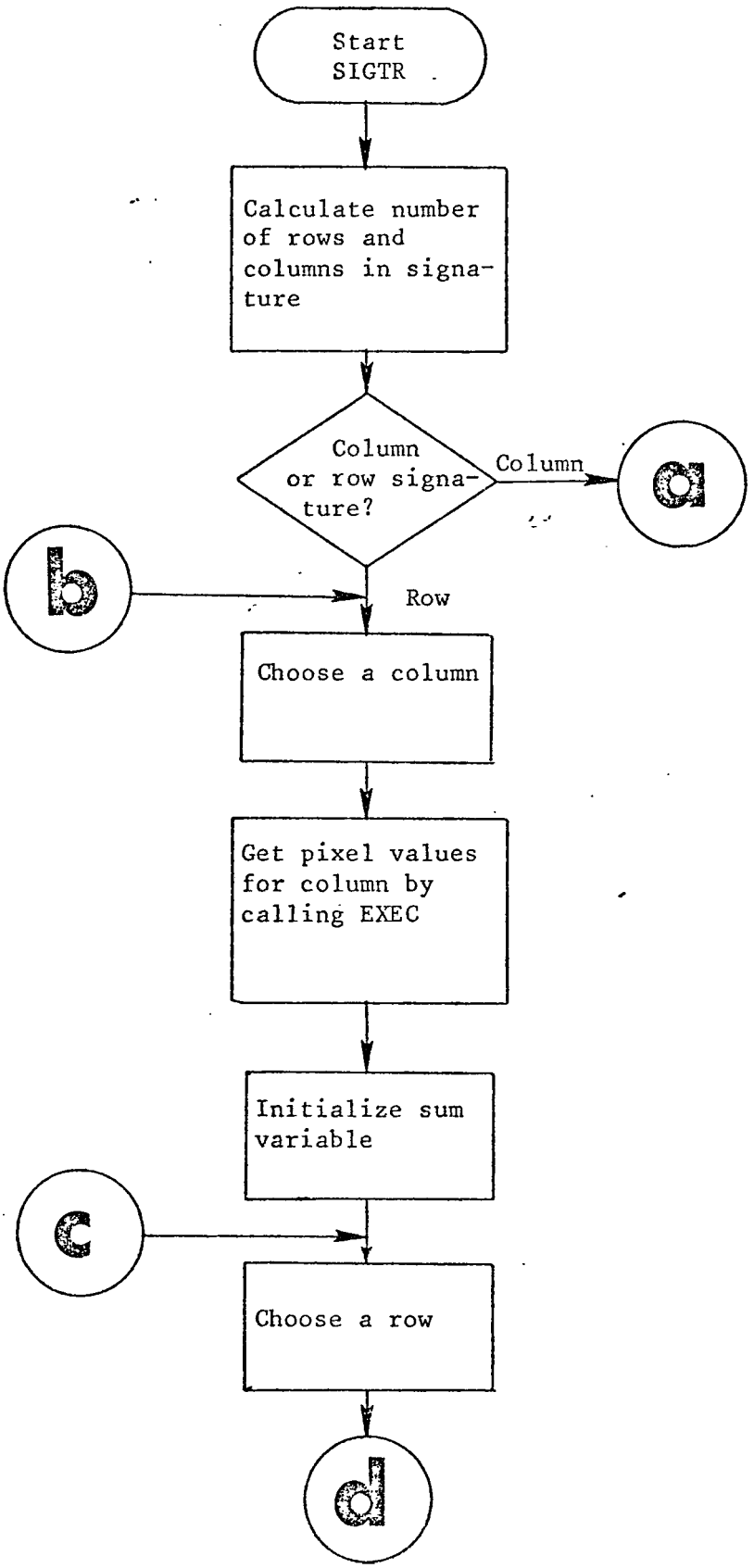
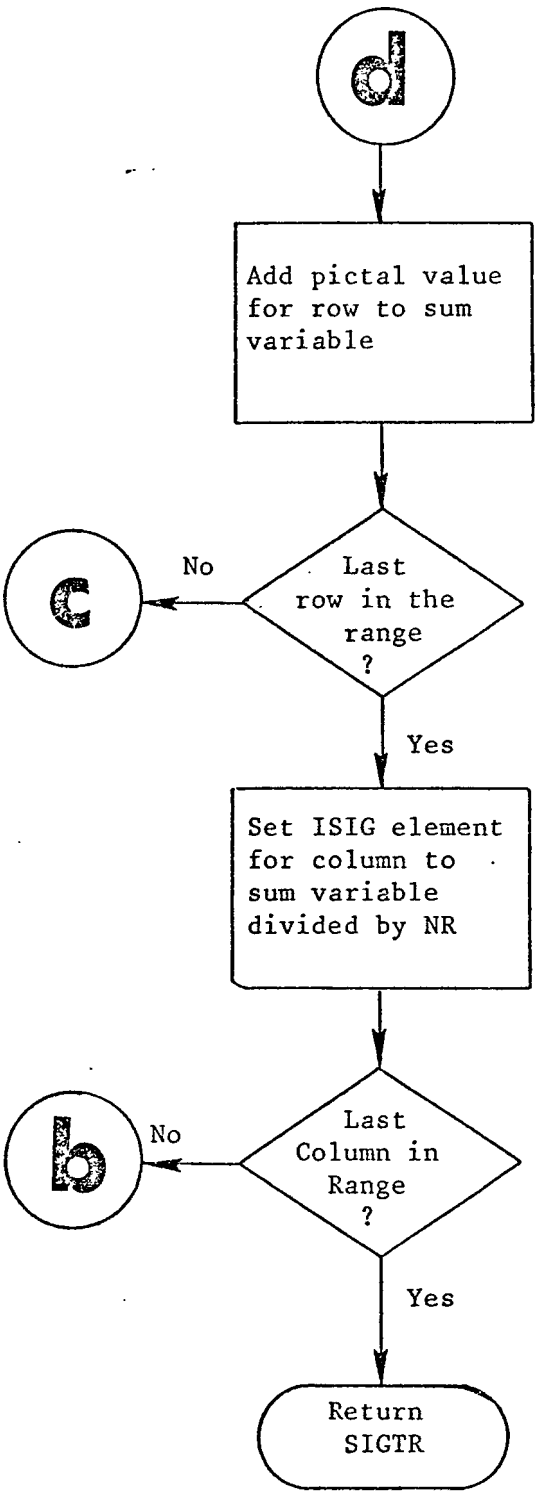
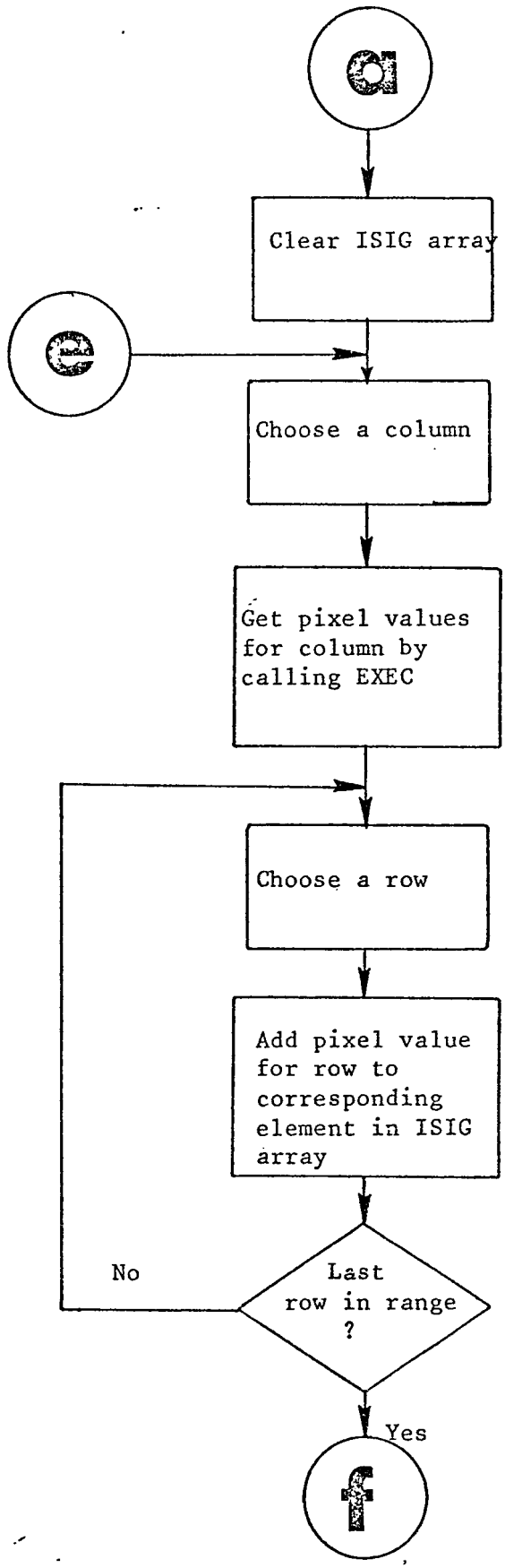
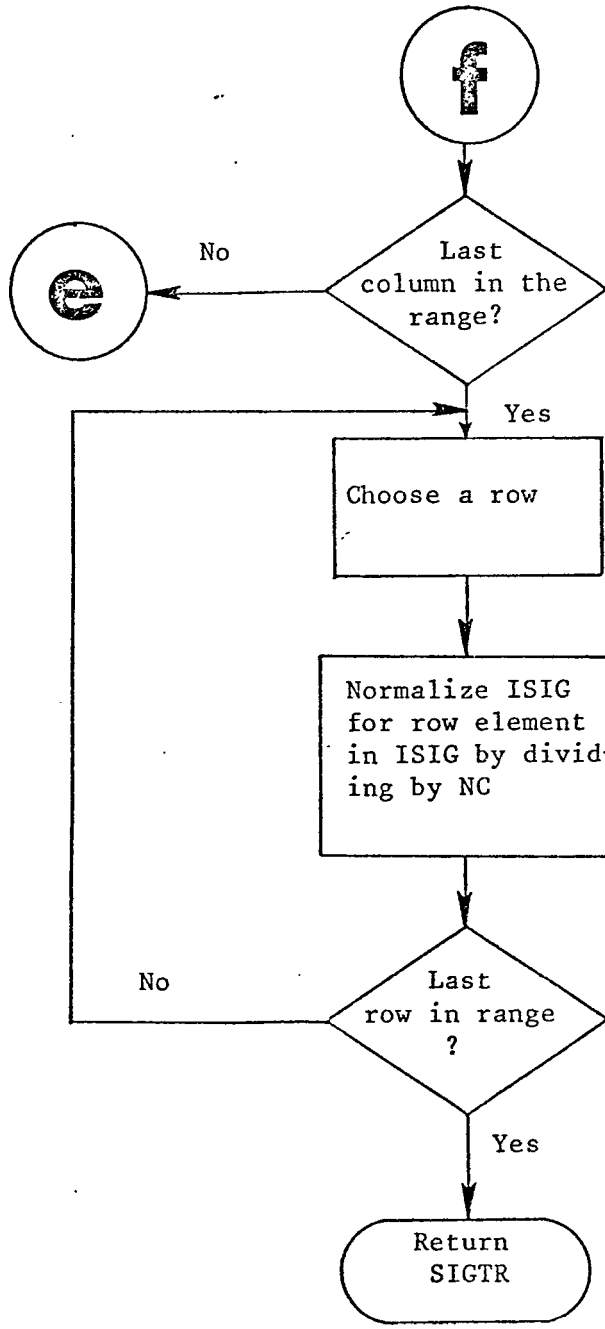


Figure D-15 SIGTR subroutine flow chart


d
Add pictal value for row to sum variable
Last row in the range ?
No
c
Yes
Set ISIG element for column to sum variable divided by NR
Last Column in Range ?
No
b
Yes
Return SIGTR


d
Clear ISIG array
e
Choose a column
Get pixel values for column by calling EXEC
Choose a row
Add pixel value for row to corresponding element in ISIG array
No
Last row in range ?
Yes
f


f
Last column in the range?
No
e
Yes
Choose a row
Normalize ISIG for row element in ISIG by dividing by NC
Last row in range ?
No
Yes
Return SIGTR

```
001   FTN
002   C       LINDA BROMLEY
003           SUBROUTINE SIGTR(NAME,IM,NCS,NCE,NRS,NRE,ISRS,ISIG)
004           DIMENSION NAME(1),IM(1),ISIG(1)
005           NR=NRE-NRS+1
006           NC=NCE-NCS+1
007           IF(ISRS.NE.1) GO TO 1
008           DO 10 J=NCS,NCE
009           CALL EXEC(14,103B,IM,128,NAME,J)
010           ISUM=0
011           DO 11 I=NRS,NRE
012        11 ISUM=ISUM+IM(I)
013           ISIG(J)=ISUM/NR
014        10 CONTINUE
015           GO TO 150
016         1 DO 12 I=NRS,NRE
017        12 ISIG(I)=0
018           DO 13 J=NCS,NCE
019           CALL EXEC(14,103B,IM,128,NAME,J)
020           DO 14 I=NRS,NRE
021        14 ISIG(I)=ISIG(I)+IM(I)
022        13 CONTINUE
023           DO 15 I=NRS,NRE
024        15 ISIG(I)=ISIG(I)/NC
025       150 RETURN
026           END
027           END$
*** LIST END ****
```

Figure D-16 SIGTR subroutine listing

Notice that a call to EXEC retrieves a column of pixel values of the image, where the first element in the array corresponds to the uppermost element of the image. So, the image will be drawn from the top down and from left to right. It is also important to notice the use of the term 256-I for the y displacement variable. This is required because although the first element of the array returned by EXEC corresponds to the upper left corner, the origin for the y displacement on the storage scope resides at the lower left corner. For example, say I=1 and J=1 in the coding above. The call to EXEC would then produce the first column of values, with the first element representing the upper left pixel of the image. The upper left element of the storage scope is defined by the coordinates (0,255). Therefore the y displacement relative to the pixel array is 256-I.

Since the maximum pixel value is 64, the intensity level for each pixel is multiplied by four so that the maximum intensity level of 255 for the storage scope can be obtained.

A horizontal line . be similarly drawn by using SCOPE in a DO loop such that the y displacement remains the same but the x displacement is varied over the desired range. A vertical line can likewise be drawn by varying only the y displacement. Whenever any type of line or plot is drawn, the intensity level is set to 255 for the maximum illumination.

Subroutine EXEC

This discussion of the CAPTURE Algorithm routines is concluded with a very brief description of library routine EXEC. Only its useage as applicable to the CAPTURE Algorithm is discussed.

EXEC routine useage. There are six variables contained in the EXEC routine argument list. The first determines whether a read (14) or write (15) function is desired. The second variable describes what type of data is to be read or written. The third argument is an array name. If a read function is called for, this array will contain the information that was read from the disk file. If a write function is desired, this array holds the information that is to be written onto the file. Thr fourth argument is a number representing the number of elements in the previously described array. The fifth argument is an array containing the name of the file to be read from or onto which the data is to be written. The last variable is a number representing the "sector" or column of the file that the data is to be read from or written into. The files used in the CAPTURE Algorithm are the digitized form of the photographs or sketches. These files are therefore a matrix of 128 rows and 128 columns. One call to EXEC will produce one column of pixel values as determined by the sixth argument number. The orientation of the resulting output array is such that element one represents the pixel at the top of the image.

For example, a call to EXEC of the form

```
CALL EXEC(15,103B,IM,128,NAME,J)
```

means that it is desired to write (15) into a binary file contained on disk (103B) the information contained in a 128 element array IM into column J of the file called NAME.