MORPHOLOGICAL CODING OF IMAGES

by

ALI D. S. ALI

S.B. Massachusetts Institute of Technology
(1978)

S.M. Massachusetts Institute of Technology
(1979)

E.E. Massachusetts Institute of Technology
(1980)

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements of the Degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1984

Signature of Author .....
Department of Electrical Engineering and Computer Science, August 1984

Certified by .....
Thesis Supervisor

Accepted by .....
Chairman, Departmental Committee on Graduate Students

MORPHOLOGICAL CODING OF IMAGES

by ALI D. S. ALI

Submitted to the department of Electrical Engineering and Computer Science on August 24, 1984, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

ABSTRACT

The potential for the use of morphological coding in transmitting head-and-shoulder images typically encountered in video teleconferencing applications is developed and evaluated in this thesis. Morphological coding is defined here as the reduction of an image to a linear superposition of basic morphemes each characterized by a coded position, intensity, and type; the morphemes may overlap spatially, may be non-orthogonal, and may be one or multi-dimensional.

The development of morphological coding here involves 1) design and implementation of optimum filters for morpheme detection and estimation, 2) morphological coding and reconstruction of one-dimensional images such as scan lines of a horizontal raster, and 3) grouping of one-dimensional morphemes in adjacent scan lines of two-dimensional images into coded "stroke" features similar to the brush strokes in an artist's painting. Procedures for achieving the above are presented and the results are evaluated using psychovisual considerations together with rms-error criteria. Causes limiting quality and bit-rate reduction in morphologically coded images are studied in detail.

Morphological coding is then compared to other methods of image coding, with emphasis on the exclusive features of morphological coding. Finally, a view in a different perspective is offered for morphological coding. It is proposed that the notion of the "morpheme" be associated with a local measure of the signal characteristic rather than with a set of basis functions.

The capability of the algorithm implemented in this thesis to reduce an image to strokes is epitomized in a 128x128 pel image which was reconstructed at a bit rate of 0.5 bit/pel and signal-to-noise ratio of 23 dB. The limitation of this method of morphological coding to produce higher quality images at lower bandwidths is traced to three fundamental issues: The choice of morphemes, the practical considerations in algorithm implementation, and the limitation of the use of simple functions to reconstruct arbitrarily complex structures. Suggestions are made to further investigate the role of each in attempting to improve the performance of morphological coding.

Thesis Supervisor: David H. Staelin, Prof. of Electrical Engineering

ACKNOWLEDGEMENTS

First and foremost, I acknowledge the invaluable supervision and help granted me by my supervisor, Professor David H. Staelin. He showed me the correct principles to follow in order to carry on a research effort and bring it to a presentable conclusion. I also thank Dr. Phillip Rosenkranz for the numerous occasions on which he patiently answered my questions.

My sincere thanks to my colleague Alain Briancon for always coming through for me at times of need. Special thanks are also due to two of my best friends : Jeff Bernstein and Brian Hinman, especially for their help in the work associated with chapter 4. For their help, I would also like to thank Mark Colavita and Philippe Cassereau.

My thanks to Kathy McCue for helping me with the typing.

Last but not least, my sincere gratitude to Jack Barrett and Cosmo Papa for all the help with which they provided me for the past seven years.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background and Definition

Reducing the amount of communicated information, and hence compressing channel bandwidth, in image transmission is becoming exceedingly important because of the increasing cost of networks employing image transmission. The importance of bandwidth compression can be appreciated by considering the problems of spectrum crowding and limited availability of wideband networks.

Several coding methods for reducing channel bit rate, such as transform coding [1-3], two band coding [4-6], predictive coding [7-9], segmentation coding [10-12], etc., have been intensely studied during the past three decades. Such methods invariably incorporate efficient intermediate representations of various types which can be communicated with fewer bits of information than is required for the original, digitized, image. The novelty of the method presented here is in using a new type of very efficient intermediate representation. This method is "Morphological Coding".

We define morphological coding as the reduction of an image to a linear superposition of basic morphemes each

characterized by a coded position, amplitude, and type; the morphemes may overlap spatially, be non-orthogonal, and have one or more dimensions. A morpheme has one of a number of possible pre-selected spatial intensity distributions (types) that generally are simple in form and occupy only a portion of the entire image.

Morphological coding differs from most other coding schemes, such as those mentioned above and as indicated in recent review literature [13-16]. The principal difference is that the location of each morpheme within an image is specified independently from its type, whereas in other representations position and type are generally unified or at least restricted in their allowed combinations. For example, in transform coding the position of specific spectral component is fixed, or restricted, and in two-band or predictive coding the positions of filtered pixel values are implicit in their sequential ordering. In segmentation coding the positions of coded segments may be predetermined, and generally are not allowed to overlap.

In addition to the principal difference mentioned above, other fundamental and procedural differences between morphological coding and the more traditional methods of coding mentioned above are the following:

1) Most coding techniques which reduce the image into

a set of constituent basis functions impose the requirement that such basis functions be orthogonal. Transform coding is the classical example of this condition. Morphological coding, on the other hand, does not restrict the basis functions, i.e. the morphemes, in this manner. While this property of morphological coding usually leads to intersymbol interference problems in the morpheme detection and estimation process, it offers a desirable flexibility in coding images, especially at low bandwidths (see section 6.1).

2) As implemented in this thesis, the morpheme detection process fails to capture the low-frequency structure (e.g. ramps, DC) of the image. This necessitates the use of a separate low-frequency estimator (see section 2.3). Such employment of a separate estimator to capture the low frequencies of the image presents a clear analogy with the two-band image coding techniques. The similarity between the two methods, however, ceases there: while the traditional two-band methods achieve its bandwidth savings mainly via employing different quantizations of the highs and the lows [5], the significance of morphological coding rests with its fundamental approach of reducing image "features" into a set of simple, efficient morphemes.

3) Although the filters used in other methods for detecting edges may be similar to those used here (i.e. the

h-filters in Chapter 2), most of the other methods rely on the zero-crossing technique [17] to accomplish the extraction of edges. This technique leads to too many false edge-detections, especially at image features which are best described by morphemes other than edges. To avoid this problem, the edge-detection portion of morphological coding was implemented here by operating on peaks, rather than zero-crossings, in the h-filter outputs (see subsection 2.2.3).

4) Segmentation coding, as presented in the above quoted sources, often uses texture as well as edges to distinguish among different image segments. Accurate rendering of texture is demonstrably unnecessary for reconstructing fair quality head-and-shoulder images (i.e. recognizable without objectionable errors). Since the images of concern here are of this type, texture was not afforded special treatment per se in morphological coding as implemented here.

5) In a broader view of morphological coding, the "morpheme" notion can be associated with a local measure of a certain signal characteristic rather than with a set of basis functions. Such is the view presented in Chapter 7; the morpheme can be the direction along which a given pel can be most efficiently predicted. Despite the similarity

between such embodiement of the morpheme notion and predictive coding, a fundamental difference is that the "direction" morpheme is very much attached to, and governed by, the behavior of local image features. More traditional predictive coding techniques do not possess such ability to cater to image features.

The essence of morphological coding is the reconstruction of an arbitrary signal as a sum of simpler functions. The simplicity of these functions is dictated by the requirement to achieve efficient coding. However, it is this very simplicity that causes such a reconstruction to be approximate. Moreover, the non-orthogonality of these functions coupled with their simplicity leads to inherent ambiguities of the reconstruction process; that is, for a given error criterion, it may be possible to code a signal using two different sets of morphemes. It is a major aim of this thesis to explore the efficiency and flexibility provided by a non-orthogonal, simple set of morphemes on the one hand, and the limitations in rendering high quality images which may be caused by the particular choice of the morpheme set, the ambiguities in the reconstruction process, and the fundamental problem of trying to approximate arbitrary-shaped as a sum of a finite set of simple non-orthogonal functions on the other hand.

Morphological coding is most useful for images in which

the important information lies in a small number of spatially localized features, substantially fewer than the number of pixels. The major problems of implementing morphological coding are:

1) selection and definition of allowed morphemes, and

2) image decomposition into morphemes.

Assume a specific ensemble of images is to be morphologically coded so as to minimize some quantitative cost function that combines a measure of image fidelity with a measure of code entropy. An immediate problem is the optimum choice of possible morphemes to be used in representing these images.

Satisfactory choices can be found in special cases where the ensemble of images is highly restricted. For example, if the images were all one-dimensional and composed of super-imposed randomly positioned boxcar functions of standard width, then the only morpheme required would be one such boxcar. The code for such an image would simply consist of the coded positions and amplitudes for each boxcar. If the boxcars randomly had one of two possible widths, then two morpheme types would be optimum. If the ensemble consisted only of random amplitude steps at random locations, then a single unit step function would be an optimum morpheme. For most images a larger set of morphemes is typically

optimum, but definition of such more general optimum sets remains an unsolved problem. Images of major interest to us here are of the type encountered in televideo conferencing, such as head-and-shoulder images. These images have a more general structure than those composed of highly restricted functions.

The lack of a rigorously structured derivation for optimum morphemes for this case prompts arbitrary choices of simple sets. We, therefore, suggest the use of a set consisting of the following two types of morphemes that perform well for one dimensional scan lines:

1) Gaussians with half-amplitude width ranging from one pixel to several tens of pixels. For example, for a 192-pixel scan line the broadest gaussian has a half-amplitude width of $\sim$ 81 pixels.
2) Edge function e(z), which we define as

$$e(z) = \int_{-\infty}^{z} [g_a(z) - g_b(z)]dz$$

where g_a and g_b are co-located positive gaussians of different widths but equal areas. In the example of the 192-pixel scan line mentioned above, the width of the wider gaussian g_b is fixed at $\sim$ 81 pixels, whereas the width of the narrower gaussian g_a varies from one to

approximately 18 pixels. Both sets of morphemes are suggested in Figure 1.1.

Although biological systems employ two dimensional operators for vision, in electronic systems it is usually easier to operate first on a raster scan line-by-line, and then to combine the results of these one-dimensional operations. We, therefore, resolve each line of scan into a set of morpheme "events", and then expolit the vertical correlation in the image by combining morpheme events on nearby lines into "strokes" similar to the brush strokes in an artist's painting. In addition to a coded starting location, a stroke can be characterized by a coded width, a coded intensity, a coded direction, and a coded length, each of which may furthermore vary along the length of a stroke.

1.2 This Thesis

We start in Chapter 2 with an analytical derivation of optimum morpheme detectors for various levels of background noise. The approximations made in implementing these theoretically derived detectors are then discussed. The major objective of such approximations is to achieve computational efficiency.

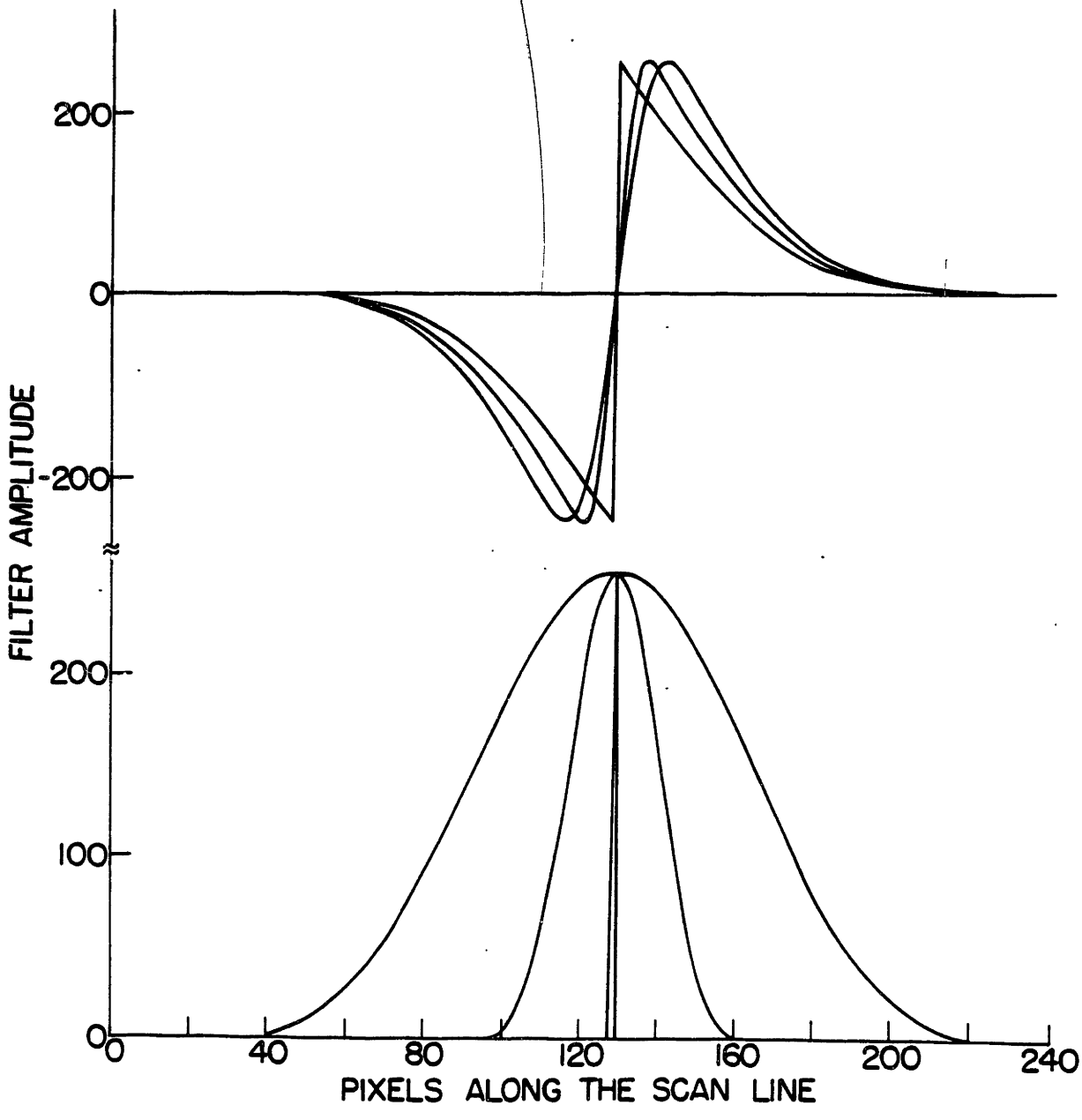


Fig. 1.1 (a) Gaussian morphemes. Three out of twenty-one gaussian morphemes are shown. Their half-amplitude widths are 1, 27, and 81 pixels; the narrowest and broadest are included. (b) Edge morphemes. Three out of twelve edge morphemes are shown. The narrower g in equation (1.1) for these edge morphemes has a half-amplitude width of 1, 9.5, and 17.9 pixels; the sharpest and softest edges are included.

Chapter 3 uses the implemented detectors in performing one-dimensional morphological coding; i.e. reducing image scan lines into one-dimensional morphemes, namely gaussians and edges. The algorithm for this method is described and the results are presented in terms of 1) numerical evaluation of the error in the reconstructed images, 2) direct graphical comparison of original and reconstructed one-dimensional images (image scan lines), and 3) original and reconstructed half-tone facial images, when morphologically coded one-dimensional scan lines are arranged to form a two-dimensional image.

Chapter 4 presents the details of the method of grouping one-dimensional morphemes occurring in image scan lines into vertical runs, i.e. strokes, using mean-square error interpolation fit of the three stroke parameters: position, width and amplitude. This process is compared and contrasted to the familiar problem of the track-while-scan radar. Stroked head-and-shoulder images, which begin to show the limitations of morphological image coding, are presented in this chapter.

Noise in the morphologically coded images presented in Chapters 3 and 4 can be classified into two major categories: one- and two-dimensional noise. Chapter 5 defines and studies the different types of imperfections of each of the above two categories. Each type is studied in as much detail

as is allowed by its separability from the other types. A detailed understanding of the imperfections in the reconstructed images is gained through processing controlled test patterns.

Chapter 6 compares between morphological coding and other coding techniques. The comparison emphasizes the exclusive features of morphological coding as opposed to most other methods of coding, especially transform coding. Images processed by each of the two methods are presented in this chapter, and their quality and bit-rate are compared.

Chapter 7 offers a view of morphological coding in a different perspective. Rather than having the notion of the "morpheme" associated with a set of basis functions, the "morpheme" is proposed to be a local measure of a certain characteristic of the signal. This chapter demonstrates the validity of one such local measure, namely the "direction" along which the signal exhibits the highest correlation locally. The chapter also sets the stage for future work to exploit this "direction" in transform-domain processing to achieve more significant bandwidth savings.

Finally, the main conclusions of this research effort are summarized in Chapter 8. The computer programs for achieving the results stated in Chapters 2, 3, and 4 are listed in the appendices.

CHAPTER 2

DESIGN & IMPLEMENTATION OF MORPHEME DETECTORS

This chapter presents a systematic analytical derivation of the impulse response of the linear filters designed to detect the morphemes introduced in the first chapter. The similarity between these derived filters and others thought to be part of the human psychovisual system is noted. Simplifications to achieve a computationally efficient algorithm implementing such morpheme detectors are presented, and justifications for these simplifications are discussed. Exact closed-form analytical expressions are derived for the implemented (i.e. simplified) morpheme detectors.

2.1 Design of Morpheme Detectors

In the case where the image is best decomposed into non-orthogonal morphemes, the desired linear filters are those which respond most uniquely to specific morphemes and simultaneously respond least to noise and to the other morphemes.

Although the motivation for adopting the signal-to-

noise ratio may be intuitive as mentioned above, this criterion can be justified more rigorously as follows. Using one of the standard, basic criterion, i.e. Bayes criterion, [18], we obtain the following statement of the general problem of signal detection: the objective is to minimize the following cost function:

$$\int_S C(s_i/s_j)\ p(s_i,s_j)$$

where $C(s_i/s_j)$ is the cost for estimating the signal to be s_i when in fact it is s_j, $p(s_i, s_j)$ is the probability that the observed signal is estimated as s_i when in fact the signal is s_j, and S is the entire signal space. Without loss of generality, we can consider the development of the above expression in the case of two signals. Assuming the cost of a wrong decision is higher than the cost of a correct decision, it can be shown [18] that minimizing the above cost function is equivalent to adopting the following likelihood ratio:

$$\frac{p(Y/s_1)}{p(Y/s_0)} \underset{\text{Say } s_0}{\overset{\text{Say } s_1}{\gtrless}} \frac{P_0(C_{10}-C_{00})}{P_1(C_{01}-C_{11})}$$

where $p(Y/s_i)$ is the probability the observed signal Y lies

in the decision region of s_i when in fact the signal is s_i, P_i is the a priori probability of s_i, and C_{ij} is the cost of "saying" s_i when in fact the signal is s_j. It can be shown [19] that the probability of error resulting from adopting the above likelihood ratio test is a monotonically decreasing function of the distance between the two signals s_1 and s_0, normalized with respect to the standard deviation of the uncorrelated noise. This is true regardless of the probability distribution of either signal. For example, for gaussian signals, this function is the complementary error function. When the problem is to decide whether a given signal corrupted by white gaussian noise exists or not, the error probability becomes:

$$\operatorname{erfc}\left[\left(\frac{E}{N}\right)^{\frac{1}{2}}\right]$$

Where erfc( ) is the complementary error function, E is the signal energy, and N is the variance of the noise. Note that the above mentioned normalized distance $(E/N)^{\frac{1}{2}}$ is nothing but the familiar signal to noise ratio; the higher it is, the lower the error probability as indicated by the above expression. Hence the justification of maximizing the signal-to-noise ratio.

We have derived the impulse response for such filters which maximize the signal-to-noise ratios for specific gaussian and edge morphemes embedded in noise plus other morphemes; the positions and amplitudes of these interfering morphemes are assumed to be distributed uniformly and randomly.

The procedure used to derive the optimum linear filter for detecting a specific morpheme g(z) is the following. Let h(z) be the desired filter for detecting g(z); we require that h(z) maximize the signal-to-noise ratio S/N, where we define

$$\frac{S}{N} = \frac{(\bar{h}\cdot\bar{g})^2}{E[(\bar{h}\cdot\bar{b})^2] + N(\bar{h}\cdot\bar{h})} \qquad (2.1)$$

and where $\bar{h}$, $\bar{g}$, and $\bar{b}$ are equivalent vector representations of the corresponding continuous functions, and $\bar{b}$ is any morpheme other than $\bar{g}$, including those which are identical to $\bar{g}$ but are displaced in position. N is the variance of the additive white gaussian noise, E [ ] is the expected value operator over the ensemble $\bar{b}$, and $\bar{h}\cdot\bar{g}$ is a vector dot product. Note that the criterion in (2.1) is identical to that for a matched filter except for the first term in the denominator. The reason for this difference is that a matched filter is used when the signal to be detected is

corrupted only by white noise. Here, the signal to be detected (i.e. the $\bar{g}$ morpheme) is also embedded in another type of noise, namely the intersymbol interference caused by other morphemes. This dictates the presence of the first term in the denominator.

The criterion in (2.1) above employs the rms values of the involved terms. In principle, other measures of these terms, such as their absolute values or any even power thereof, may be employed. The advantage of the rms measure is that it is the most commonly used in such criteria, and therefore yields most readily to comparison with results obtained by others. Moreover, it is not obvious that using any other measure would result in better filter designs than the one derived here.

We let

$$\bar{h}/a = \bar{g} + \bar{F} \tag{2.2}$$

where $\bar{g}$ and $\bar{F}$ are orthogonal vectors and a is a constant. Substitution of (2.1) into (2.2) yields a numerator that is independent of $\bar{F}$ and a denominator which is:

$$E[(\bar{g}\cdot\bar{b} + \bar{F}\cdot\bar{b})^2] + N(\bar{g}\cdot\bar{g} + \bar{F}\cdot\bar{F}) \tag{2.3}$$

Maximizing the signal-to-noise ratio then is equivalent to choosing $\bar{F}$ to minimize the denominator. We expand $\bar{F}$ in the basis $\bar{f}_i$ where

$$\bar{F} = \sum_{i=1}^{M} d_i \bar{f}_i, \tag{2.4}$$

$\{d_i\}$ are constant scalers, and $\{\bar{f}_i\}$ is a set of basis functions, not necessarily orthonormal. We let $\{\bar{f}\}$ be a full set of modified sines and cosines with a fundamental period of 256 pixels; each member was individually modified to be orthogonal to $\bar{g}$ using the Gram-Schmidt method. M was typically between 60 and 80.

Substitution of (2.3) into (2.4) and then equating to zero the derivative with respect to d_i yields M simultaneous equations:

$$d_i = \frac{1}{B_i}(C_i - D_i) \tag{2.5}$$

where $$B_i = E[(\bar{b}\cdot\bar{f}_i)^2] + N\bar{f}_i\cdot\bar{f}_i$$

$$C_i = E[\sum_{j\neq i} d_j(\bar{b}\cdot\bar{f}_j)(\bar{b}\cdot\bar{f}_i)] + N\sum_{j=i} d_j\cdot\bar{f}_i\cdot\bar{f}_j$$

$$D_i = E[(\bar{g}\cdot\bar{b})(\bar{b}\cdot\bar{f}_i)]$$

These M simultaneous equations (2.5) can be solved for $\{d_i\}$, which yields $\bar{F}$ and, within a constant, the desired optimum filter $\bar{h}$. The equations (2.5) depend on the assumed noise N and the assumed ensemble $\{\bar{b}\}$.

In general one would compute the expected value in equation (2.1) over a representative ensemble of images. Here images in this ensemble consisted of single morphemes of amplitude 255, which were assumed to be independently positioned and equiprobable; the ensemble excluded only $\bar{g}$ at its central position. Thus the first filters were designed for detecting gaussian morphemes in a background of gaussian white noise and gaussian morphemes of different equiprobable widths. The 21 possible widths at half amplitude were: 1, 2, 2.7, 3.1, 3.5, 4.5, 5.1, 5.6, 6.1, 8.4, 9.5, 12.6, 14.4, 17.9, 22, 26.9, 34.2, 43, 55, 68.6, and 80.6 pixels. The slight irregularities in this sequence resulted from compromises made to achieve computationally efficient filters for generating these gaussians. The optimum filters for detecting gaussians of width 26.9 are shown in Figure 2.1a for signal energy ($\bar{g}\cdot\bar{g}$) 1.5 dB greater than the expected value of the interference energy E[b·b] and from 13 to 23 dB greater than the noise power N. Similar optimum filter responses are obtained for detecting gaussians of other widths.

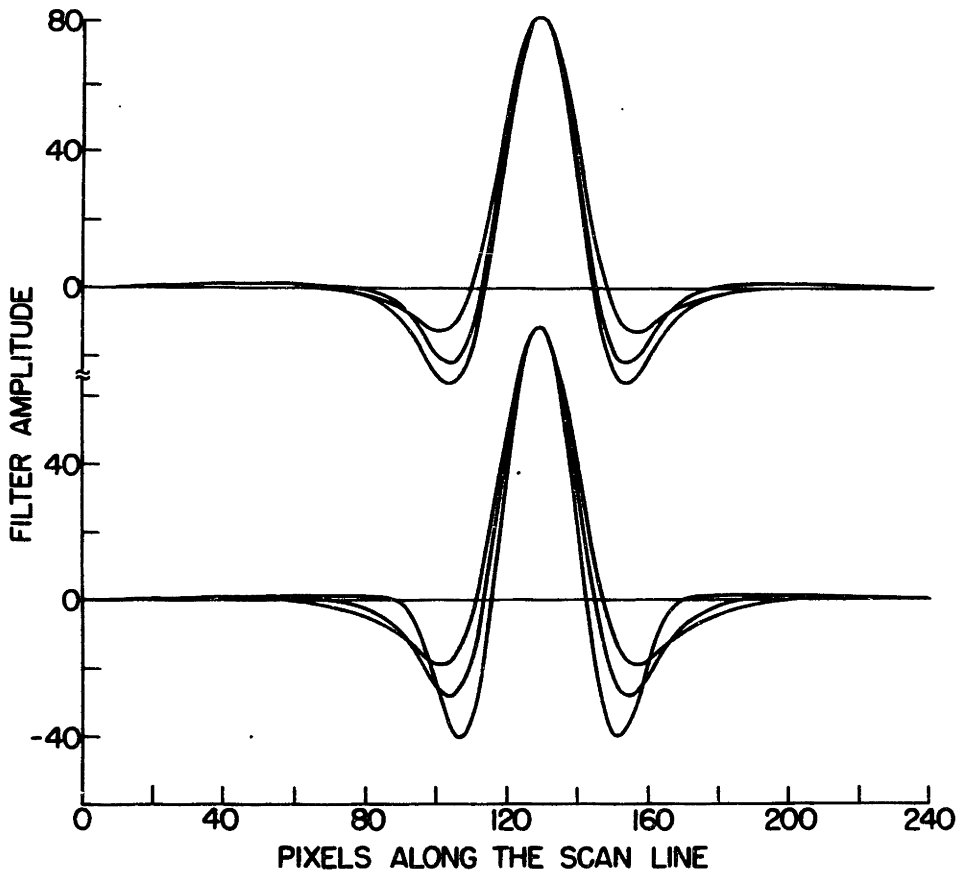


Fig. 2.1 (a) Optimum detectors of gaussian morphemes against a background of other gaussian morphemes and additive white noise. These three filters were designed to detect a gaussian morpheme with a half-amplitude width of 26.9 pixels for different noise levels.

Fig. 2.1 (b) Optimum detectors of gaussian morphemes against a background of other gaussian and edge morphemes and additive white noise. These three filters were designed to detect a gaussian morpheme with a half-amplitude width of 26.9 pixels for different noise levels. The filter with the largest sidelobes is the one from Figure 2.1(a) for the highest signal energy-to-white noise ratio.

The second set of filters was designed to detect a gaussian of width 26.9 in a background of both gaussian and edge morphemes, where the set of edge morphemes was defined by equation (1.1) for $g_a(z)$ having a width of 1, 2, 3.1, 3.5, 4.5, 5.1, 6.1, 8.4, 9.5, 12.6, 14.4, 17.9, 22, or 26.9 pixels, and $g_b(z)$ having a width of 80.6. The assumed peak-to-peak amplitudes of the interfering edges were 255. The resulting optimum filters are shown in Figure 2.1b for the signal energy 3 dB greater than $E[b \cdot b]$ and from 13 to 23 dB greater than the noise power N. In both cases, Figures 2.1a and 2.1b, it is seen that for higher noise levels the tails of the optimum filter become smaller but broader as they average over a larger surrounding area, and the response approaches that of a matched filter.

Finally, the optimum detection filters for edges where $g_a(z)$ has width 9.5 are presented in Figure 2.2 for the signal energy 6 dB greater than $E[b \cdot b]$ and from 25 to 35 dB greater than N for the same background set of morphemes as in the previous case. The broadest filters correspond to the highest noise levels.

One very interesting result is that the optimum matched filters derived here for gaussian morphemes have impulse responses that closely resemble those deduced for neurological networks in the human visual system. Wilson and Bergen [20]

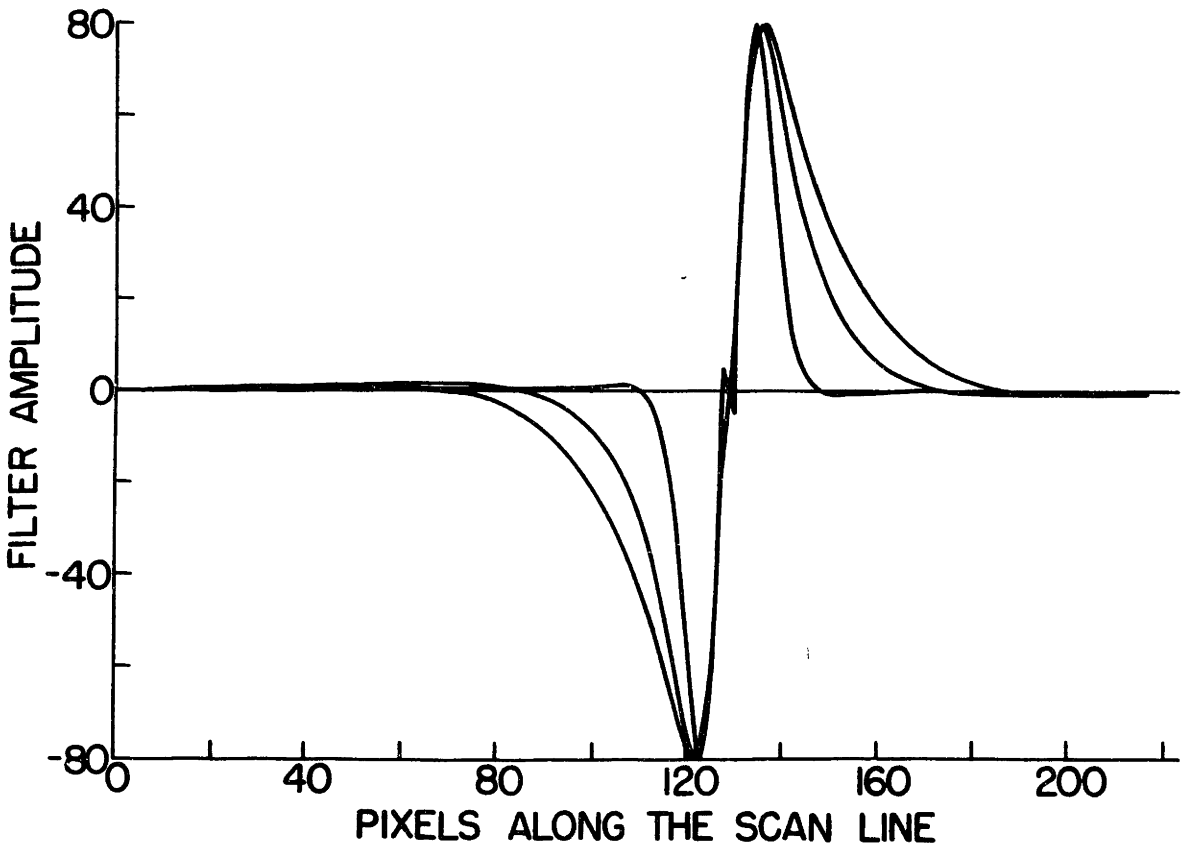


Fig. 2.2 Optimum detectors of edge morphemes against a background of other edge and gaussian morphemes and additive white noise. These three filters were designed to detect an edge for which the narrower g in equation (1,1) has a half-amplitude width of 9.5 pels, for different noise levels.

have noted that the human visual system apparently employs two-dimensional filters with impulse responses resembling the difference between two gaussians with slightly different widths; four such filter types have been identified with widths varying approximately a factor of two from one type to the next. It may be coincidence that the human visual system incorporates filters that resemble those deduced here, or it may be that one intermediate representation for images in the human system is the sum of gaussian or both gaussian and other morphemes, and the neurological filters have evolved into this nearly optimum form as a consequence.

2.2 Implementation of Morhpeme Detectors

Because the computation of these responses for morpheme filters would be burdensome, we simplified the computations in three ways: 1) efficient additive binary-tree algorithm provided approximate gaussian "g" filter impulse responses; differences between such "g" filters approximate the impulse responses in Figures 2.1, 2) only representative filters were employed, and 3) the edge "e" filters were approximated by operating only on detected peaks in other filter outputs. These three approximations are described further below.

2.2.1 Implementing the h-filters

First, we used filters that could be computed very efficiently via an additive binary-tree algorithm, and still have impulse responses that approximate those desired. The impulse responses for three of the five initial filters we employed are presented in Figure 1.1a. These approximately gaussian "g" filters were synthesized by cascading unit cells, where each unit cell consisted of a delay of D pixels and an adder, as illustrated in Figure 2.3. The delays employed in the 24 cells were 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 4, 4, 6, 6, 8, 10, 12, 16, 20, 26, 32, and 34 pixels, respectively. The g filters were modified such that mirror images were juxtaposed at each end of the original image in an attempt to avoid serious edge discontinuities. The average computation time for one cell to produce one pixel output was 3.6 μs, and the time required for 24 cells to filter completely a 192-pixel image was 17 ms using a Data General Nova-4 computer.

The impulse responses of these g-filters can be analytically formulated as follows. In the additive binary-tree algorithm which produces the g-filters, the first eight filters are produced by adding pairs of consecutive pels. The impulse responses of these g-filters are illustrated

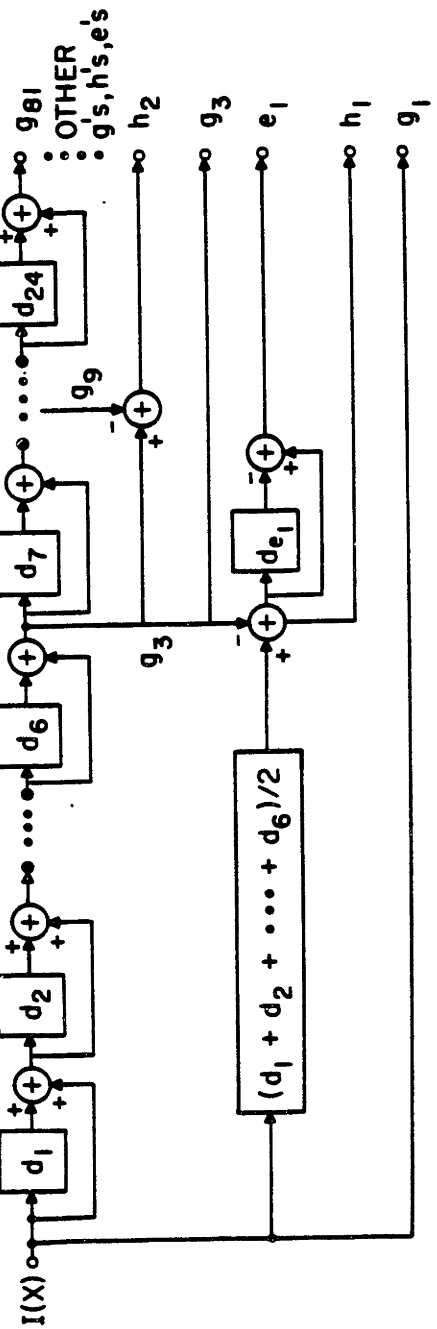


Fig. 2.3 Procedure for efficiently computing the g, h, and e filters.

below for the first four of them.

ORIGINAL LINE:	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
g_1:	0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
g_2:	0 0 0 0 0 0 0 0 1 2 1 0 0 0 0 0 0 0 0
g_3:	0 0 0 0 0 0 0 1 3 3 1 0 0 0 0 0 0 0 0
g_4:	0 0 0 0 0 0 0 1 4 6 4 1 0 0 0 0 0 0 0

The numbers in each filter g_n are identical to the coefficients of the familiar binomial theorem:

$$(a+b)^n = 1\cdot a^n + n\cdot a^{n-1}b + \frac{n(n-1)}{1\cdot 2}\cdot a^{n-2}b^2 + \cdots\cdots$$
$$+ \; n\cdot ab^{n-1} + 1\cdot b^n$$
$$= \frac{n!}{(n-r)!r!}\cdot a^{n-r}b^r, \; r = 0, 1, 2, \ldots\ldots, n$$

Therefore, with the center of each filter at the origin we can write:

$$g_n(z) = \frac{1}{2^{n-1}} \frac{n!}{[n-(z+\frac{n}{2})]!\;(z+\frac{n}{2})!}, z = -\frac{n}{2}, -\frac{n}{2}+1, \ldots, \frac{n}{2}$$
$$g_n(z) = 0 \text{ elsewhere} \qquad (2.6)$$

where: $\frac{1}{2^{n-1}}$ is for normalization

and $1 \leq n \leq 8$

Now for $n > 8$, the algorithm employs variable delays, d, greater than one; i.e. instead of equation 2.6 above we have:

$$g_n(z) = g_{n-1}(z) + g_{n-1}(z-d_{n-1}),\ d_{n-1} > 1,\ |z| < \left(\sum_{\ell=1}^{n-1} d_\ell\right) + 4$$

$$g_n(z) = 0 \text{ elsewhere} \qquad (2.7)$$

By iteration:

$$\begin{aligned} g_n(z) &= g_{n-2}(z) + g_{n-2}(z-d_{n-2}) + g_{n-2}(z-d_{n-1}) \\ &\quad + g_{n-2}\,(z-d_{n-1} - d_{n-2}) \\ &= g_{n-3}(z) + g_{n-3}(z-d_{n-3}) + g_{n-3}(z-d_{n-2}) \\ &\quad + g_{n-3}\,(z-d_{n-1}) \\ &\quad + g_{n-3}(z-d_{n-2} - d_{n-3}) + g_{n-3}(z-d_{n-1} - d_{n-3}) \\ &\quad + g_{n-3}(z-d_{n-1} - d_{n-2}) \\ &\quad + g_{n-3}(z-d_{n-1} - d_{n-2} - d_{n-3}) \end{aligned}$$

Carrying the iteration all the way down to g_8, we get:

$$2^{n-1} \cdot g_n(z) = g_8(z) + \sum_{i_1} g_8(z-d_{i_1}) + \sum_{\substack{i_1,i_2 \\ i_1 \neq i_2}} g_8(z-d_{i_1}-d_{i_2}) + \ldots$$

$$+ \sum_{\substack{i_1,i_2,\ldots,i_{n-8} \\ i_1 \neq i_2 \\ i_1 \neq i_3 \\ i_{n-9} \neq i_{n-8}}} g_8(z-d_{i_1}-d_{i_2}-\ldots.d_{i_{n-8}}),$$

$$\text{for } |z| \leq \sum_{\ell=1}^{n-1} d_\ell + 4 \qquad (2.8)$$

$$g_n(z) = 0 \text{ elsewhere}$$

where $8 < i_1, i_2, \ldots, i_{n-8} < n-1$, 2^{n-1} is for normalization, $n > 8$,

and $g_8(z) = \dfrac{8!}{(4-z)!(4+z)!}$, $|z| < 4$

$= 0$ elsewhere

} as given in eq.(2.6)

Note that the last summation in equation (2.8) has only one term and can be written as:

$$g_8(z - \sum_{j=1}^{n-8} d_j)$$

Equations (2.6) and (2.8) above give exact, closed-form analytical expressions for all g-filters.

Since each h-filter is but a difference of two g-filters, we can write:

$$h_{ij}(z) = g_i(z) - g_j(z), \quad j > i \qquad (2.9)$$

where the two g-filters above are as defined in equations (2.6) and (2.8).

2.2.2 Processing a Proper Subset of the Filters

Second, it was noted that not all filter responses required examination. We processed the output of only five g-filters having the half-amplitude widths of 1, 3.1, 9.5, 26.9, and 80.6 pixels for the case of 192 pixel image-line, and 1, 2.7, 6.1, 17.9, and 43 pixels for the case of 128 pixel image-line. The four "h-filters" which are formed as differences between each two adjacent g-filters of the five g-filters mentioned above are designated h_1 thru h_4 in order of increasing width of their impulse responses. These impulse responses, as shown in Fig. 2.4, are intended to approximate the desired impulse responses in Fig. 2.1. The approximate outputs of other h-filters, or, equivalently, g-filters, could readily be deduced from this subset as explained below.

In Fig. 2.5 the peak responses of 21 gaussian filters are

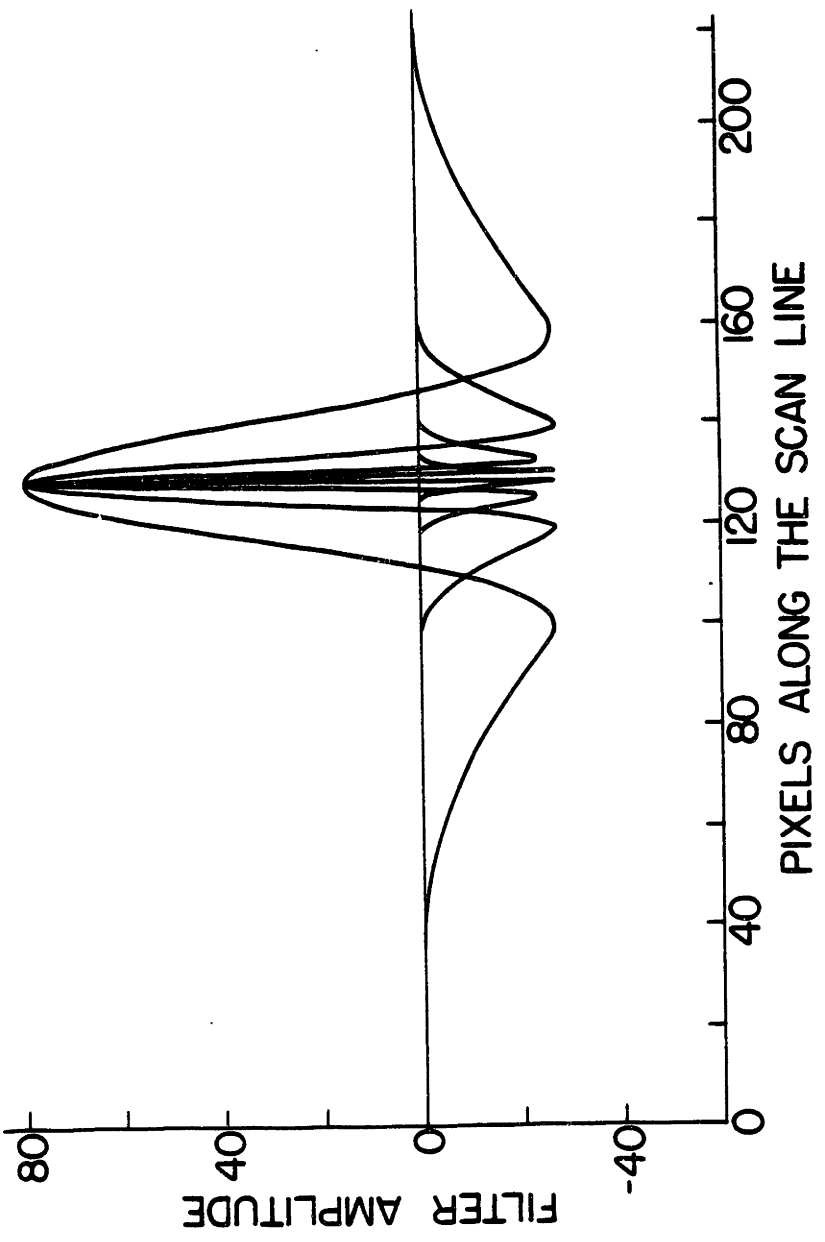


Fig. 2.4 The four h-filter impulse responses.

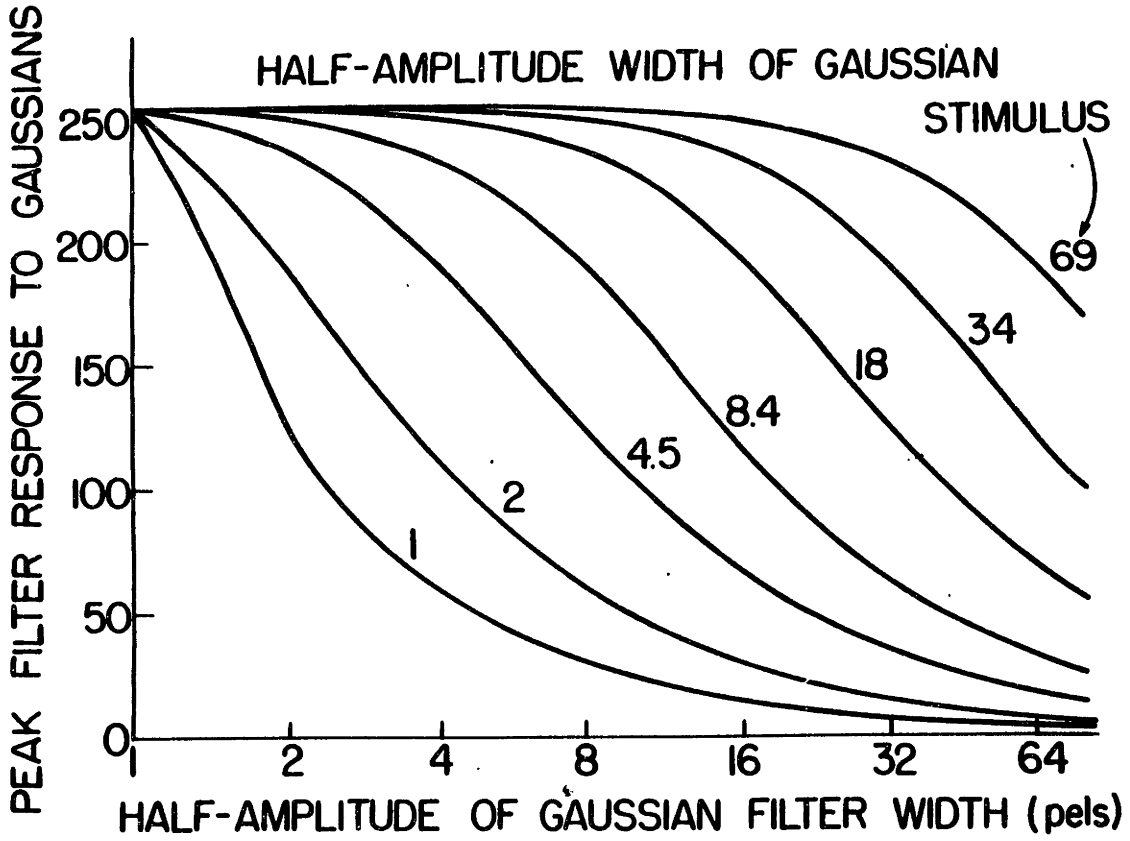


Fig. 2.5 Responses of g-filters to gaussian inputs. Each curve represents the responses of 21 g-filters, ranging in half-amplitude width from 1 to 81 pixels, to a gaussian input with amplitude 255 and with the indicated half-amplitude width.

plotted for various gaussian stimuli in the limit of a continuum rather than discrete pixels. These reponses are of similar form except for translation along the abscissa of Figure 2.5. The errors in estimating these responses from samples thereof can be deduced from spectral analysis; we regard the abscissa of Figure 2.5 as linear in "time" for this purpose. In particular, the energy in the spectrum of this filter response function below the Nyquist frequency f_N used here is approximately 18 dB greater than the aliased energy above f_N for the worst case represented by the innermost curve of Figure 2.5. This leftmost-margin would increase to 27 dB if the filters were chosen such that the raio of consecutive widths were reduced from three to two. The numbers become 40 dB and 47 dB respectively for the responses to stimuli wider than $\sim$ 8 pixels. Thus sampling errors can be small compared to errors introduced by other approximations considered later.

2.2.3 Implementing the e-filters

The third approximation was the synthesis of edge "e" filters by subtracting detected peaks in the output of each h-filter from adjacent peaks in the output from the same h-filter which have nearly equal amplitude and opposite signs, and

were offset by ∿ 3, ∿ 7, ∿ 15, and ∿ 30 pixels for h_1,...., h_4, respectively. The nominal impulse responses for three of these four synthesized edge e-filters (with typical offsets) are illustrated in Figure 2.6(b); they are reasonable approximations to the filters shown in Figure 2.2, except that they have small overshoots that reduce their response to ramp functions. The step responses of these three synthesized e-filters are illustrated in Figure 2.6(a); they have zero response to D.C. signals and nearly zero response to ramps. Their relative peak responses to a step of amplitude 255 are 94, 107, and 125, in order of increasing filter width. The use of only 4 e-filters was deemed acceptable because these responses could also be interpolated to yield reasonable estimates for peak outputs from other possible e-filters.

2.3 The "fat-g's"

The four h-filters, and for that matter any h-filter with similar shape, do not respond to low-frequency functions such as D. C. and ramps. To compensate for that, the low-frequency components were estimated separately. Two different approaches were tried for this purpose. The first was to sample the output of the widest g-filter (width = 80.6 or 43 pixels as mentioned above) at prefixed positions

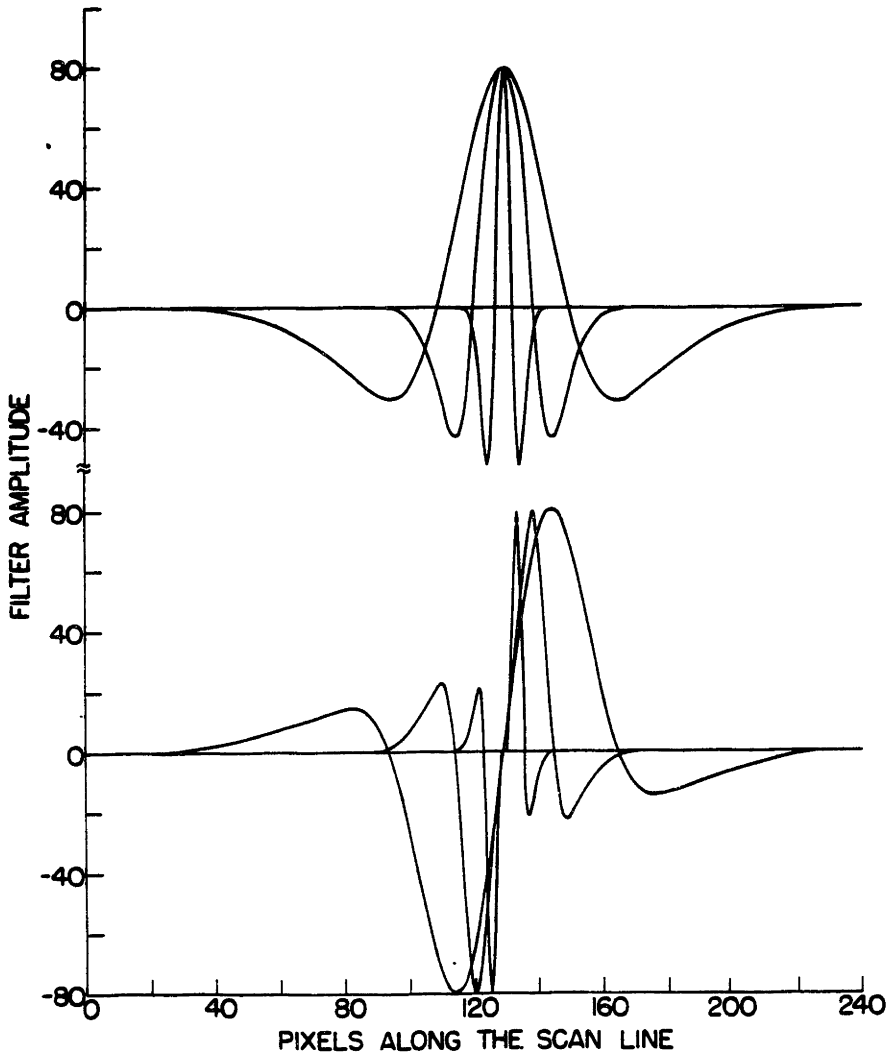


Fig. 2.6 (a) Step responses of synthesized e-filters. The relative peak responses to a step of amplitude 255 are 94, 107, and 125, in order of increasing filter width for the filters of Figure 2.6(b).
(b) Impulse responses of synthesized e-filters.

occurring at intervals of half the width of the widest g-filter. After correction for the estimated gaussian and edge morphemes, straight-line interpolations would then connect the samples to form the final estimate of the low-frequency content of the image line.

The second approach for estimating low-frequency image features is the following. A seven-point estimator, where the points are separated by the same intervals mentioned above, is computed for the image line using the least-rms error criterion. The computation is made such that one widest gaussian (hence the name "fat-g") is centered at each of the seven points. This estimation procedure is outlined in [21]. Next, this low-frequency estimator is subtracted from the original image line to produce the "highs" for the h and e-filters to operate on in isolation of low-frequency interference. Once the highs are reduced to gaussian and edge morphemes, these "estimated" highs are then subtracted from the original image, yielding the structures which the h and e-filters failed to capture. This residual component of the image is then estimated with the same seven-point operator used at the beginning of this method. Finally, the image is reconstructed as the sum of these fat-g's plus the edge and gaussian morphemes.

The second method mentioned above was found to produce

better image quality. Therefore, it was chosen over the first method for processing the images presented here.

CHAPTER 3

ONE-DIMENSIONAL IMAGE CODING: PROCEDURE & RESULTS

This chapter shows in detail how the output of the filters described in the previous chapter is used to reduce each image line into a finite set of one-dimensional morphemes. The first section below gives a brief, general overview of the method of approach. An analytical formulation of the method, together with a more detailed description of the involved processes, is provided in the second section. Finally, the third section presents results of decomposing various image lines into one-dimensional morphemes. It is these morphemes that are later combined in Chapter 4 into two-dimensional strokes.

3.1 A General Overview of the Algorithm

The least-cost decomposition of one-dimensional images, i.e. lines-of-scan in a two-dimensional image, into morphemes has no known general analytic solution and so approximate numerical techniques are used here. This fact was previously suggested by the approximate manner in which the morpheme detectors in Chapter 2 were implemented. To achieve further

computational simplicity the decomposition process has been divided into three steps:

1) The image is linearly filtered.
2) The local maxima and minima of each filter output are identified by their position and amplitude.
3) On the basis of these peak filter responses, the positions, amplitudes, and widths of morphemes are estimated.

This process is generally imperfect because the filter impulse responses were optimized only for particular noise characteristics and image ensembles, and because only a subset of all desired filters are actually employed (see Chapter 2). In addition, we elected to reduce computation further by operating only on negative and positive peak responses for each filter, where the peaks are local extrema. Local negative maxima and positive minima were ignored.

If each morpheme appeared in isolation in the input image and were noise free, then these peak filter responses would be adequate for nearly perfect image reconstruction. Fig. 3.1 illustrates how such responses can yield perfect reconstructions for isolated gaussians in an input image; the errors are less than 0.5 (out of 255) and result primarily from round-off error.

Although simultaneous equations might have been solved to estimate the non-orthogonal morphemes in a typical image

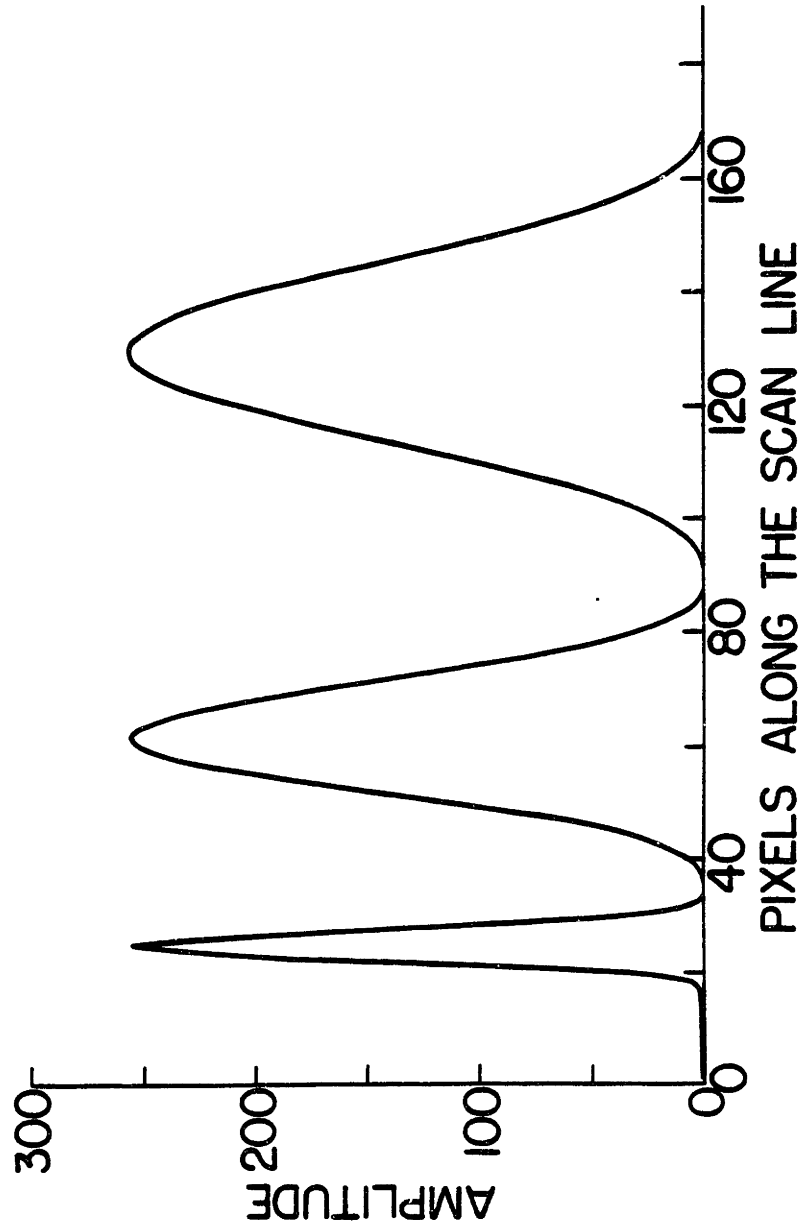


Fig. 3.1 Near perfect reconstruction of isolated gaussian morphemes. Both the original and the reconstructed morphemes are superimposed and appear to be identical.

line, the algorithm used here is sequential and employs only first and second-order corrections; it therefore is faster and more stable. Full comparison of the error performance of this iterative approach with that of other alternatives is a subject of future research. In any case, the present algorithm departs from ideal performance only as the number of overlapping morphemes in the image becomes significant.

3.2 Analytical Formulation and Detailed Description of the Algorithm

Peaks, i.e. local positive maxima and negative minima, in the outputs of the four h-filters introduced in Chapter 2, are found by detecting a change of sign in the slope. Only peaks whose absolute value is above a given threshold are retained. As indicated above, the algorithm uses these peaks, (P), to estimate morphemes, (g), employing only first and second-order sequential corrections for inter-symbol interference.

Any peak P_{mk} in the output of filter h_m due to some morpheme g_k is not only the central response of h_m to g_k, but also the response of h_m to all morphemes in the image producing a finite response value at the center of g_k.

That is:

$$P_{mk} = \int_{-\infty}^{\infty} h_m(z) \cdot [g_k(z) + \sum_{\ell \neq k} g_\ell(z - r_\ell)]dz$$

where r_ℓ is the distance between the centers of g_k and g_ℓ. P_{mk} is used to make a rough estimate $\tilde{g}_k$ of g_k, using one of the following two methods:

A) P_{mk} is paired with another peak in the output of an adjacent h-filter, h_n, if both peaks have the same sign and fall within a certain distance, (in pels), of each other. The pel "windows" used for such pairing are 1, 2, and 5 pels for h_1h_2, h_2h_3, and h_3h_4 pairs, respectively. A peak pair is used to solve for the two degrees of freedom of g_k, namely amplitude and width, as follows.

g_k is assumed to be a linear combination of two consecutive morphemes g_i and g_j produced by the additive binary tree algorithm introduced in Chapter 2. i and j are determined via a table look-up (see the table in Program LEST. SV; Subroutine SUBGEST.FR in the appendix), using the ratio P_{mk}/P_{nk}. Specifically, i and j are such that:

$$Q_{mi}/Q_{ni} < P_{mk}/P_{nk} < Q_{mj}/Q_{nj}$$

where Q_{pq} is the response of h_p to g_q when g_q has a normalized amplitude of 255 (also looked up in the above mentioned

table). The amplitude and width of the morpheme are then determined by solving the following simultaneous equations:

$$X \cdot [aQ_{mi} + (1-a)Q_{mj}] = P_{mk}$$

$$X \cdot [aQ_{ni} + (1-a)Q_{nj}] = P_{nk}$$

where amplitude of $\tilde{g}_k = X \cdot 255$
and width of $\tilde{g}_k$ = width of $ag_i + (1-a)g_j$
where $o \leq a \leq 1$

B) If there is no peak in an adjacent h-filter output in the vicinity of P_{mk}, a default value for the width of $\tilde{g}_k$ is assumed. The default half-amplitude widths used for morphemes estimated from single peaks in the h_1, h_2, h_3, and h_4 are respectively 2, 6.1, 17.9, and 55 pels for the 192-pel lines and 1, 2.7, 8.4, and 17.9 for the 128-pel lines. We therefore have:

$$\tilde{g}_k(z) = \frac{P_{mk}}{Q_{mm}} \cdot g_m(z)$$

where $g_m(z)$ is the morpheme whose width is the default width associated with h_m.

Since pairs of peaks provide more accurate morpheme estimations than single peaks, as shown in the above

description, pairs are processed first. Also, since the response of narrower h-filters usually suffer less interference, their peaks are given higher priority in the estimation process. So processing starts with the h_1h_2 pair having the largest h_1 amplitude. When more than one pair are detected for a morpheme, the estimate is the average of the individual estimates provided by each pair.

After a first-cut estimate of the morpheme is made, the rest of the pairs and singles are corrected for the contribution the just-estimated morpheme would have made to those peaks, i.e.:

$$P^c_{X\ell} = P_{X\ell} - \int_{-\infty}^{\infty} h_X(z) \cdot \tilde{g}_k(z-r_\ell)dz$$

where $P^c_{X\ell}$ is the corrected $P_{X\ell}$

and X = 1 thru 4.

The algorithm implements such corrections via another table look-up, using a table containing the full response (not just the peak), of each of the four h-filters to each of the morphemes contained in the first table mentioned above. The table entry used for correcting a particular peak is determined by the filter from which the peak came, and by the distance, in pels, between the peak and the just-estimated morpheme.

Morpheme estimation then resumes starting with the narrowest and strongest pair of the remaining peaks, and ending with the broadest and weakest single; each time correcting the remaining peaks for the estimated morpheme. After each round of corrections, peaks whose absolute values fall below the above-mentioned detection threshold are deleted. When only one peak in a pair is deleted, the other is appended to the list of singles.

After all pairs and singles are processed, a second (and final) pass of correction is made. This time each estimated morpheme, starting with the most recent, corrects all morphemes estimated prior to it. This second correction process is similar to the first one. Just as P_{mk} was used to determine g_k, $P^c_{X\ell}$ is used to determine g_ℓ. Then, for the second order correction, the response of h_m to g_ℓ at P_{mk} is subtracted from P_{mk}:

$$P^c_{mk} = P_{mk} - \int_{-\infty}^{\infty} h_m(z) \cdot g_\ell(z-r_\ell)dz$$

Finally, P^c_{mk} is used as before to determine, this time, g'_k. The final product is a list of morphemes, each with an estimated position, width and amplitude.

A closed-form expression can be written for the final estimate g'_k. Using case B above for only one ℓ, we have:

$$g'_k(z) = \frac{g_m(z)}{Q_{mm}} P^c_{mk}$$

$$= \frac{g_m(z)}{Q_{mm}} \left[P_{mk} - \int_{-\infty}^{\infty} h_m(z) \cdot \frac{P_{m\ell}}{Q_{mm}} g_m(z-r_\ell)dz\right]$$

$$= \frac{g_m(z)}{Q_{mm}} \left\{\int_{-\infty}^{\infty} h_m(z) \cdot [g_m(z) + g_\ell(z-r_\ell)]dz \right.$$

$$- \int_{-\infty}^{\infty} h_m(z) \cdot \frac{g_m(z-r_\ell)}{Q_{mm}} \cdot$$

$$\left.[P_{m\ell} - \int_{-\infty}^{\infty} h_m(y) \cdot \frac{P_{mk}}{Q_{mm}} g_m(y-r_\ell)dy]dz\right\}$$

The above expression is closed-form except for the integration. However, the functions of z are in fact discrete, and the integration (summation) need not be from $-\infty$ to $+\infty$: it only needs to cover the extent of h(z). As shown in equation 2.9, the extent of $h_m(z)$ is that of the corresponding $g_j(z)$, which (see equation 2.8) is:

$$|z| \leq \sum_{k=8}^{j-8} d_k + 4$$

where j is as in equation 2.4.

So the full closed-form expression for g becomes:

$$g'_k = \frac{g_m(z)}{Q_{mm}} \left\{ \sum_z h_m(z) \cdot [g_k(z) + g_\ell(z-r_\ell) \right.$$

$$- \sum_z h_m(z) \cdot \frac{g_m(z-r_\ell)}{Q_{mm}}$$

$$\left. \cdot [P_{m\ell} - \sum_z h_m(z) \cdot \frac{P_{mk}}{Q_{mm}} g_m(z-r_\ell)] \right\}$$

To maintain simplicity of the discussion, the above method description focused exclusively on gaussians. The estimation of edges, however, follows an identical procedure to the one explained above. The only difference between edge and gaussian estimations is that the edges are estimated first; the peaks for estimating gaussians are then corrected for the edges before processing.

3.3 Results of One-dimensional Image Processing

This section presents representative cases showing the one-dimensional reconstruction capability of the algorithm. Examples of single image lines as well as entire images are given. The reconstructions in those cases were done for various image resolutions, thresholds of detection, and with and without using edge morphemes. A preliminary discussion of the reconstruction errors is offered here; a more detailed explanation can be found in Chapter 5.

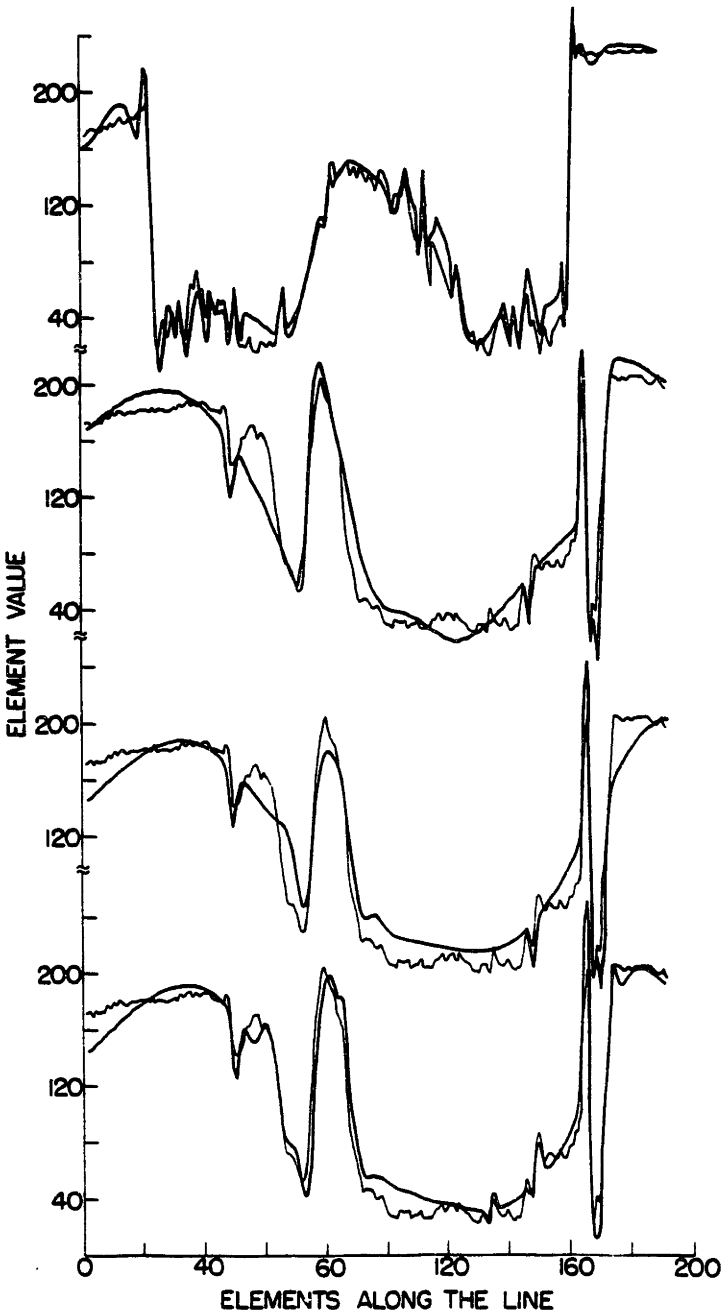


Fig. 3.2 Arranged a-d from top to bottom. See caption on the next page.

Fig. 3.2 See last page.

(a) Original and reconstructed scan lines. This scan line is indicated as line A in Figure 3.3(a). The threshold for selecting peak filter responses was 7. Only gaussian morphemes were used.

(b) Original and reconstructed scan lines. This scan line is indicated as line B in Figure 3.3(a). The threshold for selecting peak filter responses was 14. Gaussian and edge morphemes were used.

(c) Original and reconstructed scan lines; same as Figure 3.2(b), except only gaussian morphemes are used in the reconstruction.

(d) Original and reconstructed scan lines; same as Figure 3.2(c), except the threshold is 7 instead of 14.

In Figure 3.2 are presented the original and reconstructed images of two particular scan lines of a 192 by 216 pel facial image (the lines indicated in Figure 3.3a). The threshold for preserving peak responses were 7 and 14 (out of 255) units for the two cases noted. At lower thresholds, it is shown that the reconstruction can follow line details more closely due to retaining more of the smaller features. This is reflected in the corresponding average rms errors over the two lines: 13.4 vs. 17.3 for the two thresholds, respectively. Figure 3.2 also demonstrates the importance of proper selection of morphemes. The advantage of using edge morphemes in the reconstruction is seen in this figure as the better rendering of edges in the original line, and the reduction of the rms error: 17.1 compared to 18.0 for gaussians alone.

It was also found that incorporation of edges in the reconstruction of an entire facial image (as in Figures 3.3 and 3.4 below) required $\sim$ 20% fewer events than the case of using only gaussians. Thus, use of both edges and gaussians has reduced both the rms error and the number of events.

Figures 3.3 and 3.4 show original and reconstructed facial images where each scan line is morphologically coded and reconstructed independently. Figure 3.3 features a 192 by 216 pel image, while the resolution of the image in Figure 3.4 is 128 by 128 pels. These two figures show that

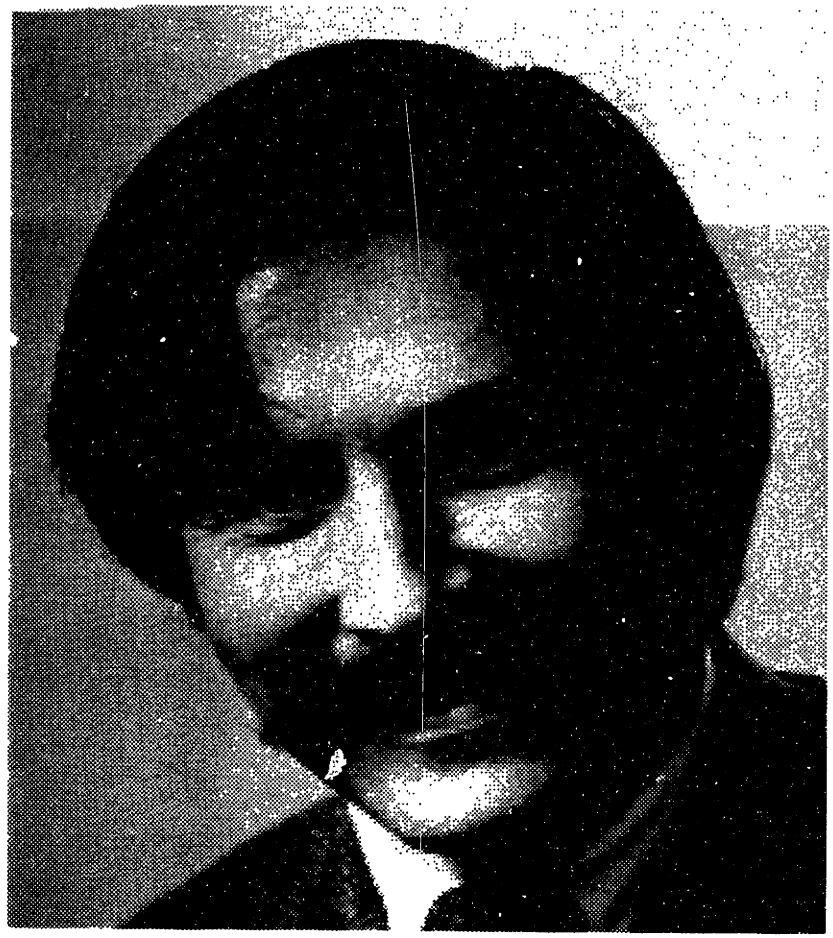


Fig. 3.3 (a) Original 192 x 216 pixel half-tone image.

Fig. 3.3 (b) Reconstructed 192 x 216 pixel half-tone image. The threshold was 7 and only gaussian morphemes were used. Average No. of events per line = 19; rms error = 15.5.

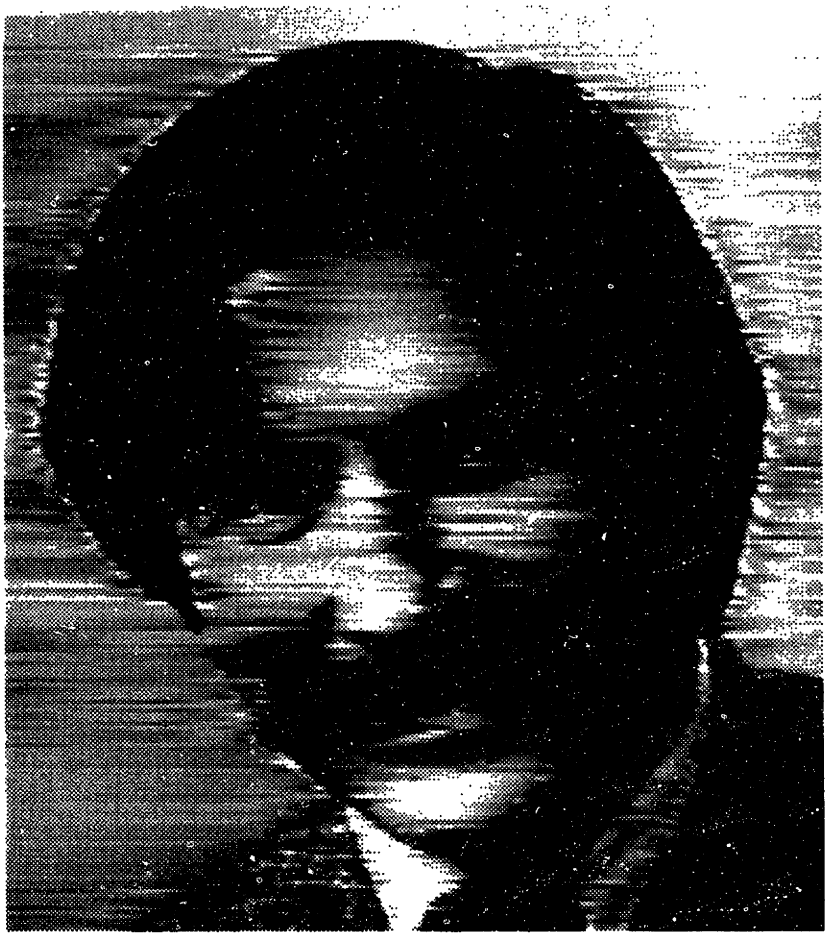


Fig. 3.3 (c) Reconstructed 192 x 216 pixel half-tone image. The threshold was 14 and only gaussian morphemes were used. Average No. of events per line = 11.5; rms error = 17.8.

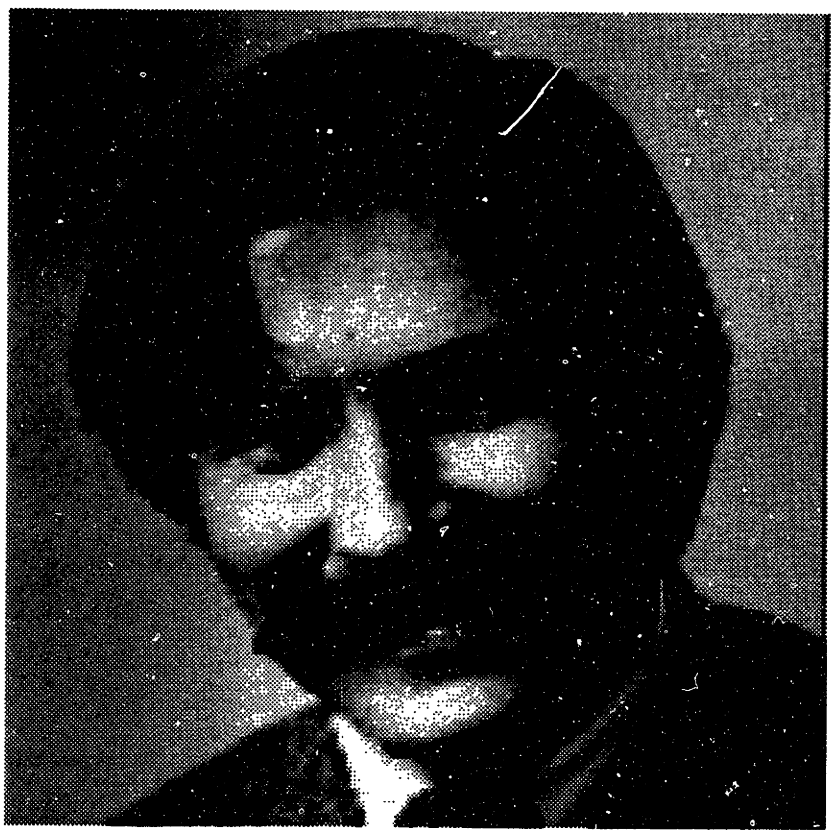


Fig. 3.4 (a) Original 128 x 128 pixel half-tone image.

Fig. 3.4 (b) Reconstructed 128 x 128 pixel half-tone image. The threshold was 7 for all h-filters and both edge and gaussian morphemes were used. Average No. of events per line = 15; rms error = 15.

Fig. 3.4 (c) Same as Figure 3.4 (b) except the threshold was 14. Average No. of events per line = 9.2; rms error = 16.8.

Fig. 3.4 (d) Same as Figure 3.4 (c) except the threshold was 14 for the h_1 and h_2 filters and 7 for the h_3 and h_4 filters. Average No. of events per line = 9.8; rms error = 15.3.

Figure 3.4 (e) Same as Fig. 3.4(d) except only gaussians are used in the reconstruction. Notice the halo around the head owing to the exclusion of edges from the set of morphemes. The number of events has increased by 20% and the rms error has increased by 15% compared to Fig. 3.4(d).

incorporating edge morphemes in the reconstruction have resulted in less horizontal streaking and the elimination of the halo around the head. The discontinuities (i.e. horizontal streaking) evident in the reconstructed images are an indication of the instabilities in this morpheme estimation process due to marginal inclusion or exclusion of morphemes and to changes in the sequence in which they are estimated.

The major sources of error between the original image and one reconstructed from superimposed morphemes for which the types, amplitudes, and positions were estimated as described above, are (1) the neglect of all small amplitude filter responses, and (2) interferences between morphemes. The errors introduced by neglect of small filter responses include those due to small responses which are properly neglected in the process of simplifying the image, as well as responses that fall below the detection threshold in the correction process. Interference errors include shifts in apparent morpheme position, amplitude, and type. The iteration process employed here corrects only the first and second-order consequences of such interference, and higher-order corrections would be necessary to reduce these errors further. The rationale for the chosen process is computational simplicity and the fact that the error due to image

simplification via omission of weak morphemes increasingly dominate as compression increases to the high levels of interest to us; at such compressions the number of morphemes is generally less than 10 percent of the number of pixels.

CHAPTER 4

TWO-DIMENSIONAL CODING: PROCEDURE AND RESULTS

This chapter presents the details of the method of grouping one-dimensional morphemes occurring in image scan-lines into vertical runs, i.e. strokes, using mean-square error interpolation fit of the three stroke parameters: position, width and amplitude. This task can be divided into two steps. The first, henceforth referred to as the "grouping" problem, involves making the decision as to which of these events should be grouped from line to line to help form continuous strokes. The second step, henceforth referred to as the "coding" problem, defines and implements a simple and efficient code for representing the strokes. Stroked head-and-shoulder images are presented at the end of this chapter. These images begin to show the limitation of morphological coding.

4.1 GLOBAL ALGORITHM FLOW

The approach taken to solve the two-dimensional morphological coding problem relies upon close interaction between the grouping and coding algorithms. The entire process begins

with an initial one-dimensional coding representation of each scan line. As shown in Chapter 3, the one-dimensional coding produces a list of gaussians, edges, and background "fat" gaussians. The latter are fixed in position and width and vary only in amplitude. Edges are composed of a pair of offset gaussian-peaks of opposite sign. The gaussian-peak list initially contains all gaussians, including those which have become paired as edges. As it would seem, this list contains a certain amount of redundant information. To understand why the gaussian list has this form, we must describe the three pass grouping, coding and correction process.

Edge events are passed to the edge grouping program. This program examines the correlation between edge events and groups them into strokes. Strokes are vertical runs of events on scan lines. No stroke may contain gaps without events. However, the edge grouping program may introduce new events to satisfy this criteria. That is, if two events which are not on consecutive lines are considered highly correlated, the edge grouping program will introduce interpolated events on the lines separating the two. In addition to adding events, the edge grouping program will discard a certain number of edges which are inconsistent with any major stroke. The edge coding program represents each edge stroke by a linear or quadratic interpolation of each of the three

stroke parameters. The resulting codes are not modified once created and may be transmitted or stored at this time.

Because the edge codes do not exactly represent the original edges, it is necessary to incorporate the changes in the original gaussian list. A regeneration program takes the edge codes and produces the approximation of the edge events created by the edge grouping program. Using the resulting events of the regeneration program along with the initial edge events created by the one-dimensional coding program, a hybrid edge list is created for correction purposes. All of the initial edge events which were not discarded by the grouping program are placed on this list. The width and position of these events are the same as initially, but the amplitudes are from the new list of regenerated edges. All of the new edge events from the regeneration program (i.e. those created by bridging gaps) are placed on the hybrid list with their own width, amplitude and position.

The hybrid edge list serves as a guide to correct the gaussian event list. As one will recall, the gaussian list contains all gaussians, even those paired as edges. In most cases, the paired gaussians will disappear after correction for edges since the corresponding edges will cause cancellation. In other cases, the pairs will remain if the edge was discarded or substantially shifted due to coding error.

The remainder list of gaussians is passed to the gaussian grouping and coding programs. In theory, these programs run in an identical fashion to the edge grouping and coding programs. Some gaussian events will be added, some discarded, and a certain degree of error will be introduced by interpolative coding. Nevertheless, the gaussian codes are fixed upon creation and may be transmitted or stored. As before, a regeneration program produces an approximation of the events retained by the grouping program. This list, together with the edge list, serves as a guide for the correction of fat gaussians. The corrected list of fat gaussians does not need to be grouped, since they already form seven vertical strokes. However, they are interpolatively coded similar to the edge and gaussian strokes.

After all of the above is done, three code sets will have been created: one each for the edges, gaussians, and fat gaussians. Constructing an image from the codes begins by regenerating the events from all three code sets. Next, the events are reconstructed on each line as the actual pixel values. The reconstructed image is then ready for display.

4.2 GROUPING EVENTS INTO STROKES

4.2.1 Philosophy

The problem of grouping events into highly correlated strokes is very similar to the track-while-scan concept in radar surveillance. In track-while-scan (TWS) systems, several airborne targets are simultaneously tracked by a computer using periodic information from a pulsed radar [22]. The radar system provides an estimate of target positions (and perhaps velocity) based upon an analysis of the return signal from a given output pulse. Because targets move slowly in comparison to the radar pulse rate, it is clear that the set of return pulses from consecutive output pulses will be highly correlated. Consequently, a smoothly varying prediction track function can be created for each target based upon the function data from past pulses. By extrapolating the function into the future, the TWS computer can estimate the position of the target after the next radar pulse. When the radar actually gives the next set of positions, the target position which most closely correlates with a particular estimate will be incorporated in the appropriate track. The next estimate will use the new point along with the previous. It is entirely possible for a target to have a very small cross-

sectional area or be sufficiently far away that its return pulse will not cause a "hit" (i.e. the output of the matched filter does not exceed the fixed threshold). An intelligent TWS system will allow for a "miss", and merely extrapolates to the next pulse return position.

Considering horizontal image scan lines as return signals from a pulsed radar illustrates the similarity between the track-while-scan problem and the grouping problem in morphological coding. The scan lines contain event pulses which are highly correlated from one line to the next. We are attempting to group correlated events into strokes or in TWS terminology, into tracks. While the similarity between the two problems is striking, the differences should not be ignored. In particular, we must consider four major distinctions:

(1.) Intersymbol interference is a much greater problem in the morphological image coding case. Therefore, one cannot assume that the noise is mostly white gaussian as in the track-while-scan example.

(2.) The "hits" are characterized by position, amplitude and width rather than just position as in the radar case. This turns out to be an advantage since we have more information from which to differentiate the targets.

(3.) The morphological strokes do not necessarily

persist through many lines (as shall be seen below) as do the tracks through many return signals. Strokes may begin and end within a matter of five or so lines. Having the possibility of many short strokes is a major disadvantage since predictive tracking estimators would have so little information from which it can predict the next event.

(4.) In morphological image coding, all of the scan lines are available for stroking at the same time. This means that strokes can be formed by tracking upward or downward. In radar, the TWS system does not have this luxury, because tracks must be produced in real-time without a substantial delay.

In addition to the fundamental differences cited, we also had the desire to develop an event grouping algorithm that is very efficient from a computational standpoint. Without an efficient algorithm, the morphological coding method could never hope to be implemented as a real-time system.

4.2.2 General Approach

It was recognized early that the most important strokes in an image would be those having the most energy. To prevent less important strokes from capturing events from an important stroke, it was decided that the high energy strokes should be

established first. Unfortunately, there is no way of knowing the energy of a stroke until it has been created. However, because events are well correlated within a stroke, one can obtain an indication of the total energy of a stroke from the energy of a single event belonging to it. With this in mind, we decided to begin new strokes from the highest energy event which has not yet been captured as a member of another stroke. The grouping process thus begins by ranking all of the events by their energy.

Having established an initial event for a stroke, it is necessary to begin tracking other events in both directions. Realizing that strokes can be rather short and that we need efficiency, we chose a zeroth order tracking algorithm. It always expects that the next event will appear on the following line with the same amplitude, position and width as the present event. This simplifies the problem of determining the next event considerably. Provided that such an event is found whose correlation exceeds a minimum threshold, it becomes a member of the stroke and the process continues. We always begin by stroking an event in the downward direction. Once the correlation threshold is exceeded, the stroke terminates on the downward side and stroking begins in the upward direction from the initial stroke event.

The single most important issue in this process is how to measure the correlation between two distinct events. As mentioned in earlier discussion, each advent can be described by its line number, horizontal position, width and amplitude. This applies for both edges and gaussians. Unlike the TWS case, we cannot expect geometric separation to be the best estimate of anti-correlation or "distance". Instead, we must hypothesize a combined distance measure which accounts for anti-correlation in the geometric, amplitude and width senses. The geometric distance is a function of the line numbers and horizontal positions of the two particular events. Similarly, amplitude and width distances are functions of the individual event amplitudes and widths. Under the assumption that the geometric, amplitude and width distances are independent functions, we write

$$TD^2 = C1 \cdot GD^2 + C2 \cdot AD^2 + C3 \cdot WD^2$$

TD: Total Distance
GD: Geometric Distance
AD: Amplitude Distance
WD: Width Distance
C1, C2, C3: Normalizing Constants

Without knowing exactly how important each distance measure is to the total, we have introduced scaling constants for all three.

Assuming that differences in line number and in horizontal position contribute with equal importance to the geometric measure, we selected the simple and intuitively derived form:

$$GD = \frac{(X2 - X1)^2 + (Y2-Y1)^2}{A2 \cdot A2 \cdot W2}$$

(X1,Y1),(X2,Y2): Coordinates of the selected events.
A2: Amplitude of the second event
W2: Width of the second event

If two events are at equal distance from the source event but have different energies, this distance measure will select the larger.

As for the amplitude and width function, both were selected to estimate fractional changes

$$AD^2 = \frac{(A2 - A1)^2}{(A2 + A1)^2}$$

$$WD^2 = \frac{(W2 - W1)^2}{(W2 + W1)^2}$$

Absolute change is undesirable because it would result in an unfair treatment of the larger width and amplitude events. For example, two events having amplitudes of 0.9 and 0.8 would

be, with an absolute measurement of change, assigned an amplitude distance ten times greater than the similar pair of events having amplitudes 0.09 and 0.08. One can imagine the same situation arising in width. The proposed formulas counteract this effect by dividing by the mean amplitude or width so as to measure a fractional change.

One would expect that the next event to be chained to the end of a stroke would be fairly close in a physical separation sense. After all, a sufficient geometric distance alone can cause the total distance measure to exceed the maximum threshold. To increase the computational efficiency, we have limited the region over which the distance measure is performed to a ten pixel hemisphere centered on the last event in the stroke. The hemisphere points upward or downward according to the direction of travel. Once no events are found in the hemisphere which satisfy the distance bound, the stroke is terminated in that direction. In practice, it was realized that two events should never be grouped which come close to a ten pixel separation. Thus, we did not believe that the hemisphere would in any way perturb the desired results of the distance measure alone.

After the distance threshold is exceeded at both ends of a stroke, a new stroke is begun from the highest energy event which has not yet been used by another stroke. This

process continues until a certain fraction of the total events have been used, after which no new strokes are formed. To justify such a procedure, we must remark that a large number of the low energy events passed from the one-dimensional coding program were found to be inconsistent with surrounding events and hence characterized as noise. By discarding a portion of the low energy events, one would not expect a notable reduction in image quality. If such events were retained, coding efficiency would suffer tremendously since most would form singular event strokes.

Because the strokes are not created simultaneously, it was realized that strokes could develop the bad habit of "stealing" events from a stroke that has not yet been formed. By allowing a stroke to steal events from a future-stroke, the future stroke might be broken into two strokes and cause a reduction between two events (of the stealing stroke) would be filled with events which are inconsistent with the image. To deal with this problem, it was decided to form the strokes with two passes of the grouping algorithm. The first pass, as described above, establishes all of the strokes in the image. The distance threshold is set fairly low on the first pass so that the strokes will not grow to their fullest extent. On the second pass, the distance constant is greatly increased. No new strokes are formed on

this pass. Rather, the old strokes are permitted to continue growing. Since all of the strokes are already well established, the problem of stealing has been reduced. It could be further reduced by multiple passes of this type, although the computational penalty would not be justified.

During the actual event grouping process, gaps (i.e. lines without events) may have been left between successive events in a stroke. To maximize code efficiency, gaps were usually filled with interpolated events. Some exceptions were later made when the ratio of the length of the gap to the length of the entire stroke exceeded a preset threshold. In that case the stroke was broken into two at the gap.

The above description has summarized the initial approach taken to group both edge and gaussian events into strokes. The two sections which follow describe the results of applying this approach to both types of events and the changes that were necessary to achieve satisfactory performance.

4.2.4 Results of Edge Event Grouping

The results of grouping edge events into strokes were very encouraging. The initial approach algorithm described in the previous section provided excellent performance. By

varying only the threshold and distance constants, we solved the following problems which arose:

(1.) Connecting edges of opposite amplitude.

(2.) Eliminating obvious strokes entirely.

(3.) Describing an obvious stroke as two separate strokes.

(4.) Leaving off distant end points on a stroke.

(5.) Jumping over existing events in strokes and replacing them with interpolated events.

For example, to prevent edges of opposite sign from connecting, the constant affecting the amplitude distance measure must be increased. As expected, this brought an immediate cure. The second problem was due to the fact that too few edge events were included during the first pass of the grouping algorithm. That is, the program had eliminated low energy strokes that were actually quite consistent. By increasing the first pass percentage of retained events, such strokes were saved at the expense of adding a number of singular event strokes, which were later discarded.

By correcting the gaussian list from the list of grouped and coded edges, (see next section for details of the coding process) we were able to reconstruct images. The intention was to examine any side-effects of edge interpolation, elimination and the like. While the images were actually quite good, we did notice the following interesting phenomena:

(1.) Edges were in some places unnaturally sharp due to interpolation.

(2.) Coding error casued certain smooth edge strokes to become angular. This was noted at the side of the subject's head.

(3.) Interpolating seemed to cause minor changes in image features. For example, the subject's nose was "straightnened".

Overall, we were content with the results and proceeded toward gaussian event grouping in the same manner.

4.2.4 Results of Gaussian Event Grouping

We corrected the gaussian list directly from the grouped and coded edge list. The corrected gaussian list was then itself grouped and coded. The fat gaussian list was left as it had originally been created. All three lists together served as a reconstruction guide. The results were less than impressive. A great number of problems and solutions were examined.

With the interest of efficient computation, the edge events were sorted in energy by coarsely setting linear ranges and selecting the appropriate range for each event. Edge event energy was approximated by amplitude squared. It turned out that the probability density function of energy

for the edge events was roughly uniform. Therefore, the linear sort gave decent results. We tried applying a coarse linear sort to gaussian events and ran into trouble. Gaussian event energy was approximated as amplitude squared times the width at the half amplitude points. An examination of an energy histogram revealed a very non-uniform probability distribution. Sorting in a linear fashion caused most of the events to fall in the smallest range. Hence, the events were barely sorted at all. To counteract this problem, the ranges were set in exponential proportions. Furthermore, because there were so many more gaussian events than edge events, we increased the total number of ranges by a factor of five. The resulting sort placed roughly an even number of events within each range as desired.

We began to question the validity of the geometric distance measure because it provides equal treatment to vertical and horizontal displacement, while the two are fundamentally quire different. By examining charts of coded gaussian events, it appeared that grouping errors could be attributed to the circular shape of the geometric distance function for a fixed value of distance. For example, it intuitively seemed that events which are straight down should have a closer geometric distance than those off to an angle. We began to view the stroking problem as guiding a horse down a path. Perhaps the horse needed blinders to keep from

losing its way. The extreme "blinder" function was identified as the Lemniscate of Bernoulli. We desired to vary the shape of the constant geometric distance function between the circle which we presently had and the degenerate Lemniscate. The resulting function is called an Oval of Cassini [23]. The geometric distance formula thus became:

$$GD^2 = \frac{Q^2(X^2 - Y^2) + \sqrt{X^4 + 2X^2Y^2(1-Q^4) + Y^4}}{(1 - Q^4)(A2^2)}$$

Where $X = X2 - X1$
$Y = Y2 - Y1$
$0 < Q < 1$

A value of Q equal to zero provides the same geometric distance measure as before, while at Q equal to one, the function is purely Lemniscate.

The Q constant was allowed to take on several values in the range of zero to one. At each value, image quality was compared to an image created using the original distance measure. For values of Q near one, the resulting images were noticeably inferior to the original. As Q decreased, quality gradually increased. In the range of Q between 0.0 and 0.6, no difference in quality was apparent in comparison to the original. This test suggested that the Oval of Cassini geometric distance measure is worthless. Nevertheless, it

remains incorporated in both the gaussian and edge grouping programs in case further research in this area is warranted.

The notion of dividing the geometric distance by amplitude squared was then put to a test. We decided to examine whether the concept of tracking the highest amplitude events was not in fact forcing the strokes to accept decorrelated events just because of their amplitude. Isolated cases seemed to confirm this belief. As a test, the amplitude bias was entirely removed from the geometric distance measure and a suitable constant served as a substitute. This constant was adjusted until the total number of strokes created roughly equalled those using the amplitude bias. A comparison of image quality revealed certain improvements and degradations. Overall quality seemed a bit worse, especially in regions of detail. Once again, it appeared that the original grouping procedure was correct.

For wide gaussian events, it was noticed that interpolation errors were quite frequent. Events on adjacent lines were often skipped because of excessive horizontal error. It is known that the error in estimating the position of a gaussian is directly proportional to the width of that gaussian. Clearly, this fact should have played a role in determining the geometric distance function. Specifically, the horizontal displacement between two events should be

divided by the width of the present event and the result should be appropriately scaled. To allow variable testing of this theory, we made the following modification to the geometric distance function:

$$XD = \frac{(X2 - X1)}{W1^{V}}$$

$$GD = f(XD, Y2-Y1)$$

The V constant was allowed to run from 0 (which represents no change from before) to 1. As V increased, the errors noticed above began to disappear. Image quality seemed best at a value of around 0.8. Hence, this modification actually produced a noticeable increase in quality.

Interpolation errors continued to seriously degrade the image quality. It was decided that the grouping program should not be permitted to jump over a substantial number of events in any one stroke. Unlike the case of edges where improper interpolation can be almost entirely compensated for when the gaussian list is corrected, the only safeguard for improper gaussian interpolation is the correction of fat gaussians. Because the fat g's are fixed in position and width, the correction process is not nearly as complete. The first step to reducing interpolation errors was to truncate the searching hemisphere so that only events within four lines

of the present would be considered as next event candidates. This alone seemed unsatisfactory because strokes could still be largely composed of interpolated events. Prior to the interpolation process, it appeared that some type of validity check could be performed upon a group of events purported to represent a stroke. If a group of events failed the test, it would be appropriately broken and the two halfs would each be checked. The process would continue until all groups of events passed the stroke validity test.

The probability that a group of events represent a stroke can be thought of in terms of a signal to noise ratio within the proposed stroke. Any events interpolated to form the stroke are noise since the grouping program assumes that they were omitted erroneously by the one-dimensional coding program. The signal can be thought of as all of the proposed events in the stroke (i.e. originals and interpolated events). With this in mind, an appropriate check would be to sum all of the interpolated energy, divide it by the total energy, and determine whether the quotient exceeds a minimum threshold. As a rough trial of the idea, we assumed all events within a stroke to be of equal energy and merely took a ratio of the number of interpolated events to the total. If the ratio was less than the threshold, the group was split at the largest gap and the process continued. Image quality improved considerably.

Soon after the above experiment, another was performed to determine image quality if no interpolation were allowed with gaussian events. That is, the searching hemisphere was truncated to a height of a single pixel. Only events on the very next line would be considered as continuations of a stroke. The results were surprising. Image quality improved dramatically. Although the number of singular event strokes increased, the coding program discarded them anyway. We concluded that the grouping program does not have the knowledge necessary to decide which events are omitted in an image. In more cases than not, adding events produced more error than it cured.

4.3 THE CODING PROBLEM

The coding problem can be viewed as two distinct problems. The first is the problem of obtaining a simple way to represent the strokes while preserving their important qualities. In general, the qualities that must be preserved are not the exact details of the parameter values of each event, but the general tendencies of these parameters over the length of a stroke. Because of this, we can represent a stroke by fitting simple functions to the parameter values (position, amplitude and width). The second problem is the actual coding of the data that will be transmitted. This coding must be done in a manner

that minimizes the amount of data that is needed for transmission, while still maintaining the desired image quality.

4.3.1 Fitting Algorithms

The basic operation of the coding process involves independently fitting simple functions to the position, amplitude, and width for each stroke in the image. In general, these parameters are fairly well correlated from event to event within the same stroke. The small variations from line to line emerging from the one-dimensional process are not important for image quality, however, and in fact can even degrade the quality of the image if reproduced accurately. All of the useful information about the image is generally contained only in the overall tendencies of the parameters in the stroke. For this reason it is reasonable to code the strokes by fitting the parameters with simple, smooth function. Polynomial functions were chosen for this purpose because of the simplicity of polynomial fitting algorithms and because of the ease with which they can be regenerated to form the strokes at the receiving end.

The fitting process has been tried using many different variations. However, the general approach to the problem has remained nearly the same in all of these variations. The

general fitting algorithm is shown below. This is done separately for the position, amplitude, and width of every stroke.

1. Fix the beginning of the polynomial function to the first event in the stroke.
2. For each polynomial segment, the fit begins using the smallest possible number of events.
3. Fit the polynomial using a mean squared error fit.
4. Check the distance from each event to the fitted polynomial. If any of these distances are larger than the pre-selected error bound, then break the segment at the end of the polynomial last fitted, fix the beginning of the next polynomial segment to the end of the last, and then continue from step 2.
5. Extend the number of events to be included in the next fitting operation by one.
6. If all of the events in the stroke have not been used then continue from step 3.

In simple terms, the algorithm fits a longer and longer polynomial to the events until the maximum error between the polynomial and any event is more than a pre-selected bound. It then breaks the segment and starts a new segment connected to the end of the last.

Two basic aspects of the algorithm have been implemented with several variations. One was the method of fitting. Both first order (linear) and second order polynomial fitting were tried, each with two different mean squared fitting methods. The second aspect of the algorithm that was varied was the distance measure used in determining the maximum error value for breaking a segment.

First, the simplest type of fitting algorithm was used. A first order mean squared error fit was used along with a simple vertical measure for the maximum error. The process used for the linear fit involves finding only the slope of the line since it is assumed that the beginning point is fixed at the origin. The equation to find the slope is:

$$m = \frac{\sum_i X_i Y_i}{\sum_i X_i^2}$$

In this case X is the number of each event along the stroke and Y is its corresponding value: either position, amplitude, or width. The error measure used for the segment breaking was the vertical distance:

$$e_i = |Y_i - mX_i|$$

It seems apparent that this simple measure of error from the fitted line is not necessarily the most appropriate one to use. By using this measure, the cases where there may be a generally large slope over a portion of a stroke (for any of the three parameters), will be fitted much more closely than cases where there is a very small slope. While this is desirable in the case of accurate curve fitting algorithms, it is not desirable for coding purposes. The tight following of a large-sloped section increases the number of segments needed to fit this section over the number needed to fit a similarly shaped section with a small slope. Since it increases the number of segments it also increases the number of codes needed to describe the stroke. To account for this problem, a different error measure was chosen. It was decided that the amount of error measured from the fitted curve to the events should be independent of the overall slope of the curve, thus implying a perpendicular error measure. The equation for error used was therefore of the form:

$$e_i = \frac{mX_i - Y_i}{\sqrt{m^2 + i}}$$

The equation for the slope using a mean squared perpendicular error fit was derived and found to be the solution of a quadratic equation. Since, in general there are two

different solutions to a quadratic, the algorithm was modified to include a test to determine which of the two results gives the correct slope. The equation for the slope in this type of fitting algorithm is shown below:

$$m = \frac{R - Q \pm \sqrt{Q^2 - 2QR^2 + R^2 + 4S^2}}{2S}$$

Where:

$$Q = \sum_i X_i^2$$

$$R = \sum_i Y_i^2$$

$$S = \sum_i X_i Y_i$$

This fitting procedure was implemented and did produce slightly better fits. However, the overall performance was still very poor, especially considering the large number of segments needed to fit each stroke. We therefore decided to fit second degree polynomials to the strokes. Again, in the first attempt, the choice was to use an ordinary mean squared error fit algorithm and a vertical maximum error measure. A second degree polynomial with the end point fixed at zero has the equation:

$$Y = aX^2 + bX$$

The mean squared error fit involves finding a and b as shown below:

$$a = \frac{\sum_j Y_j [X_{ji}^2 \sum_i X_i^2 - X_{ji} \sum_i X_i^3]}{\sum_i X_{ii}^2 \sum_i X_i^4 - [\sum_i X_i^2]^2}$$

$$b = \frac{\sum_j X_j [X_{ji} \sum_i X_i^4 - X_{ji}^2 \sum_i X_i^3]}{\sum_i X_{ii}^2 \sum_i X_i^4 - [\sum_i X_i^3]^2}$$

This fitting method, albeit more complex, produced much better results. It was able to fit the general tendencies of the strokes very closely with very few breaks. In this case the vertical error,

$$e_i = Y_i - aX_i^2 - bX_i$$

was used to determine the break points. Again, it seemed as if there may be some improvement by attempting to use a perpendicular error measure. The perpendicular error measure, in this case, was determined to be a solution to a third degree equation. The sclution to a third degree polynomial is relatively complicated and therefore very computationally inefficient. Also, since there are three solutions, each must be tested to find the actual minimum distance. Because of this added complexity, it was considered not worthwhile to

use this distance measure.

The final algorithm that is presently being used is a second order mean squared fit with a vertical distance measure. There are still a number of problems with this method, however. These will be discussed in detail in Chapter 5.

4.3.2 Coded Data Format

After all of the strokes have been fitted with first or second order polynomials, they must be coded in a simple form that is both efficient and relatively easy to decode at the receiving end. The codes are put into three distinct groups: edges, gaussians, and fat gaussians, and are coded independently. Within each of these groups, each stroke is specified completely before moving on to the next. The first piece of information that is given for a stroke includes the starting values for the X and Y position, the amplitude and the width. All subsequent values are obtained from the polynomial functions that are described relative to these initial points. Following these starting values the function descriptions for position, amplitude and width, repsectively, are given. Each parameter is defined completely over the entire stroke before moving on to the next one. In the case of a linear fit, these

descriptions consist of a list of lengths and slopes of the individual segments over the entire stroke. The exact format that has been used in this case is shown in the table below.

Start of stroke ---	Starting Y
	Starting X
	Starting A
	Starting W
	Length of first X segment
	Slope of first X segment
	Length of second X segment
	Slope of second X segment
	.
	.
	.
	0 (Zero)
	Length of first A segment
	Slope of first A segment
	Length of second A segment
	Slope of second A segment
	.
	.
	.
	0 (Zero)
	Length of first W segment
	Slope of first W segment
	Length of second W segment
	Slope of second W segment
End of stroke ---	0 (Zero)

For the fat gaussian functions that are coded, this format can be simplified considerably. For these functions, the starting values for X, Y, and W are always known, and there are no charges in position of width along the stroke. The only variable that must be coded is the amplitude. The

code that is used is a simplified verison of the one used in the other cases.

In the second order fitting routine, this coding format was modified slightly. In this case, of course, two parameters must be sent for each segment. In order to simplify the event regeneration process, the parameters that were chosen to be sent are the initial slope of the segment and the second derivative, or acceleration. This simplifies the event regeneration process since a simple recursive procedure can be used to reconstruct the polynomials without the need for a squaring operation. The code that is sent is identical to the one for a linear fit except for the addition of an acceleration term that is added after each of the slope terms.

In the coding system that is presently being used, all of the codes that are sent are real numbers. Obviously, in any real coding system all of these values would be quantized and coded with a finite number of bits. The quantization should be done so that the data can be coded with the fewest number of bits needed to prevent significant degradation of the image.

4.4 Representative Results

This section presents results of the image-stroking process discussed in this chapter. These results begin to

show the limitations of this type of morphological coding, especially in regards to its application in image bandwidth compression.

Figure 4.1 shows charts of stroked and unstroked edges and gaussians for the image in Fig. 4.2. Letters occur at central positions of events in this figure. Capital letters indicate positive amplitudes; small letters, negative amplitudes. The later the letter falls in the alphabet, the higher the amplitude. The scale is on the order of ten amplitude units (out of a maximum of 255) for each letter. The image of Fig. 3.4 is shown stroked in Fig. 4.2, with several aspects of the stroking algorithm varied towards improving the image quality. Examples of the varied techniques are: whether to fill stroke-gaps with interpolated events, method of reconstructing the fat-g's (see section 2.3), and the degree of smoothing the amplitude and/or the width within the same stroke. The stroked image in Figure 4.2(e) was reconstructed at a bit rate of $\sim$ 0.5 bits/pixel and a rms error of 7.5 (units out of 255). This image serves as a canonical indication of the capability of the present morphological coding algorithm of producing stroked two-dimensional half-tone facial images. The causes hampering the achievement of better-quality images at lower bit rates are discussed in detail in the next chapter.

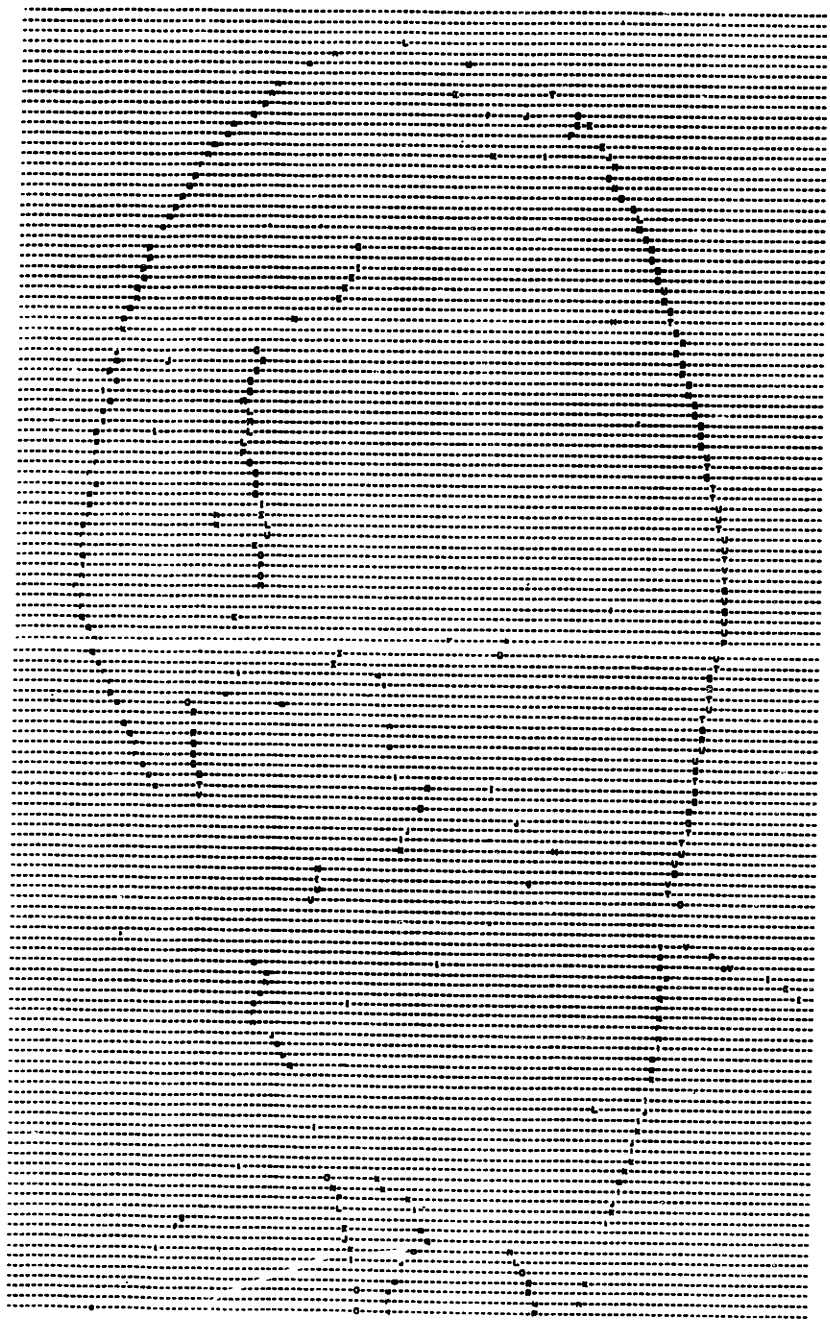


Figure 4.1 (a) A chart of unstroked edges.
(i.e. before processing by the stroking routine)

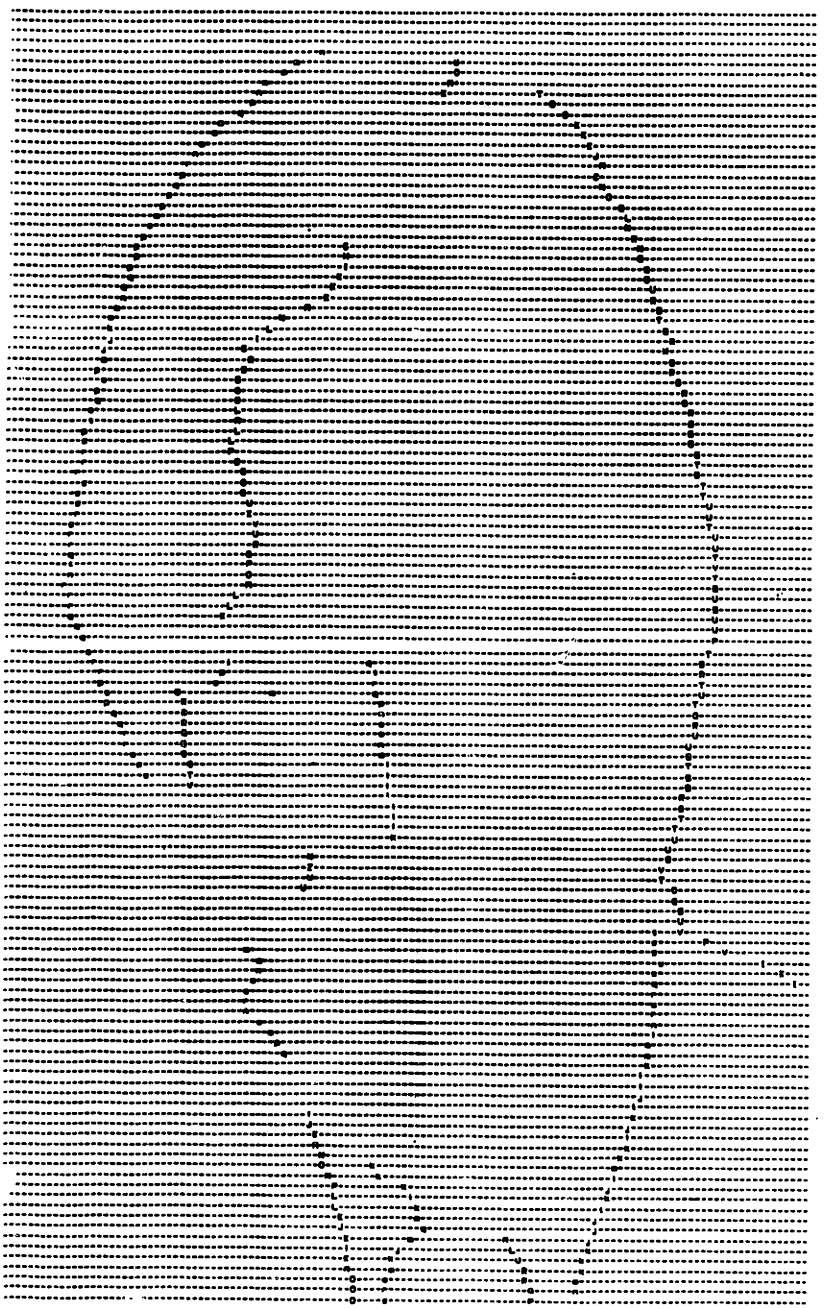


Figure 4.1 (b) A chart of stroked edges.
(i.e. after processing by the stroking routine)

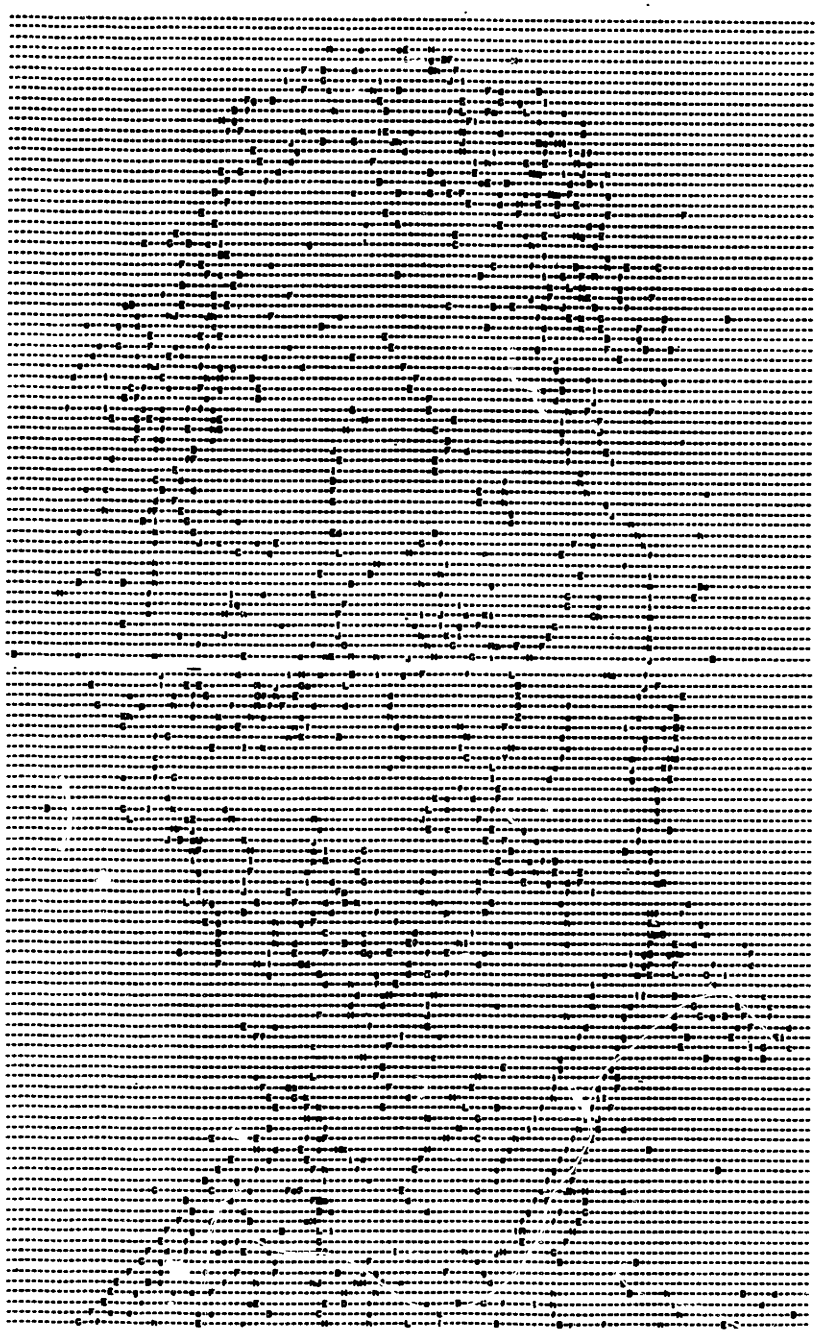


Figure 4.1 (c) A chart of unstroked gaussians.
(i.e. before processing by the stroking routine)

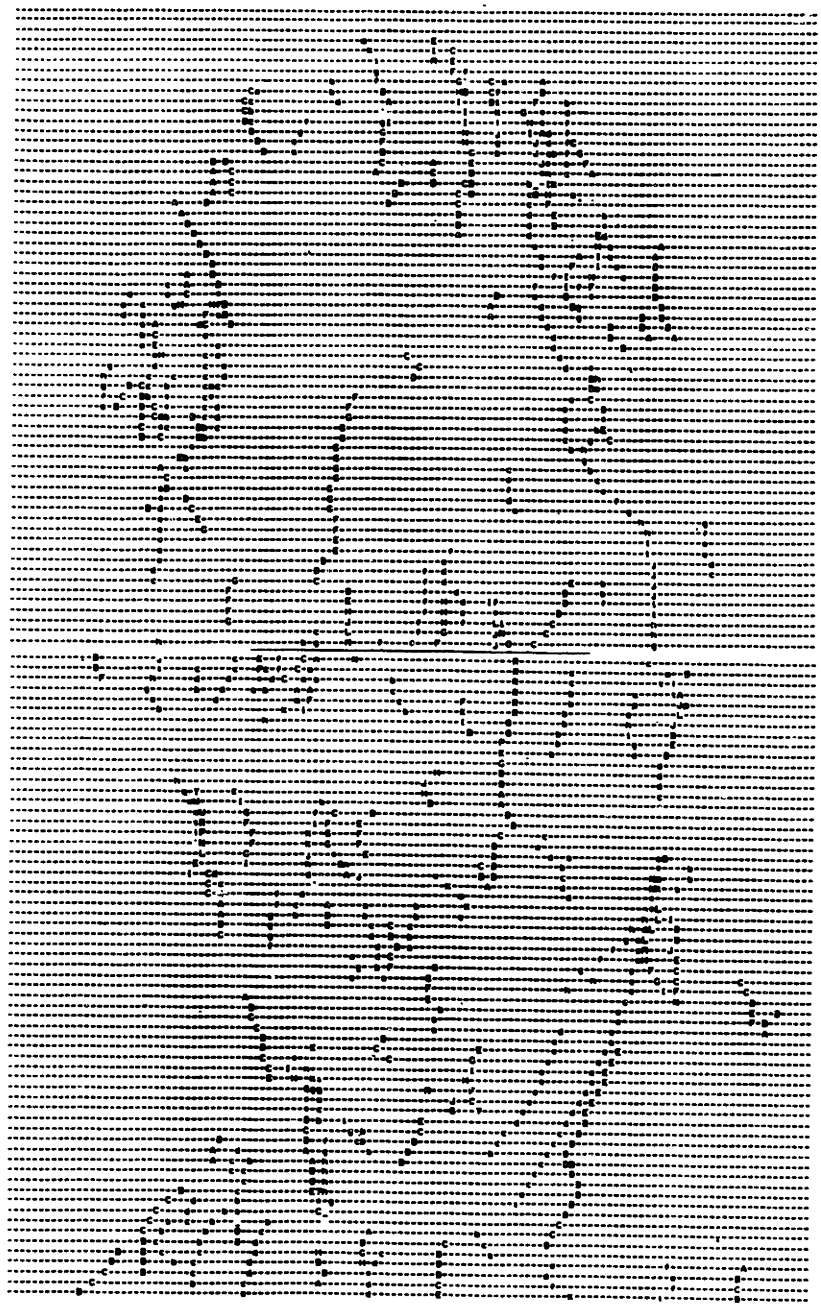


Figure 4.1 (d) A chart of stroked gaussians.

(i.e. after processing by the stroking routine)

Figure 4.2 (a) An early attempt at stroking an image. Stroke-gaps of up to nine lines are filled with interpolated events. No energy bias (see section 4.2.4) is used in correlating event position. Single events are not included. The first method (see section 2.3) of reconstructing fat-g's is used.

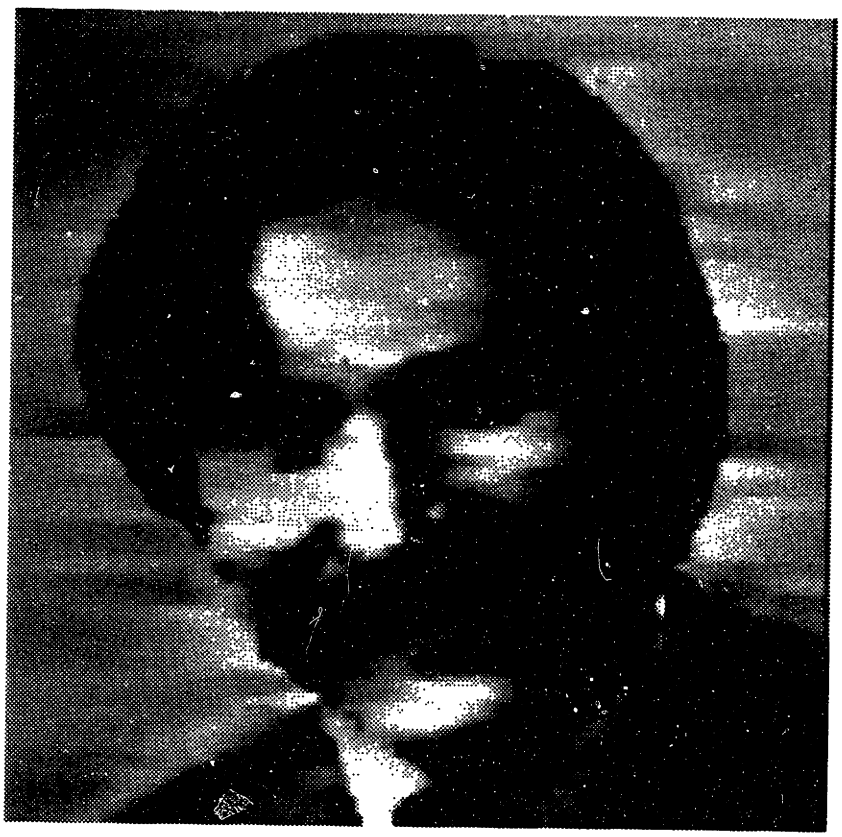


Figure 4.2 (b) Same as Figure 4.4(a) except a simple median filter is used. While less horizontal streaks are noticeable, the overall quality is still unsatisfactory.

Figure 4.2 (c) Same as Figure 4.2(a) except the amplitude is smoothed with one parabola within a stroke, the energy threshold for including sorted events is increased by 10%, and stroke-gaps of only up to 4 lines are filled with interpolated events.

Figure 4.2 (d) Same as Figure 4.2 (c) except both the amplitude and the width are smoothed with one parabola (each) within a stroke.

Figure 4.2 (e) Same as Figure 4.2(d) except the energy threshold is the same as Figure 4.2(a), the second method of reconstructing the fat-g's is used, energy bias for correlating event position is used, gaps of up to 4 lines are filled only if gap ratio (see section 4.2.4) is less than 15%.

CHAPTER 5

STUDYING THE NOISE (IMPERFECTIONS) IN THE RECONSTRUCTED IMAGES

Noise in the morphologically coded images presented in the previous chapters can be classified into two major categories :

1) One-dimensional noise, and

2) Two-dimensional noise.

This chapter defines and studies the different types of imperfections of each of the above two categories. Each type is studied in as much detail as is allowed by its separability from the other types. Due to the complexity of the problem at hand, it was decided that a detailed understanding of the imperfections in the reconstructed images can best be gained through processing controlled test patterns. This approach is followed in the sections below.

5.1 One-dimensional noise

One-dimensional noise is defined here as the imperfections in the reconstructed image resulting during the stage of coding each image line separately using the one-dimensional morphemes, excluding the imperfections defined below in section 5.2. All of the types of imperfections under

this category can be traced to one major cause, namely the non-orthogonal nature of the one-dimensional morphemes used here and the consequent significant overlap among morphemes; in other words, it is a problem of strong inter-symbol interference.

The one-dimensional reconstruction of each image line involves estimating three parameters for each detected morpheme. Therefore, the problem of understanding one-dimensional noise, given a detected morpheme, reduces to determining the effect of inter-symbol interference on the estimation of each of the three morpheme parameters; i.e. we need to determine errors in estimating the morpheme's position, width, and amplitude.

In the following subsections, the effect of missed (i.e. undetected) morphemes, as well as the behavior of each of the above three types of errors for the detected morphemes, is observed through processing controlled test patterns. Most of the test-pattern runs were performed once for the case of using the "fat-g's" introduced in chapter 3, and again for the case of not using them. This helps illustrate the value of the fat-g's, especially in retrieving low frequencies. The reconstruction error, as dicussed below, is the rms difference between original and reconstructed lines, computed over the entire line. Unless otherwise noted, all processing was done at a detection threshold of 14.

5.1.1 Reconstruction error as a function of position

Reconstruction error is a function of position only so far as the relative position of each morpheme with respect to other morphemes (i.e. the separation between morphemes) is concerned. In other words, if we ignore for the moment the slight variations introduced by line-end effects (see chapter 3) the manner in which the algorithm acts on each morpheme, and therefore any resulting reconstruction error, is the same regardless of the morpheme's absolute position on the image line. The dependence of reconstruction errors on event separation is studied here by fixing the events' amplitudes and widths in the test pattern, while varying their separation and interpreting the resulting reconstruction errors. To understand the intricacies of the algorithm's operations, it is deemed necessary to resort to a walk-through style of explaining the "nooks and crannies" in the resulting error curves as shown below.

We start with a test pattern consisting of only two gaussians. Fig. 5.1 shows the reconstruction error, with and without the use of fat-g's, as a function of the separation between two events having a fixed half- amplitude width of 17.8 pels and a fixed amplitude of 100 (out of a maximum of 255). The error curves were computed at increments of one pel for separations ranging from 0 pels to 88 pels. It is clear that for event separations greater than 40 pels, the

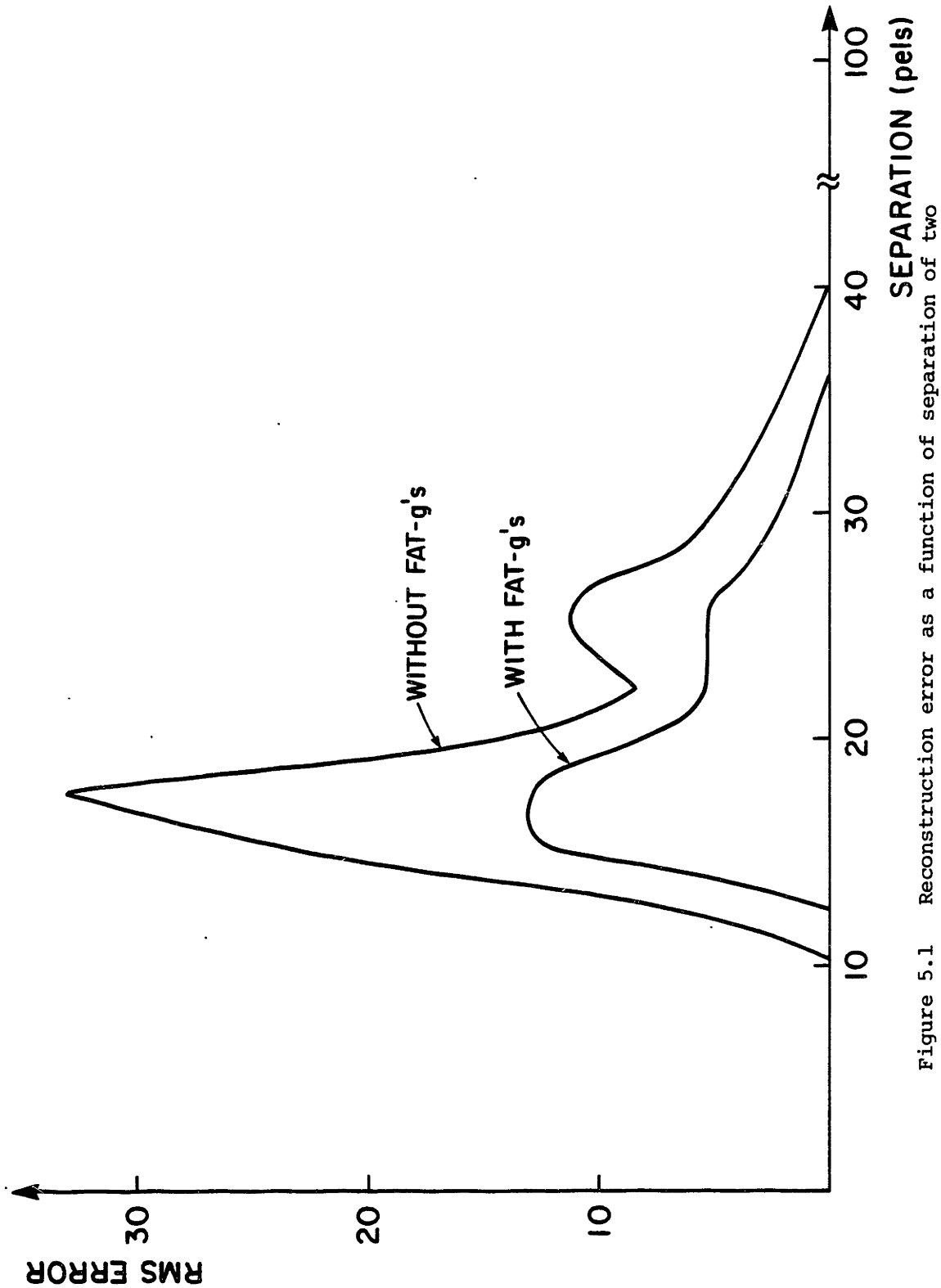


Figure 5.1 Reconstruction error as a function of separation of two events (gaussians) having constant width (17.8 pels) and amplitude (100).

reconstruction is perfect, i.e. the error is zero, within the roundoff error of the algorithm. This is because at such wide separations the events don't overlap and, therefore, there is no inter-symbol interference to cause any reconstruction errors. As the separation between the two events falls below 40 pels, the overlap between them increases, and so does the reconstruction error.

The increase in the inter-symbol interference between the events in fig. 5.1 and the consequent increase in the reconstruction error continues down to a separation of 25 pels, at which point an interesting occurrence takes place: the estimation errors in two morpheme parameters (position and width) are of such magnitude as to compensate for (i.e. partially cancel) one another, leading to an overall drop in the reconstruction error. Specifically, the effect of inter-symbol interference on estimating the width at separations between 21 and 25 in fig. 5.1 is to make the width narrower. Counter-balancing that is the error in estimating the position of the two events : the separation between them is estimated to be higher than it actually is. For separations greater than 25 or less than 21, the above two errors diverge, i.e. cease to be of the right magnitudes to (partially) cancel each other.

At separations between 18 and 21 pels, the reconstruction error increases rapidly. This happens as the two events approach a merger into one single wider event, causing two adverse effects, 1) the disappearance, (i.e. by falling

below the detection threshold), of the peaks of the narrower (h3) of the two h-filters used to estimate the events, and 2) the rapid deterioration in the quality of position estimation of the peaks of the wider (h4) of these two h-filters. The error peaks at 18 pels, where the h3 peaks have completely disappeared leaving only one h4 peak right in the middle of the two events. This h4 peak uses its default width (as explained in chapter 3), which happens to be the same as the width of the events in this test pattern (i.e. 17.8 pels), for the reconstruction.

The error then falls to zero, (again, within the roundoff error of the algorithm), since reconstructing with a single event becomes more accurate as the two events gradually merge to become one event of width 17.8 pels and of amplitude equal to twice that of the original individual events , i.e. 200. In fact at separations less than 12 pels, h3 rejoins h4, (when the single event formed by the two original events become sufficiently narrow for h3 to detect), to form a pair resulting in a more accurate estimation than what's achieved with an h4 single.

The error curve for the case of using the fat-g's follows the general behavior of the error curve for the case of not using them, albeit at a lower reconstruction error for all separations. This shows the value of using the fat-g's. The contribution of the fat-g's to reducing the reconstruction error is most significant for separations in the range 12-21 pels. As mentioned above, in that range the two events start

to merge into a single wider event, causing the h-filters to fail to capture such a low frequency structure; it is this very kind of structure that the fat-g's are tailored for.

The next test pattern for studying the effects of event separation on reconstruction errors consists of three gaussians with a fixed width of 17.8 pels and a fixed amplitude of 80 (out of a maximum of 255). The middle event was fixed at a position in the middle of the line (pel no. 64), while the separation between the outer two events was varied by one pel at a time from zero, (i.e. all three events co-located at pel no. 64), to 88, (i.e. the events located at pels 20, 64 and 108). The resulting error curves are shown in fig. 5.2. The separation on the horizontal axis, and as referred to below, is that between the outer two of the three events in the test pattern.

Starting at the right-hand side, we notice a similar behavior to that of fig. 5.1 until a jump in the error curve occurs at a separation of 66 pels. This sudden increase in the error is caused by one peak of the pair used to estimate the middle event falling below the detection threshold. It is the peak of the wider filter (h4) that dropped out, leaving the peak of the narrower filter (h3) and suddenly forcing the use of the default width of h3, which happens to be narrower than the event's width of 17.8 pels. Hence the sudden increase in the reconstruction error due to the sudden deterioration in the accuracy of estimating the width of the

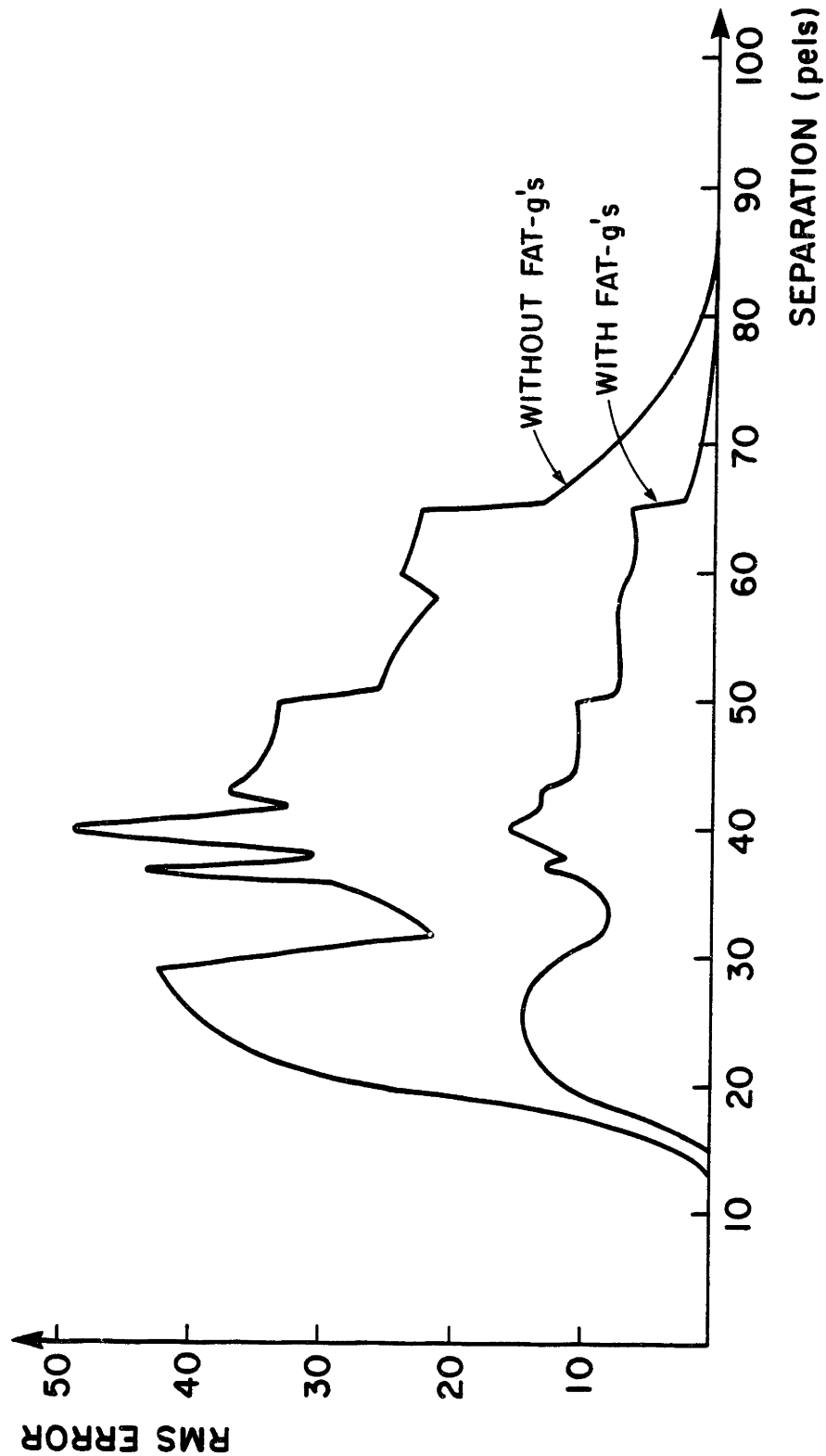


Figure 5.2 Reconstruction error as a function of separation of the outer two of three events (gaussians) having constant width (17.8) and amplitude (100).

middle event. It is reasonable to expect the h4 peak to fall below the detection threshold before the h3 peak does (i.e. as we follow the behavior of the curve from right to left), because the overlap of the negative sidelobes of the filter response to the outer events would pull down the middle peak in h4 sooner than it does in h3. This is due to the wider response of h4 making such overlapping effects take place at wider separations. When the h3 peak associated with the middle event drops out (due to the same reason its associated h4 peak dropped out earlier), it causes another jump in the error curve at a separation of 51 pels. The intervening temporary drop in the error for the separations between 55 and 60 pels is analogous to the error drop between 21 and 25 pels in fig. 5.1 as explained above.

The two narrow peaks in the error curve at separations 40 and 37 pels are caused when the h-filter-output peaks of the outer two events respectively drop below the detection threshold as they get pulled down by the negative sidelobes of the filter response to the middle event. As the events move closer to each other, the maximum points of these negative sidelobes pass by the peaks of the outer events. The outer events would then quickly reappear, causing the narrowness of the peaks in the error curve at 37 and 40 pels. It is for that same reason, i.e. the maximum point of the negative sidelobes passing by and the zero of the filter response to the middle event approaching the filter peaks of the outer events, that we see a general trend of improvement down to a

separation of 32 pels. This is due to the lower inter-symbol interference when the zero in the filter response to one event is close to the peak filter response to another event.

The last peak in the error curve of fig. 5.2 and its final decay resemble those of fig. 5.1 and can be explained in a similar fashion (see the explanation above for fig. 5.1). Just as we observed in fig. 5.1, the curve for the case of using the fat-g's follows the general behavior of the curve for the case of not using them, yet shows a reduction in the reconstruction error at all separations.

Many of the jumps in the error curves of fig. 5.2 are caused by filter peaks dropping below the detection threshold. Therefore, it was thought reasonable to try to eliminate, or dilute, that effect by performing the detection at a lower threshold. The result of processing (without the fat-g's) the test pattern of fig. 5.2. at a lower threshold, 7 vs 14 (out of a maximum of 255), is shown in fig. 5.3. The indications in fig. 5.3 are that while lowering the detection threshold may eliminate a few of the jumps (e.g. the jumps at 66 and 51 pels in fig. 5.2), and dilute others (e.g. compare those at 40 and 37 pels in fig. 5.2 with those of fig. 5.3), the error curves may still show a few such jumps (e.g. the jump at 56 pels in fig. 5.3). In fact the jump at 56 pels in fig. 5.3 has the identical cause as the jump at 66 pels in fig. 5.2, albeit postponed by the obvious fact that filter peaks stay above lower thresholds longer than they may stay

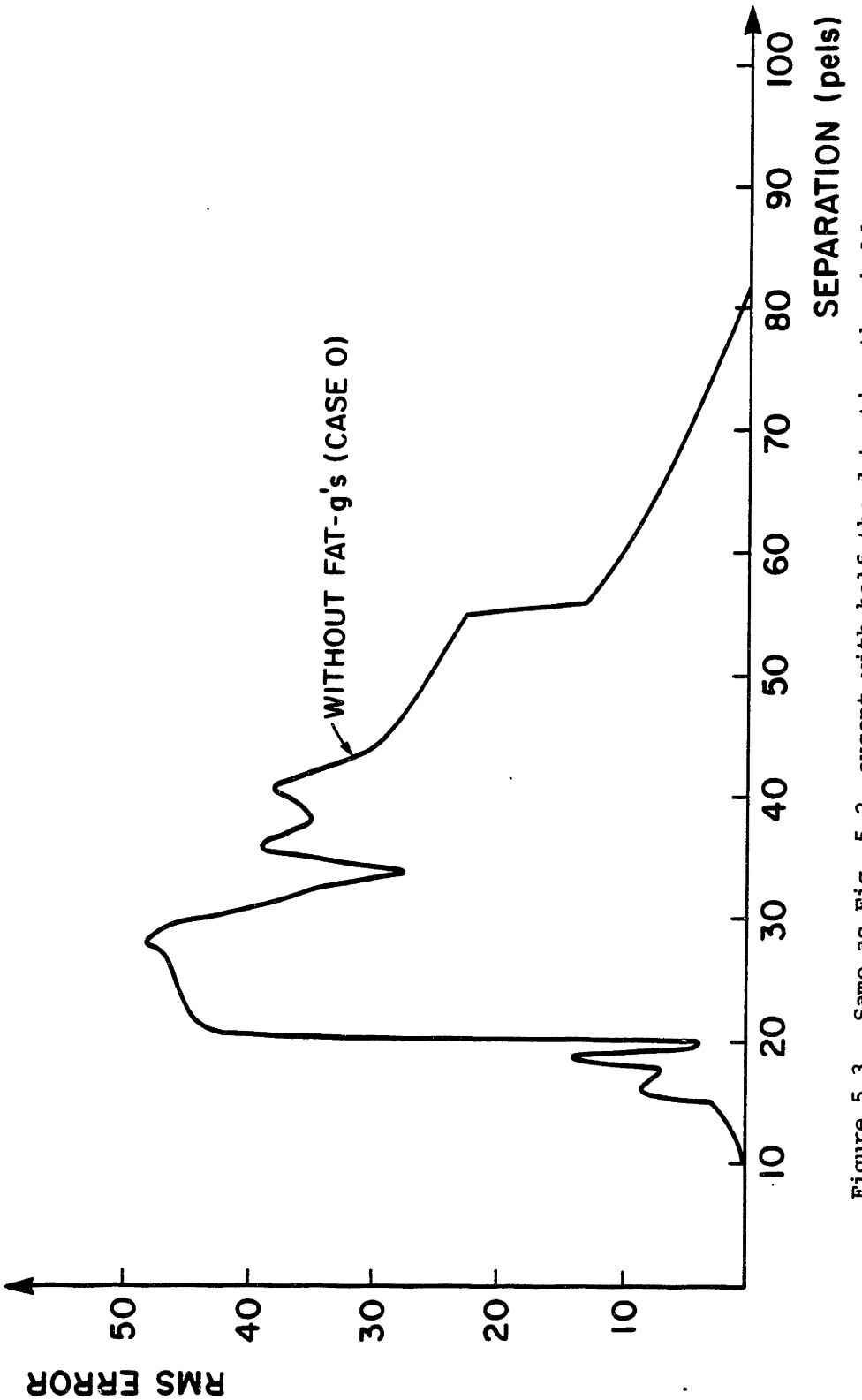


Figure 5.3 Same as Fig. 5.2, except with half the detection threshold: 7 vs. 14.

above higher thresholds.

In addition to the lower threshold retaining a few jumps in the error curves, it may introduce a new behavior in the error curves as compared with the higher threshold case. This new behavior may exacerbate the reconstruction error as can be seen by comparing fig. 5.2 and 5.3 in the separation range of 10-17 pels. The reason for this behavior in fig. 5.3 is the retaining of more negative sidelobes when the threshold is lower, even after the first and second order corrections performed by the reconstruction algorithm.

The above demonstration showed that lowering the threshold does not completely eliminate the effect of missing the detection of some morphemes when their corresponding filter peaks fall below the detection threshold. Furthermore, lowering the threshold may open a "whole new can of worms" by introducing a deteriorated error behavior, as seen in fig. 5.3 between 10-17 pels. A more direct way of isolating such lack of detectability from the rest of the causes of reconstruction errors may be to process the test patterns with a priori knowledge of the existence of the morphemes. This can be accomplished by considering the position of the morphemes as given (henceforth referred to as case 1), and processing the test patterns to solve only for the width and amplitude of each morpheme. Note that in this case no peak detection is performed on the filter outputs; instead, the value in the filter output at the given morpheme position is used for the

estimation regardless of whether there is a peak at that position. For completeness, an additional processing run on the test patterns was made where the width, as well as the position, was given (henceforth referred to as case 2). In this case a further modificaton of the algorithm results : since the width of the gaussian event is a priori given, we can exclusively use the h-filter that is optimum for that width, to solve only for the event's amplitude. The h4 filter happens to be the optimum for the width of 17.8 pels used in these test patterns.

In principle, the above three types of processing, (i.e. once with no given parameters, henceforth referred to as case 0; a second time with only the position given; and a third time with both the position and the width given), would help isolate the individual contributions of the inaccuracies in estimating each of the three morpheme parameters to the overall reconstruction error. As the results below show, however, this goal was not achieved using the above three types of processing.

Fig. 5.4 shows the error curves for cases 1 and 2 above, computed (without the use of fat-g's in the reconstruction) for the two-event test pattern of fig. 5.1 and superimposed on the curves of that figure. As expected, the error is less for all separations when both the width and the position are given, than when just the position is given. However, the following main features of the case 1 and case 2 curves need to be further explained.

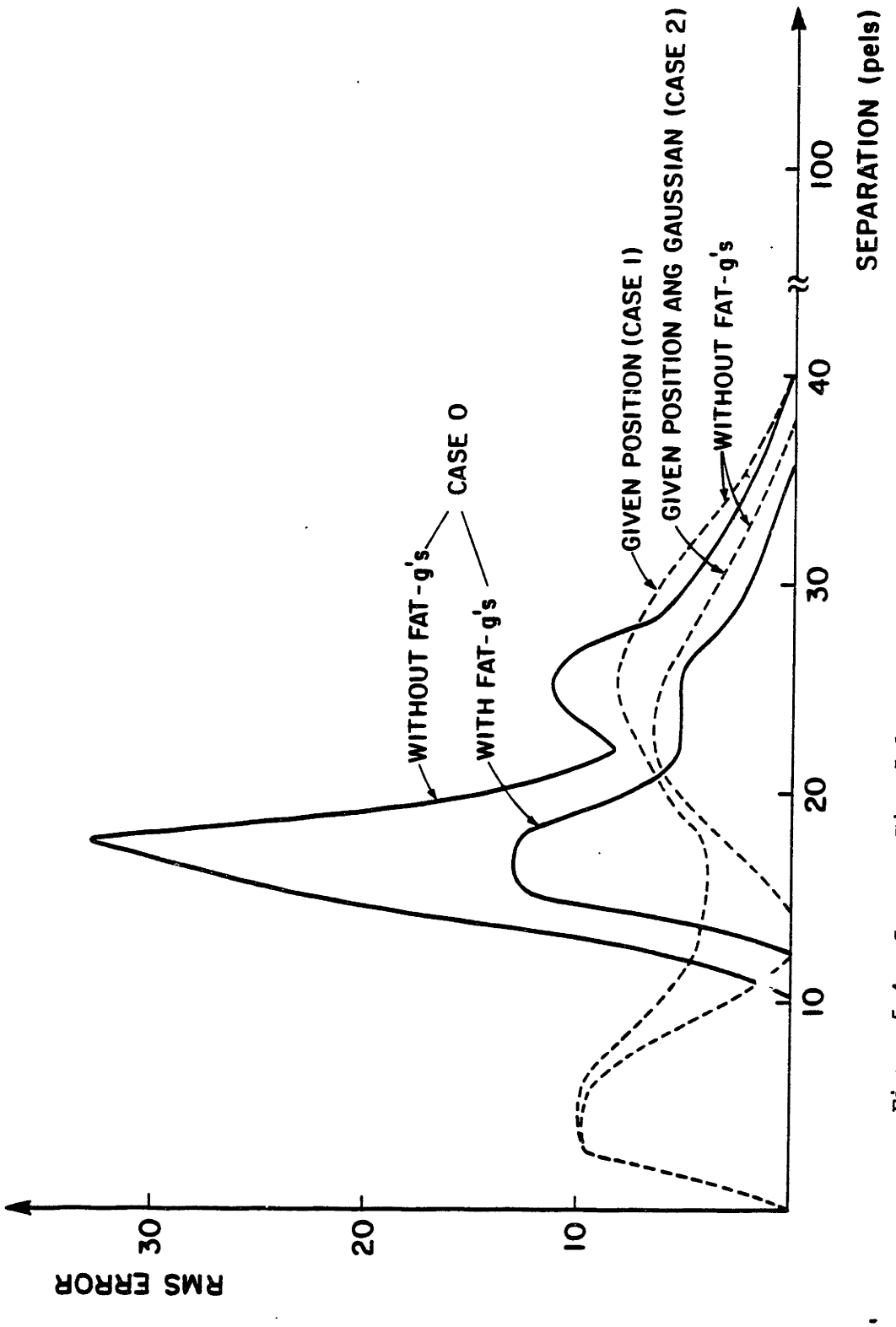


Figure 5.4 Same as Fig. 5.1, except cases 1 and 2 are also shown.

First, the curve for case 1 rises above that of case 0 in the range 27-40 pels of separation. This is because the error in the (non-given) position in case 0 tended to partially cancel the error in the other parameters. This is similar to what is observed for the range of 21-25 pels in fig. 5.1, as explained above. Since the position is given in case 1, it is free of any error to partially cancel the error in the other two parameters; hence the increase in the overall error in case 1 over case 0 in the 27-40 pel range. Second, for a reason similar to what caused the error reduction in fig. 5.2 in the 32-36 separation range, we observe a dip in the case 1 and case 2 error curves in the 10-17 separation range. Third, the huge increase in the error of both cases for separations less than 10 pels, where the case 0 error decayed rapidly to zero, is due to forcing the reconstruction to use two events (when a position is given for each) at a time when the two events have merged into one. The two curves, however, do finally decay to zero when one of the events corrects the other out as they get sufficiently close to one another.

The first and third features of the two curves above eliminate their value in trying to isolate the individual reconstruction-error contributions of the inaccuracies in estimating each of the three morpheme parameters; such a method of isolation requires that the individual contributions be additive.

Fig. 5.5 shows case 1 and case 2 curves computed

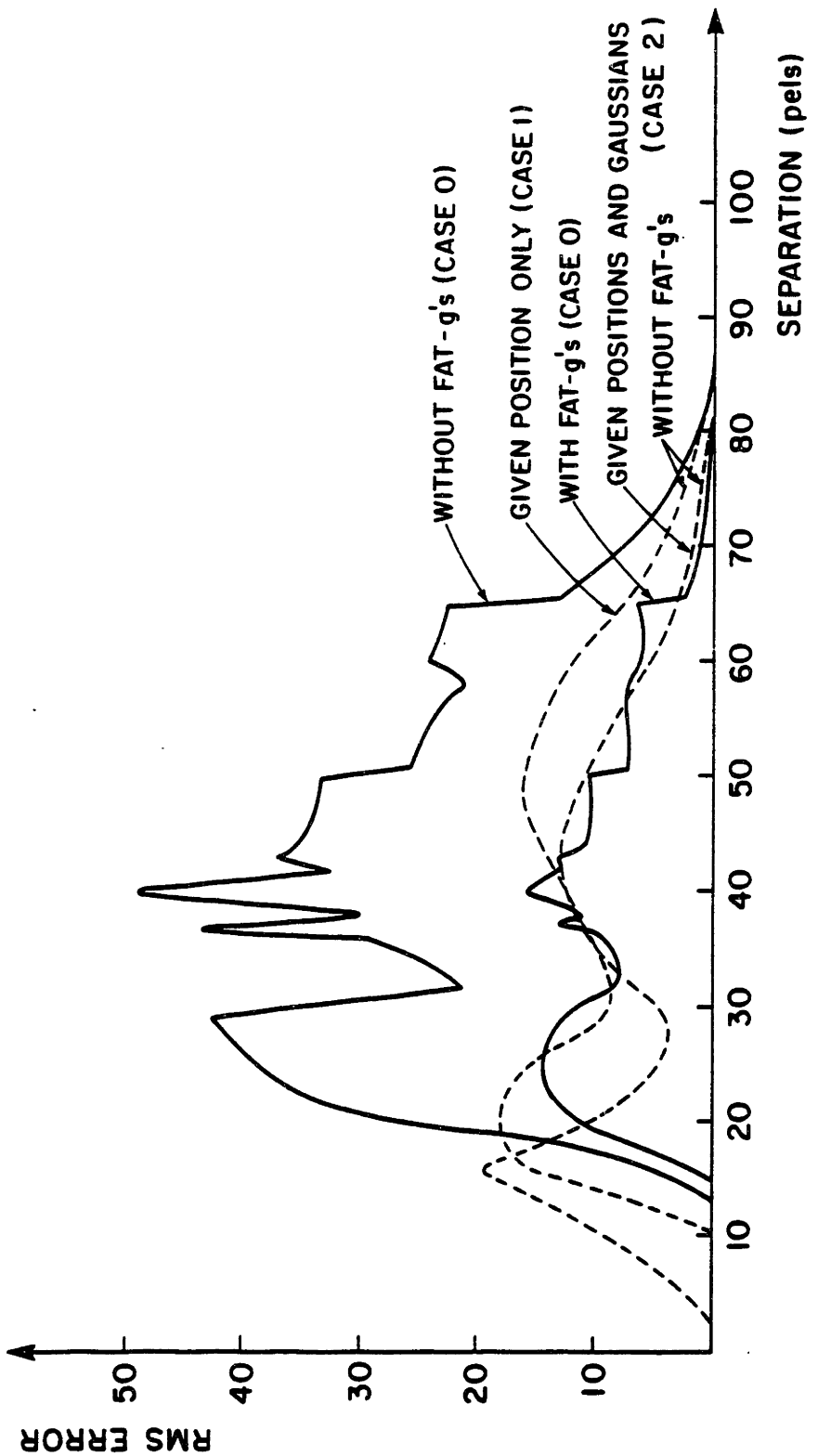


Figure 5.5 Same as Fig. 5.2, except cases 1 and 2 are also shown.

(again, without the use of the fat-g's in the reconstruction) for the three-event test pattern of fig. 5.2. The two curves are similar to those of fig. 5.4 except for the rise of the case 2 curve above that of case 1 for separations less than 16 pels. The reason is the partial compensation of the inaccuracies in width and amplitude estimations for each other in case 1, as explained above.

5.1.2 Reconstruction error as a function of amplitude

Similar to the approach in the previous subsection, the dependence of reconstruction errors on event amplitude is studied here by fixing the event separation and widths while varying the amplitude and interpreting the resulting reconstruction errors. Again, we start with a test pattern consisting of only two gaussians. Fig. 5.6 shows the reconstruction error, with and without the use of fat-g's, as a function of the amplitude of one of two events having a fixed half-amplitude width of 17.8 pels and a fixed separation of 21 pels between them. The amplitude of one of the events was held constant at 100 while that of the other event was varied from 0 to 255. The curve for the case of having the positions of the events known a priori (i.e. case 1) is also shown in fig. 5.6. The case of having both the position and the width known a priori is not shown since, within the roundoff error, it is identical to the case 1 curve for all amplitudes. This is consistent with fig. 5.4 which shows case 1 and case 2 error curves to be almost touching each other at the 21 pel

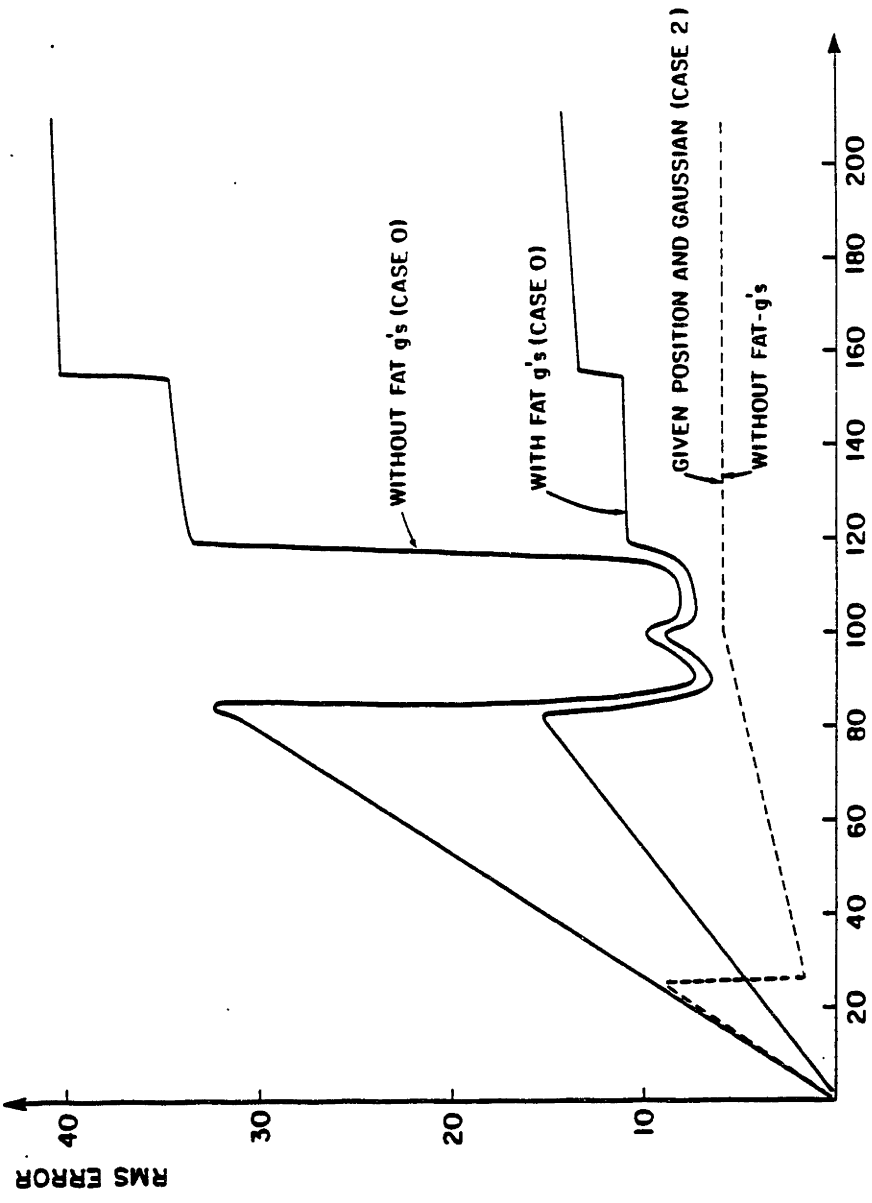


Figure 5.6 Reconstruction error as a function of the amplitude of one of two events (gaussians) having constant width (17.8 pels) and separation (21 pels).

separation, (i.e. the separation for fig. 5.6). All error curves in fig. 5.6 were computed at amplitude increments of 2.

Starting at the origin of the case 0 error curve, (i.e. no parameter is a priori known), computed without the fat-g's we observe a continuous linear rise up to an amplitude of 84. This happens as the filter peaks corresponding to the variable-amplitude event stay buried under the skirts of the the fixed-amplitude event until the former is large enough to stand out in the detection process. Note that this problem is not nearly as severe in the case 1 error curve; it rises only to an amplitude of 25. This is because, as explained above, case 1 does not use the usual filter-peak detection but rather uses the given position to read the corresponding filter output for each morpheme. The fact that the error curve for case 1 shows any rise at all at the beginning is due to the smaller event being corrected out, i.e. eliminated, by the larger fixed-amplitude event until the former is large enough to remain above the detection threshold after such correction.

Continuing with the case 0 curve, we observe a small peak shortly after the sharp drop in the error. This peak is caused by a slight deterioration in the position estimation when the two events become equal in amplitude. The next main feature is the sharp rise at amplitude 115 being caused by the peak of the wider (h4) of the two h-filters used to estimate the fixed-amplitude event getting buried under the skirts of the variable-amplitude event, and therefore being undetected. When the other peak (i.e. that of h3) gets similarly buried,

another jump in the error curve takes place at amplitude 155. It is reasonable to expect the peak of the wider filter to suffer from such burial before the peak of the narrower filter due to the obvious fact that such skirt-effects occur sooner in the wider filter as the amplitude increases. As evident in fig. 5.6, the case 1 error curve does not show such jumps, since this case does not suffer from such lack of detectability.

While the curve for the case of using the fat-g's in the reconstruction lies below that for the case of not using them for all amplitudes, it indicates that the fat-g's are even more valuable where one of the events is undetected. This can be seen by comparing the two solid curves in fig. 5.6 in the amplitude ranges 0-84 and 115-255.

The initial rise of the case 1 curve of fig. 5.6 was explained above. The more gradual rise that follows and the final constant level of the curve, however, need to be elaborated further. The difference between the gradual rise occurring in the amplitude range of 25 to 100 and the constant level occurring at amplitudes greater than 100 is the change of order of event estimation as the variable-amplitude event rises above the fixed-amplitude event. Since, as shown at the end of this subsection, the reconstruction error in a two-event case is a function only of the smaller event, there is a continuous change of this error in the range 25 to 100, where the variable-amplitude event is the smaller of the two. Above the 100 amplitude, the smaller event is that with the fixed-

amplitude, leading to no change in the reconstruction error.

Fig. 5.7 shows the corresponding three-gaussian case. In this case the amplitudes of the two outer events were held constant at 100, while the amplitude of the middle event varied from 0 to 255. Again, we note that the closeness of the case 1 and case 2 curves is consistent with fig. 5.5 where it is shown that the two curves are quite close at the 42 pel separation (which is the separation for fig. 5.7). The curves in fig. 5.7 are similar to those in fig. 5.6, except for two differences.

First, the dip in the case 0 curves at an amplitude of 40. This happens when the negative sidelobes, (of the outer events), at the position of the middle event are corrected to yield a positive peak which is large enough (i.e. above the detection threshold) to be retained. Below 40, the negative sidelobes are too large to yield an appreciable (i.e. above the detection threshold) positive peak after correction. Above 40, the sidelobes themselves are not large enough to be detected to begin with, leaving nothing at the position of the middle event to account for it. Second, the error curves for cases 1 and 2 have two middle slopes, instead of one as in fig. 5.6, in the range 77-125. The reason is that the three events undergo two changes in the order of event estimation rather than just one as in the case of the two-event test pattern explained above.

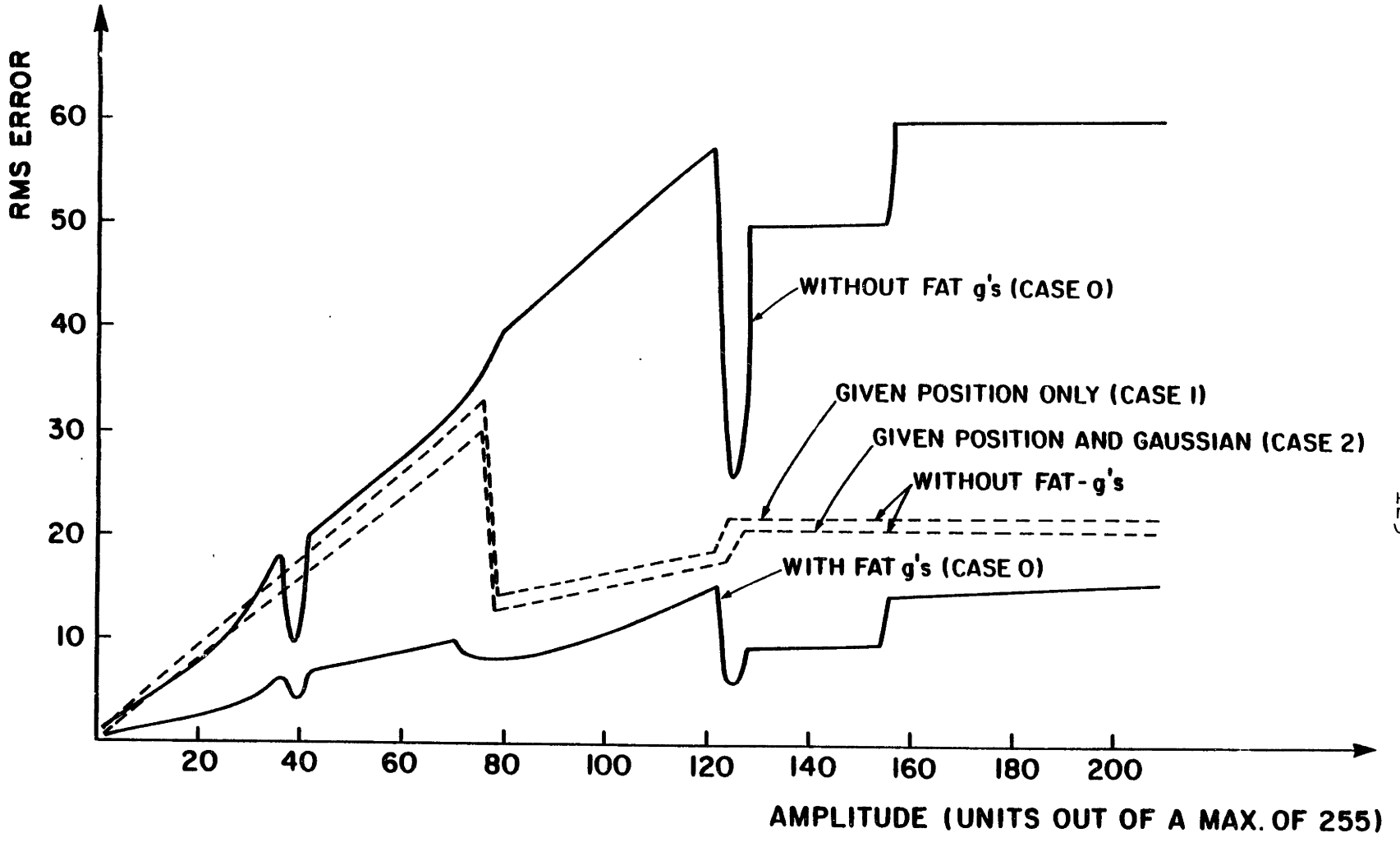


Figure 5.7 Reconstruction error as a function of the amplitude of the middle of three events (gaussians) having constant width (17.8 pels) and separation (21 pels).

This subsection is concluded by showing that for any two events, the reconstruction error is a function of only the one with the smaller amplitude.

Let the filter peak response to the larger of two events in the absence of the smaller event be P_l°. Let the peak in the presence of the smaller event be $P_l(P_s)$, because it's a function of the smaller event. Let P_s° and $P_s(P_l)$ be the corresponding peak responses to the smaller event. Let P_s° and $P_s(P_l)$ be the corresponding peak responses to the smaller event.

We have:

$$P_s(P_l) = P_s^\circ + P_l^\circ \bar{h}(r)$$

where $\bar{h}(r)$ is the normalized h-filter response at r, (the distance between the two events).

Similarly:

$$P_l(P_s) = P_l^\circ + P_s^\circ \bar{h}(r)$$

Note that since both events, in this case, have the same widths, $\bar{h}(r)$ is the same for both of them.

When $P_s(P_l)$ is corrected to become P_s^c we have:

$$P_s^c = P_s(P_l) - P_l(P_s) \cdot \bar{h}(r)$$

$$= [P_s^\circ + P_l^\circ \bar{h}(r)] = [P_l^\circ + P_s^\circ \bar{h}(r)] \cdot \bar{h}(r)$$

$$= P_s^\circ (1-[\bar{h}(r)]^2). \qquad (5.1)$$

$P_1(P_s)$ will then be corrected using P_s^c,

$$P_1^c = P_1(P_s) - P_s^c \cdot \bar{h}\ (r)$$

$$= [P_1^\circ + P_s^\circ \bar{h}\ (r)] - [P_s^\circ(1-[\bar{h}(r)]^2)] \cdot \bar{h}(r)$$

$$= P_1^\circ + P_s^\circ[\bar{h}(r)]^3 \qquad (5.2)$$

Using equations 5.1 and 5.2 above to determine the errors in estimating the events we get:

$$\text{error in } P_s = P_s^\circ - P_s^\circ(1-[\bar{h}(r)]^2) = P_s^\circ[\bar{h}(r)]^2$$

$$\text{error in } P_1 = P_1^\circ - P_1^\circ - P_s^\circ[\bar{h}(r)]^3 = -P_s^\circ[\bar{h}(r)]^3$$

Therefore both errors are functions of only the smaller event.

5.1.3 Reconstruction error as a function of width

In this subsection, the dependence of the reconstruction error on the parameter in question (in this case the width) is again shown by holding the other two morpheme parameters constant while varying the width, and interpreting the resulting variation in the reconstruction error. Fig. 5.8 shows the two-event case where the amplitude and the separation of two gaussians are held constant at 100 and 21 respectively, while their widths are simultaneously varied from 1 to 43 pels. The curves were computed at points corresponding to the widths of the first 13 of the 21 g-filters, (see chapter 1).

The main features of the case 0 curve without the fat-g's is the sharp rise in the error at a separation of 21 pels, which is the zero-crossings width of the widest h-filter (h4), and the continuous linear increase in the error after that. This is intuitively reasonable, since no h-filter is expected to be able to retrieve wider events separated by a distance less than or equal to the width of the h-filter. The necessity of having the seven "fat" g's to retrieve the low frequency structure in the image is clearly demonstrated by comparing the two case 0 curves, and by comparing the case 0 curve with the fat-g's with the case 1 curve without them.

To provide a broader-scoped indication of the dependence of the reconstruction error on event width for the two-event case, fig. 5.9 shows contours of rms reconstruction

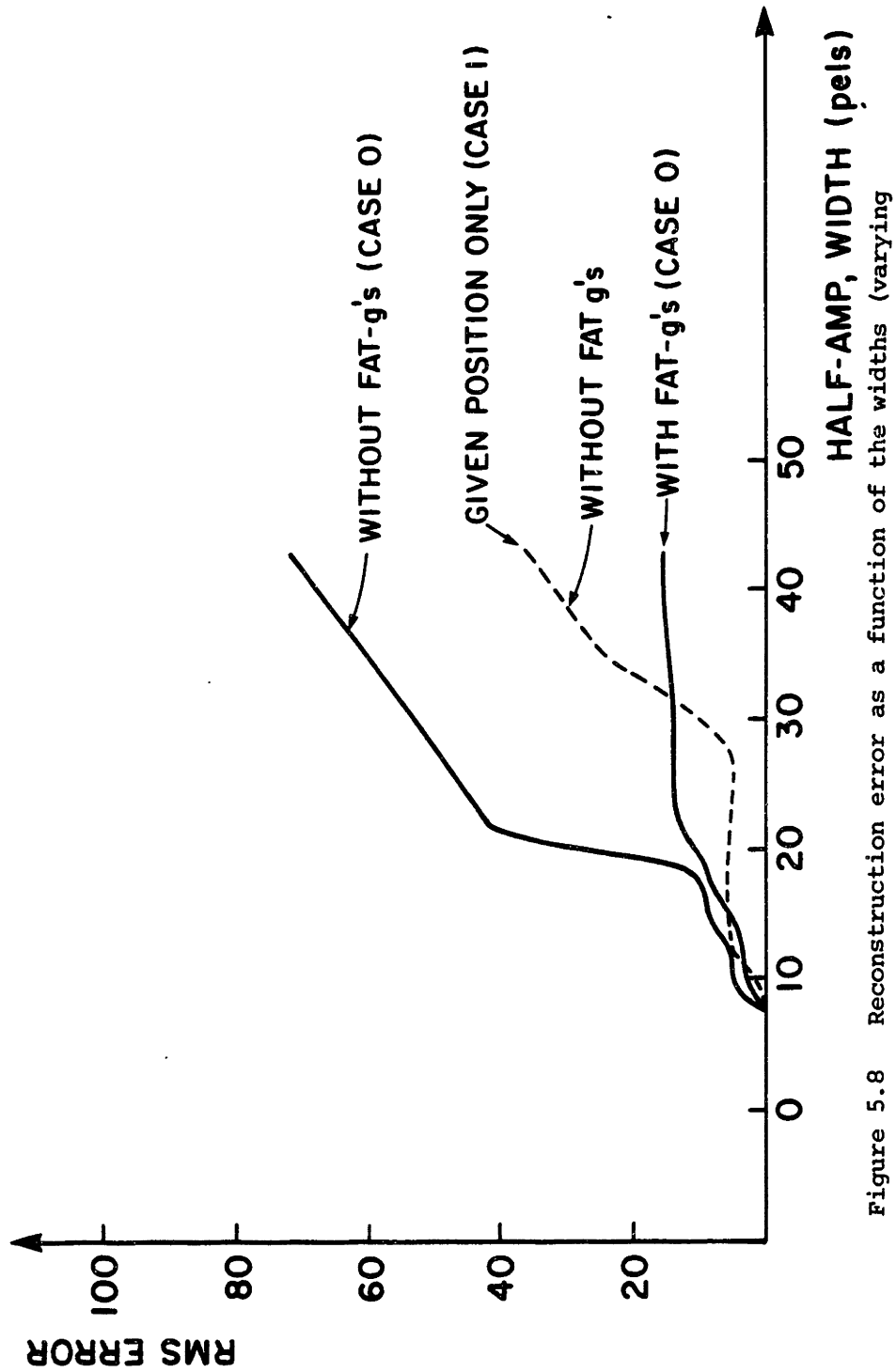


Figure 5.8 Reconstruction error as a function of the widths (varying in trnadum) of two events (gaussians) having constant separation (21 pels) and amplitude (100).

error for the same test pattern of fig. 5.8 (without the use of fat-g's in the reconstruction), except here the widths of the events are more broadly varied as indicated by the axes. Fig. 5.9 shows two regions of rms contours separated by the boundry at § the width of the widest h-filter, h4, with the higher errors on the outside, i.e. towards the wider events. The reason for such regional division is identical to that for the sharp rise in the case 0 error curve in fig. 5.8, as explained above. The corresponding contour plot for the case of using the fat-g's in the reconstruction is shown in fig. 5.10. Note that the case 0 curves in fig. 5.8 follow the error behavior along the diagonals of figs. 5.9 and 5.10.

Fig. 5.11 shows the three-gaussian case. The amplitudes in this figure were held constant at 80 and the positions of the events at 43, 64, and 85 pels, while the widths were varied simultaneously from 1 to 43 pels. Again, the curves were computed at points corresponding to the first 18 of the 21 gaussians mentioned above. The peak in the case 1 error curve is due to the error dropping as the one-event reconstruction finally takes over to provide a more accurate estimate of the original line, as explained for figure 5.4 above. The A-B-C inversion in the case 0 curve without using the fat-g's is best explained using fig. 5.12 as follows.

Figs. 5.12a and 5.12b show the original line and the corresponding h4 filter response for peak A in fig. 5.11. In these two figures, it is clear that since the original line

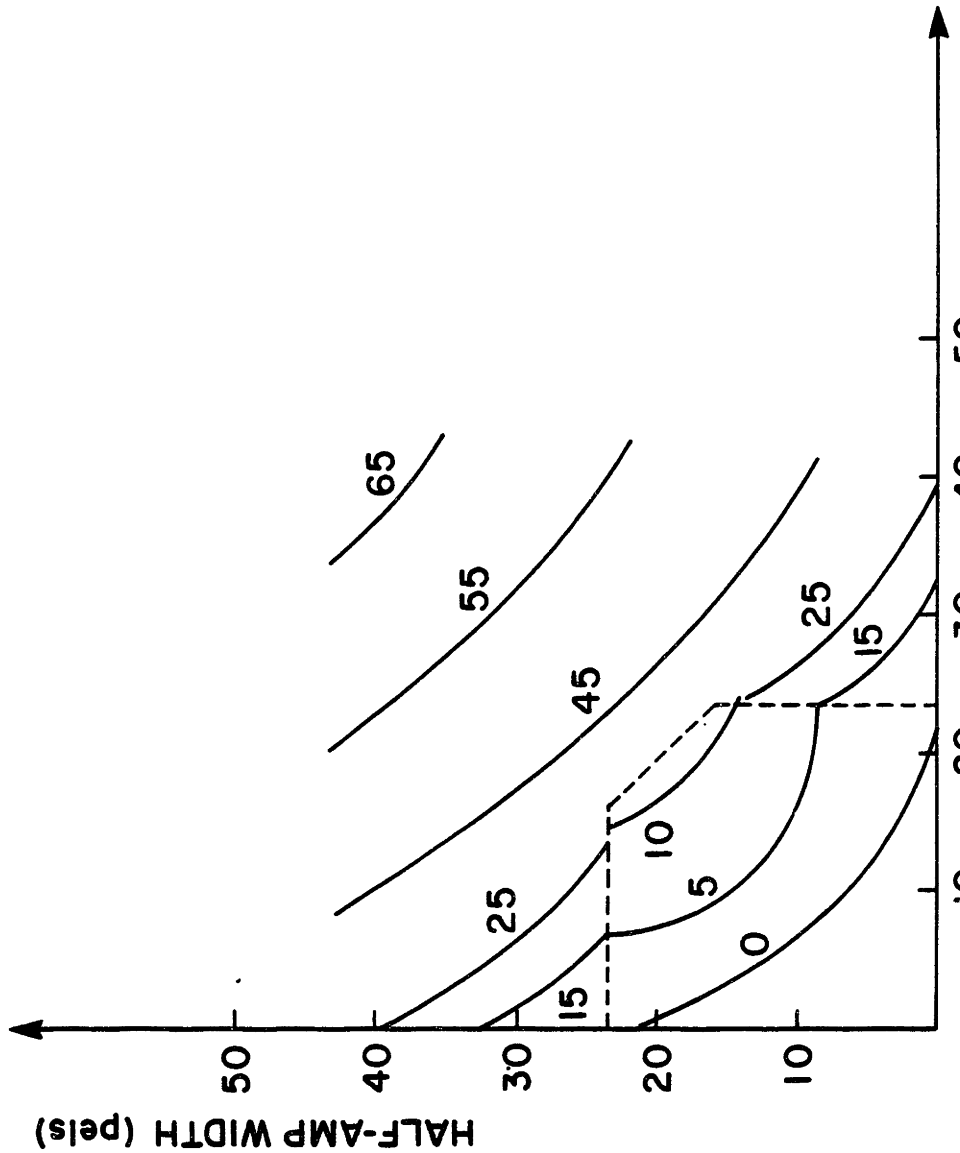


Figure 5.9 Reconstruction error as a function of the widths of two events (gaussians) having constant separation (21 pels) and amplitude (100).

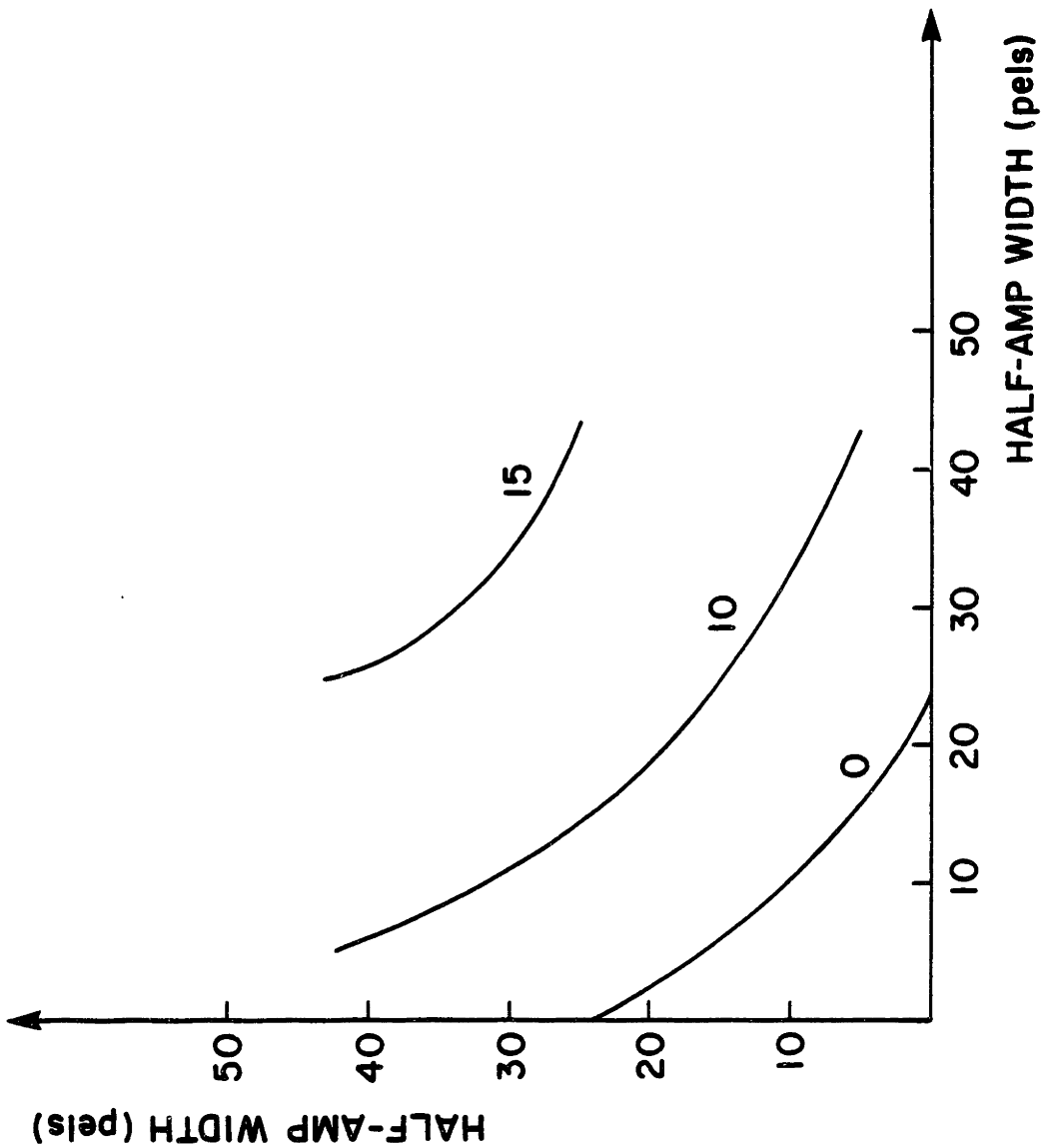


Figure 5.10 Same as Figure 5.9, except the fat-g's are used.

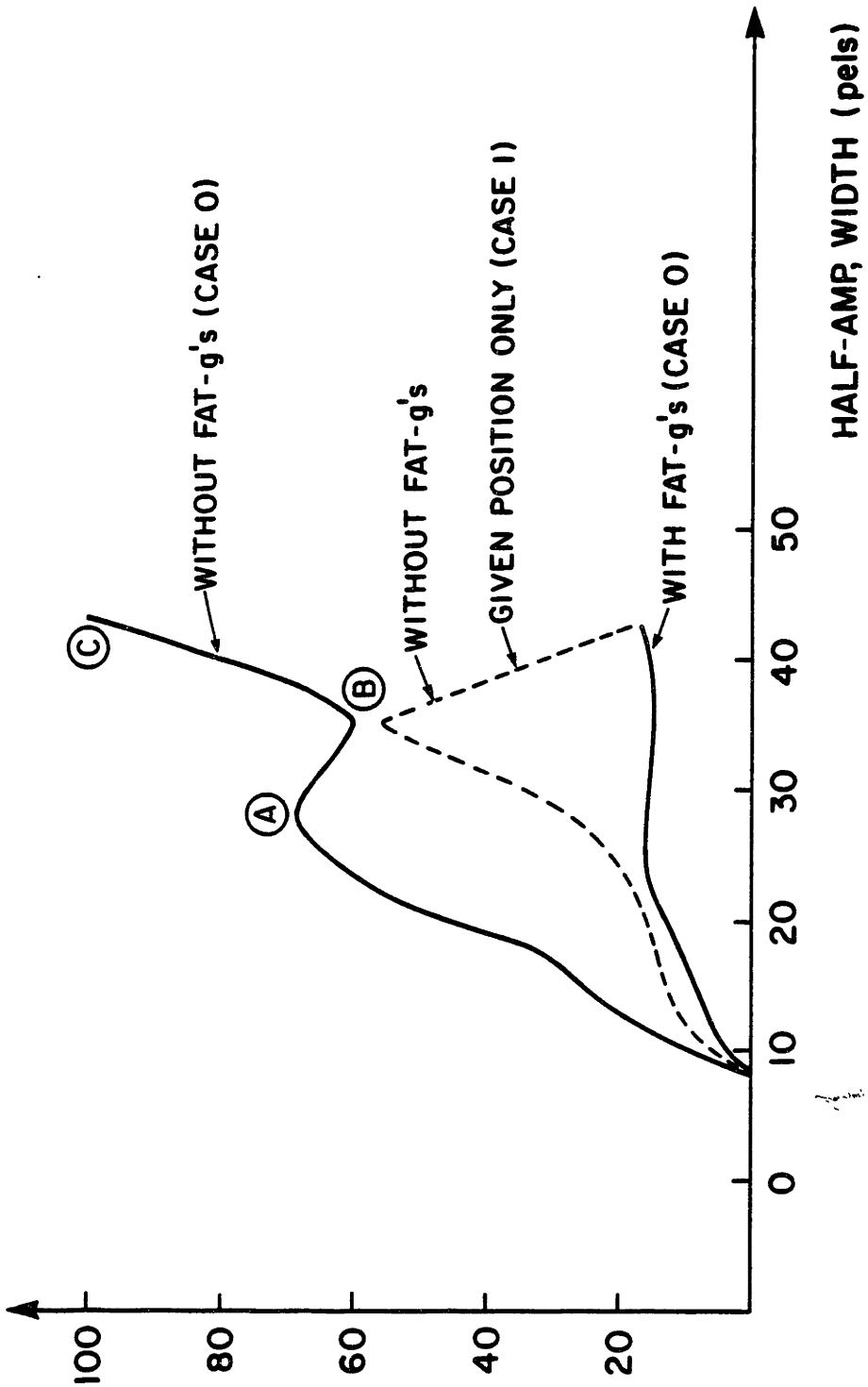
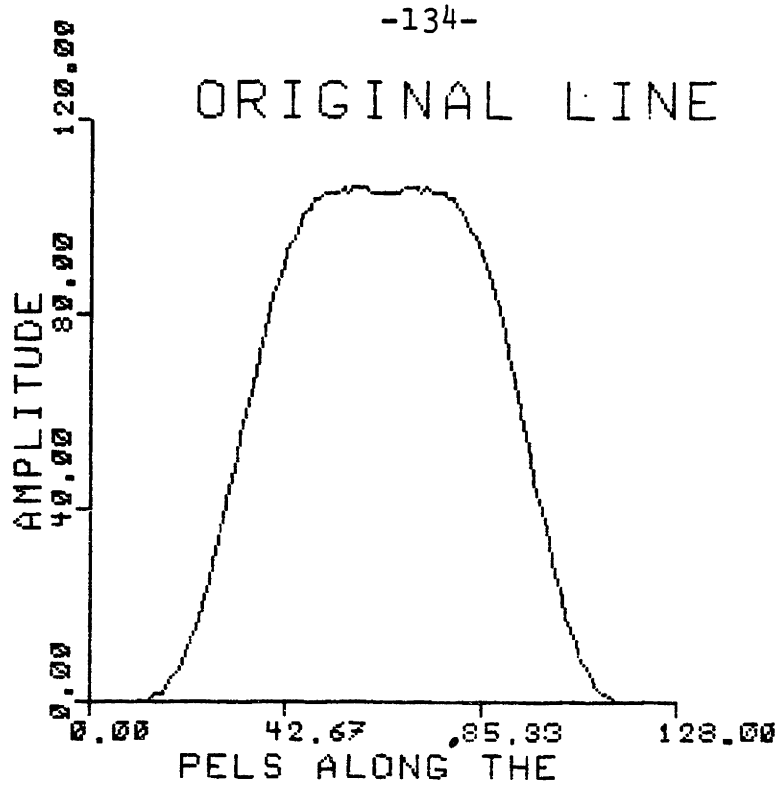


Figure 5.11 Reconstruction error as a function of the widths (varying in tandum) of three events having constant separation (21, 22 pels) and amplitude (100).



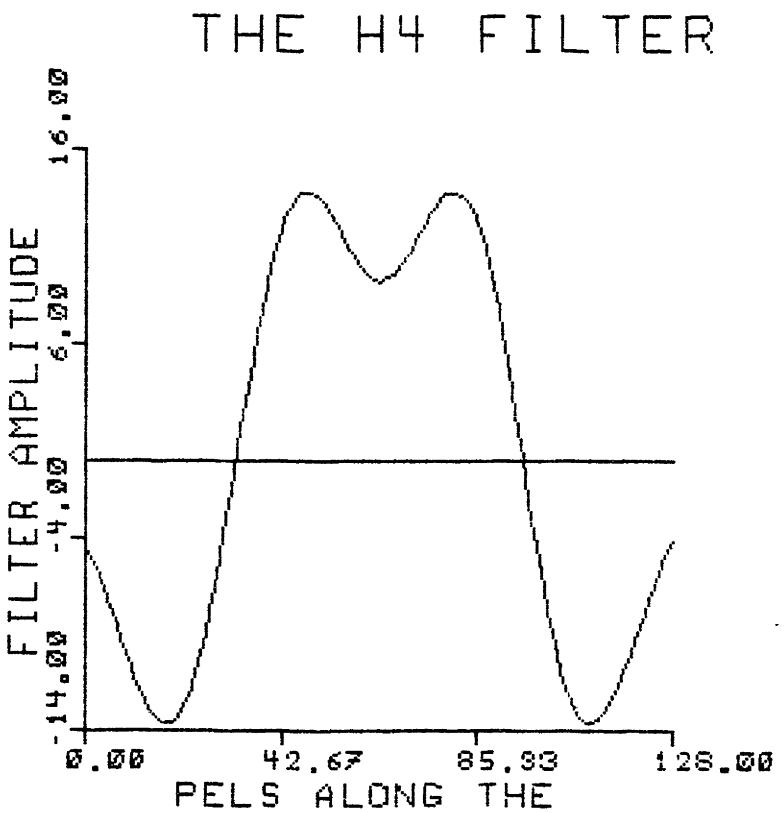
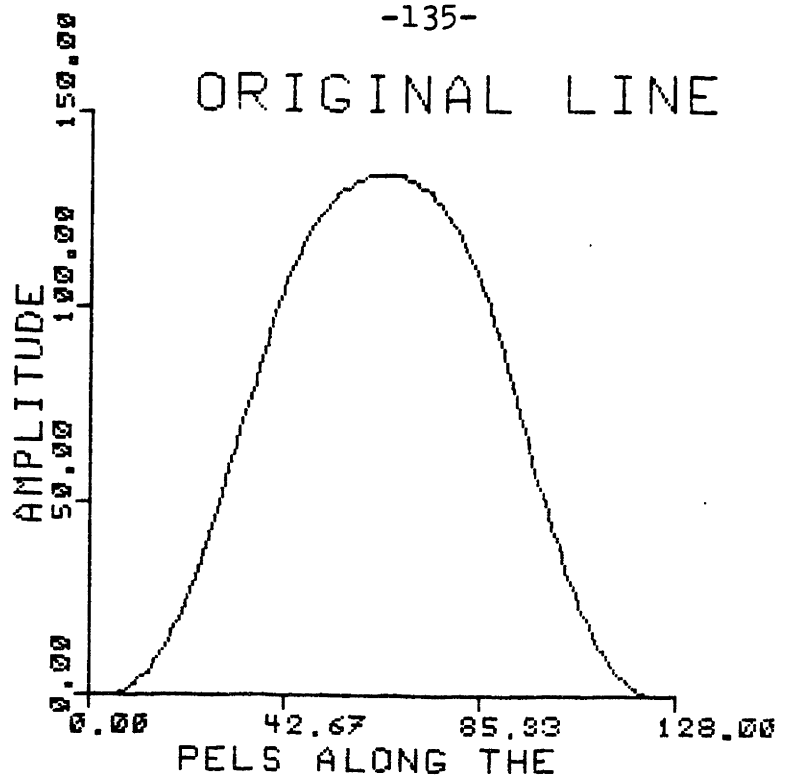


Figure 5.12 (a,b) Point A of the curve in Fig. 5.11. The h4-filter peaks are below the detection threshold of 14.



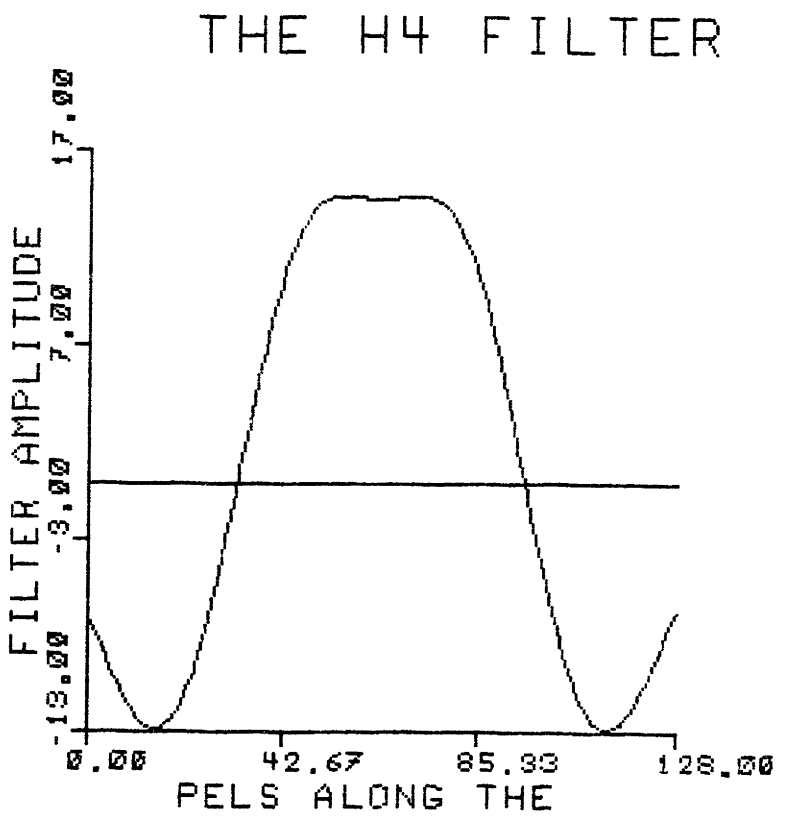


Figure 5.12 (c,d) Point B of the curve in Fig. 5.11. The h4-filter peaks are above the detection threshold of 14.

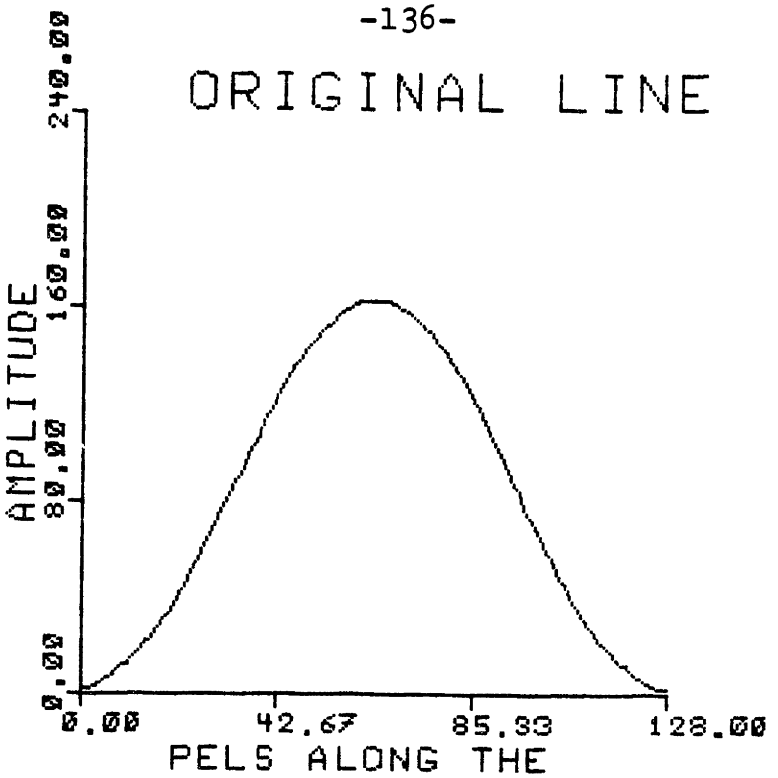


Figure 5.12 (e) Point C of the curve in Fig. 5.11. The resulting shape is much too wide at the half-amplitude points for the h4-filter to yield any accurate estimation.

has a very broad peak, no appreciable (i.e. above detection threshold) peak resulted in the h4 filter output or, for that matter, any other filter output. As the width changed to where point B is, we see from figs. 5.12c and 5.12d that the line looks much more like a bell-shaped curve, which is the shape the h-filters were designed to detect; hence the appreciable peak filter-response of h4 and the consequent reduction in error at point B. At point C, however, even though the line looks even more like a bell-shaped curve, as shown in fig. 5.12e, it is much too wide at the half-amplitude points for the h4 filter to yield any accurate estimation; hence the rise in the error curve in fig. 5.11. Case 1, on the other hand, achieved a drop in the error at this point, because it used h3, (regardless of whether it responded with an appreciable peak to such a wide event), in addition to h4, thereby, in this case, yielding a more accurate event estimation.

Note that since the fat-g's are always used when coding an image, explanations of some of the errors above, such as those in the previous paragraph, are only necessary to understand the ramifications of the operations involved in the coding algorithm. Furthermore, they help demonstrate the need for the seven fat-g's in the reconstruction process.

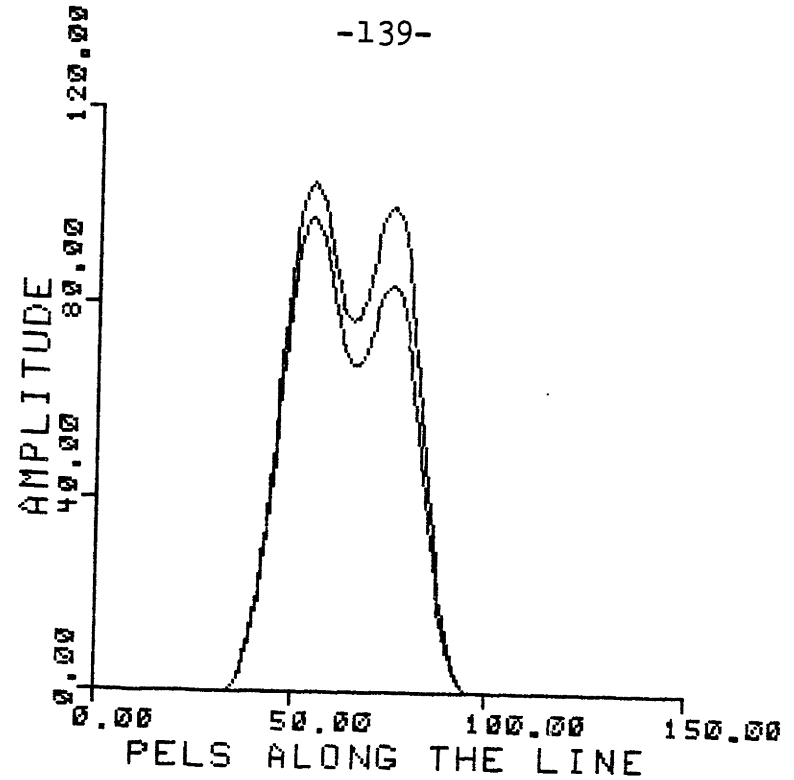
5.2 Two-dimensional noise

Two-dimensional noise is defined here as the imperfections in the reconstructed image which appear after the final stage of coding the image as a two-dimensional structure, i.e. after concatenating the one-dimensional morphemes into strokes. Some of the types of imperfections under this category have their origin in this final, i.e. the stroking, stage; others originate in the preceding stage of reconstructing each image line separately with one-dimensional morphemes. However, all types can be traced to the fact that one-dimensional, rather than two-dimensional, morphemes are used here as the basic building blocks for reconstructing a two-dimensional signal. Being restricted to one dimension, such morphemes, even after being concatenated into strokes, sometimes fail to accurately capture the signal behavior in the other dimension.

Four types of two-dimensional noise have been identified. Each type is explained and demonstrated below to the extent it was possible to isolate it from other noise-types of this category and of the category of section 5.1.

5.2.1 Change of order of event estimation from line to line

The type of imperfection resulting from change of order of event estimation is demonstrated in figs. 5.13a thru 5.13d. Figs. 5.13a and 5.13b show original lines, each with



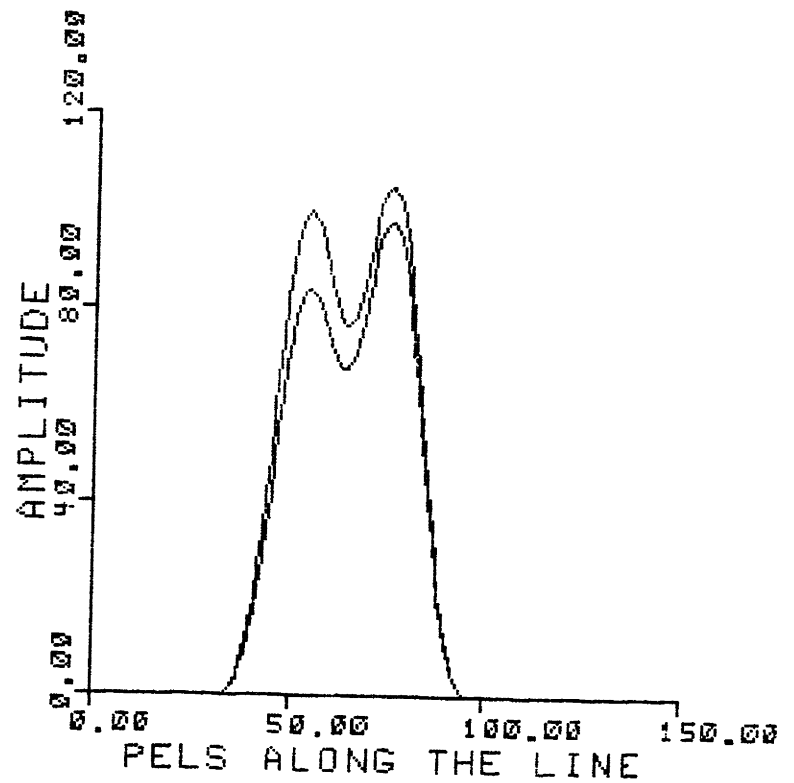
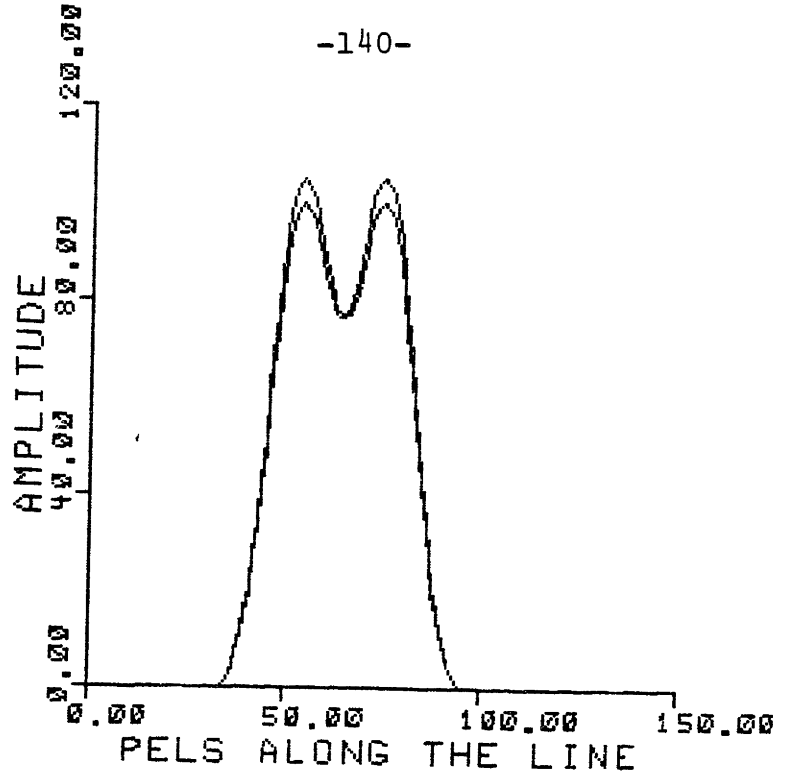


Figure 5.13 (a,b) Two original lines, each with the corresponding reconstructed line. Originals have larger amplitudes.



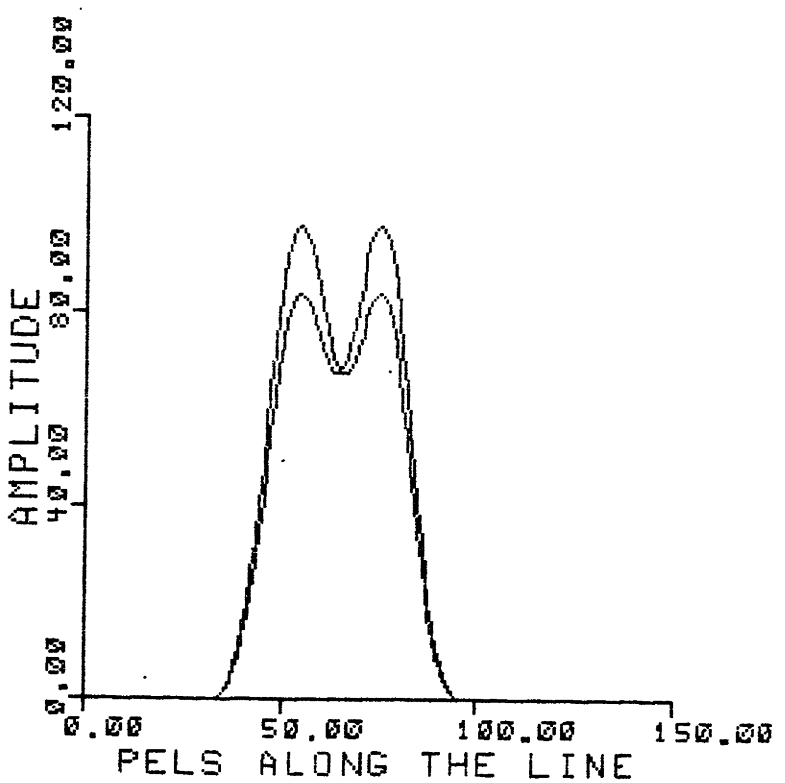


Figure 5.13 Effect of change of order of event estimation. (c) The two original lines. (d) The two reconstructed lines.

its reconstructed line, where two events have slightly changed rank in amplitude from one line to the other. This is further demonstrated in fig. 5.13c which shows the two original lines. The slight difference between the two lines of the original image is to be contrasted with the amplified difference as it appears between the corresponding lines of the reconstructed image, which are shown in fig. 5.13d.

The cause of such amplification of the difference between the lines is the change of order of event estimation as the smaller of the events in one line becomes the larger of the two in the other, thereby reversing the order of event estimation which always starts with the largest event. Since the simplified method of first and second order corrections used in the coding algorithm does not treat all events symmetrically, (since symmetry would require a much more computationally demanding method such as simultaneous or iterative solutions), such an amplified difference results. This type of noise contributes to one of the most objectionable imperfections in the reconstructed image; namely the horizontal streaks. Although smoothing in the stroking stage goes a long way to alleviate that effect, as shown in chapter 4, it still falls short of a perfect result.

5.2.2 Approximating arbitrary shaped strokes with a series of concatenated parabolas.

Figs. 5.14a thru 5.14c demonstrate the imperfection in reconstructed images resulting from this type of noise. Fig.

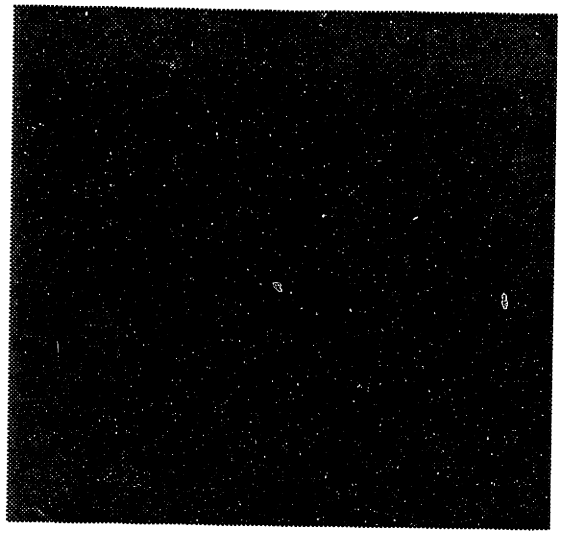


Figure 5.14 (a) Original image.

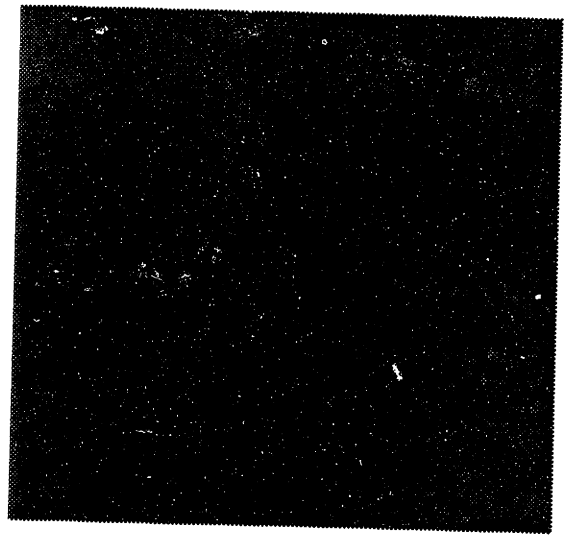


Figure 5.14 (b) One dimensionally reconstructed image.

Figure 5.14 (c) Two-dimensionally (stroked) reconstructed image.

Figure 5.14 Effect of approximating arbitrary shapes using concentrated parabolas.

5.14a shows the original image; 5.14b the one-dimensionally reconstructed image, (i.e. before stroking); and fig. 5.14c the final, i.e. stroked, reconstructed image. The image in fig. 5.14c exhibits only one imperfection compared to the original image in fig 5.14a; namely the discontinuity in the smooth flow of the shape right after the inversion in the S-shaped feature. This aberration in the smoothness occurs right where the two positional, (see chapter 4), parabolas that approximate the S-shaped curve meet.

To prevent such discontinuities, it would be necessary to break the shape into two parabolas at another point which would produce a less objectionable or no discontinuity, and therefore a more nearly accurate, if not a perfect, reconstruction of the image. Such an objective requires an iterative or a simultaneous method, entailing a prohibitive increase in the computational cost. This contradicted the original goal of maintaining low computational cost at the various stages of the reconstruction algorithm.

The requirement of low computational cost dictated the algorithm's method of sequentially forming these parabolas by concatenating one line event at a time and testing to see if the resulting parabola still meets a given error criterion as compared to the unstroked image. It is not accurate, however, to think that discontinuities such as the one discussed above can be eliminated by tightening that error criterion, since it is lower bounded by the amount of error flexibility required to achieve the smoothness evident by comparing figs. 5.14b and

5.14c.

5.2.3 Change of signal representation.

This type of noise leads to an imperfection in the reconstructed image when two, very slightly different, contiguous one-dimensional signals, i.e. image lines, are reconstructed with two very different sets of morphemes. As in subsection 5.2.1 above, when each reconstructed line is considered separately, its closeness to the corresponding original line may be quite satisfactory. However, it is when the two reconstructed lines are put side by side that discontinuities in image smoothness become apparent.

Such a case is demonstrated in figs. 5.15a and 5.15b. As the triangular shape narrows towards the bottom of the image, each of the two steps on either side of its vertical axis start being represented by a set of gaussian morphemes, rather than the more appropriate edge morpheme. This happens due to the lack of breadth in the black region inherently required for estimating an edge morpheme; (see chapter 3). This change of representation contributes to the streak seen in the next-to-last stage of the triangular figure. Note, however, that this is not the only cause of that streak; a lot of it is due to errors of the types discussed in section 5.1 above which occurred because of the strong interaction within the set of gaussians trying to reconstruct a step. As indicated in the beginning of section 5.2, it is therefore impossible to process that triangular pattern to yield the

Figure 5.15 Effect of change of signal representation. (a) Original image, (b) Reconstructed image.

effect of change of signal representation in isolation from the one-dimensional noise that is defined in section 5.1.

5.2.4 Discontinuities in the fat-g's

Since the fat-g's in each image line are intended to capture what is missed by the gaussian and edge morphemes, they are computed after the estimation of those morphemes. Since the morpheme set for each image line may vary drastically from line to line, so may the reconstruction of the fat-g's. Such variation in the reconstruction of the fat-g's manifests itself as wide horizontal streaks in the image.

The problem is particularly irritating because of the wide extent of each fat-g over the image line. An example of such a case is shown in fig. 6.4 of chapter 6. When the squares are reconstructed using strokes, horizontal streaks are noticed on both sides of each square. The reason is the change in the fat-g's where each square starts, or ends, as they compensate for the dip in the edge morphemes at the vertical axis of each square. Because of the above mentioned wide extent of each fat-g, such a change in the fat-g's spills over the left and right edges of the square. In addition, due to the strong interaction between adjacent fat-g's, such spill over is felt all the way to the end of the line.

5.3 CONCLUSION

This chapter showed the noise in the morphologically reconstructed images to be of a highly nonlinear and irregular nature. Such complexity prohibits tractable analytical formulation of the kind which might lead to more general conclusions than the ones presented by this chapter. For example, the original aim of processing two- and three-event test patterns was to explore the functional dependence of reconstruction errors on the number of events. However, the figures above show the change of behavior of the reconstruction errors to be sufficiently irregular to prevent a more systematic interpretation of such a functional relationship.

Even though the test patterns studied in this chapter consisted of gaussian morphemes only, it is expected that the reconstruction error-types for edge morphemes are largely identical to those covered above. Finally, it is hoped that the explanation of the noise behavior in the cases considered in this chapter will help provide sufficient insight into the intricacies of the operations involved in the morphological coding algorithm.

CHAPTER 6

COMPARING MORPHOLOGICAL CODING TO OTHER IMAGE CODING TECHNIQUES

Comparison between morphological coding and other coding techniques is most informing when it seeks to emphasize the exclusive features of morphological coding as opposed to most other methods of coding. Such comparison is of special interest, as shown below, when it is done with various types of transform coding. The following sections present major aspects in which morphological coding differs from other image coding techniques.

6.1 The use of non-orthogonal basis functions.

Almost all types of transform coding, (DCT, KL, Hadamard etc.), employ orthogonal basis functions for reconstructing images. The non-orthogonality of the morphemes used in morphological coding allows for more flexibilty in rendering images. This advantage is well demonstrated by considering image quality vs. its bandwidth, especially at low bandwidths, of images coded using both types of methods.

Figs. 6.1a thru 6.1i illustrate this advantage. The original image in fig. 6.1a is shown coded at various bandwidths, once using DCT, and again using morphological coding.

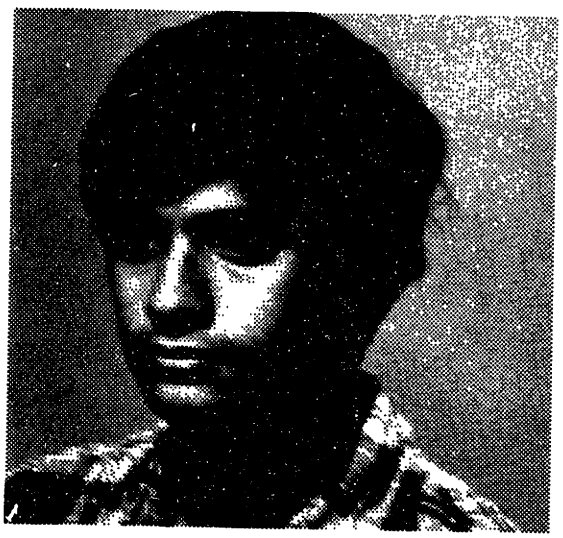


Figure 6.1 (a) Original image.

Figure 6.1 (b) Morphologically (One-dimensionally) reconstructed image with an average of 26 events per line; rms error = 12.5.

Figure 6.1 (c) DCT reconstructed image at the same bit rate as Fig. 6.1 (b); rms error = 7.1.

Figure 6.1 (d) Morphologically (One dimensionally) reconstructed image with an average of 10 events per line; rms error = 13.6.

Figure 6.1 (e) DCT reconstructed image at the same bit rate as Fig. 6.1 (d); rms error = 14.2.

Figure 6.1 (f) Morphologically (One-dimensionally) reconstructed image with an average of 5 events per line; rms error = 15.4.

Figure 6.1 (g) DCT reconstructed image at the same bit rate as Fig. 6.1 (f); rms error = 19.6.

Figure 6.1 (h) Morphologically (One-dimensionally) reconstructed image with an average of 3 events per line; rms error = 18.1.

Figure 6.1 (i) DCT reconstructed image at the same bit rate as Fig. 6.1(h) ; rms error = 23.9.

Figure 6.1 (j) the stroked image of Fig. 6.1(h). The bit rate is 0.25 bits/pel.

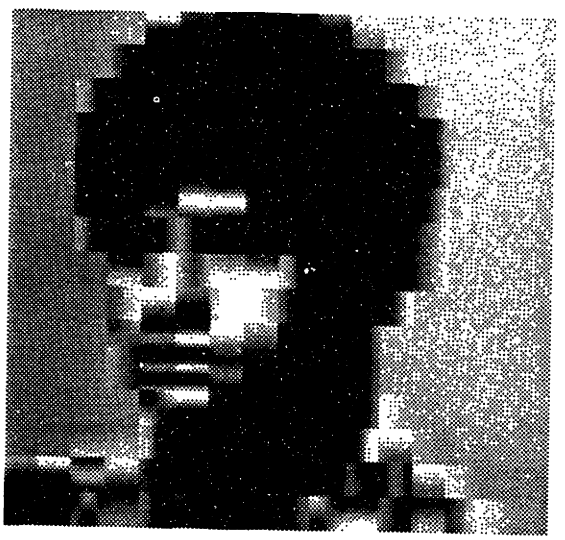


Figure 6.1 (k) the two-dimensional DCT reconstruction of the image at the same bit rate as Fig. 6.1(j).

For the sake of simplicity, both methods were restricted to one-dimensional coding, and the use of bit-distribution techniques was excluded for both. When computing the bandwidth, three parameters were counted for each morpheme, (position, amplitude, and width), while only two were counted for DCT coefficients, (position and amplitude). After the amplitudes of the seven fat-g's are also taken into account, the DCT coefficients per line, at a given bandwidth, outnumber the morphemes per line by a ratio of more than three to two. It's the number of the latter that is indicated in fig. 6.1.

Images coded at low bandwidths, (figs. 6.1f-6.1i), show a definite advantage for morphological coding; the morphologically coded images present a clearly more recognizable subject than those coded using DCT. At higher bandwidths, (figs. 6.1b-6.1e), although it is arguable that DCT shows a better performance because of the horizontal streaks in the morphologically coded image, the sharpness of the image is definitely better in the latter, even at the highest bandwidth shown, (figs. 6.1b and 6.1c). Figures 6.1j and 6.1k show the corresponding two-dimensional reconstruction of the images in figures 6.1h and 6.1i. Again, the morphologically coded image in fig. 6.1j presents a significantly more recognizable subject than the DCT coded image in fig. 6.1k.

Another way to compare the performances of the two types of coding is by considering the rms error of each at a given bandwidth. The rms error (in units out of a maximum of

255) is indicated for each coded picture in fig. 6.1. A plot of these errors is shown in fig. 6.2. The break-even point is at ten events per line; for a number of events less than ten, morphological coding is clearly the superior method. It's in this low bandwidth domain that morphological coding was intended to be used from the very beginning of its conception.

6.2 Processing is done in the spatial rather than in the frequency domain.

As the frequency content in an image increases, so does the bandwidth resulting from transform coding techniques. On the other hand, since morphological coding is feature (rather than frequency) oriented, increasing the frequency content of the image, for a given number of features, has no effect on the bandwidth of a morphologically coded image. This can be illustrated with the example shown in fig. 6.3. As the sharpness of the edge in figs. 6.3a-6.3c. increases from image to image, so does its frequency content. Morphological coding perfectly codes each of the three images with one edge stroke, plus the "ever-present" fat-g's. Hence, a fixed bandwidth results for all of the three edges. Figs. 6.3d-6.3f, on the other hand, show the results of the DCT coding of each of the three images with the bandwidth fixed at that of fig. 6.3d. The decrease in image quality is apparent as the sharpness of the edge increases.

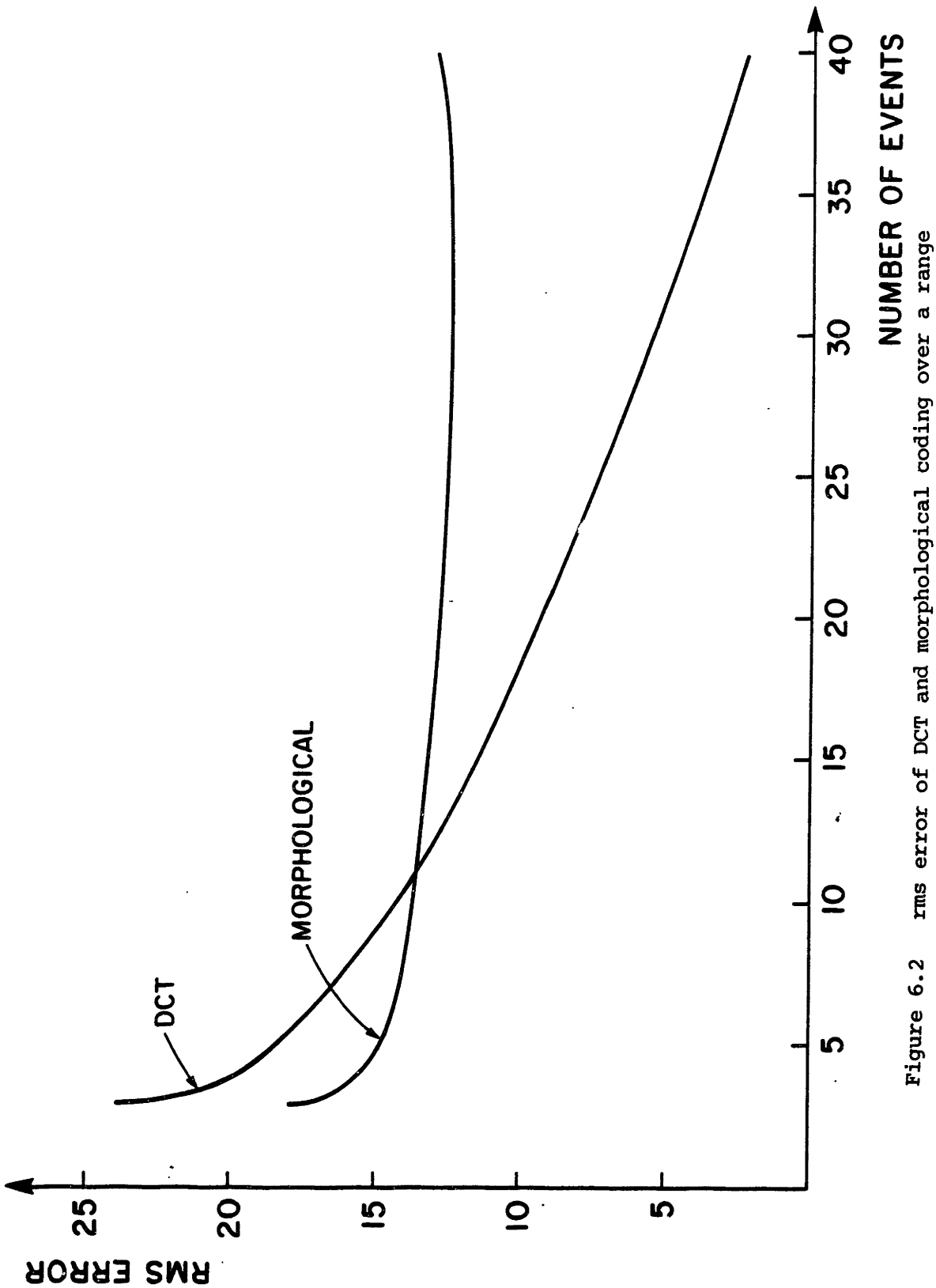


Figure 6.2 rms error of DCT and morphological coding over a range of bandwidths.

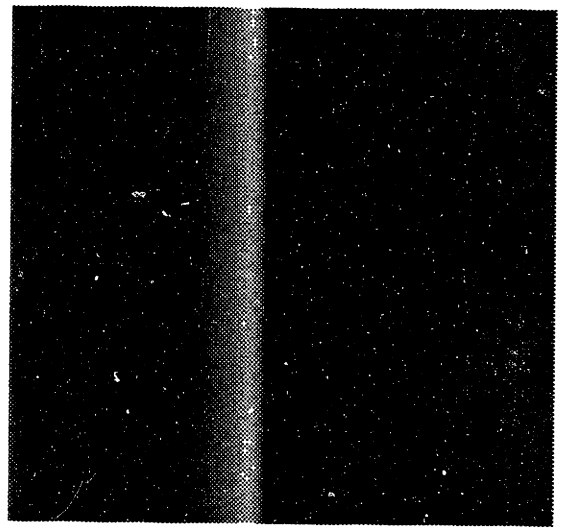


Figure 6.3 (a) Original image: smooth edge.

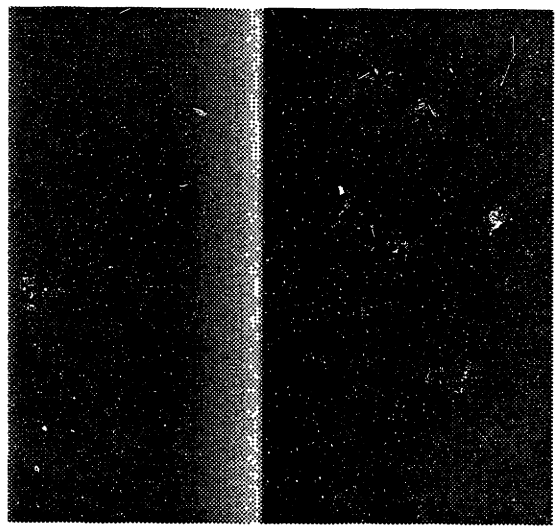


Figure 6.3 (b) Original image: meduim-sharp edge.

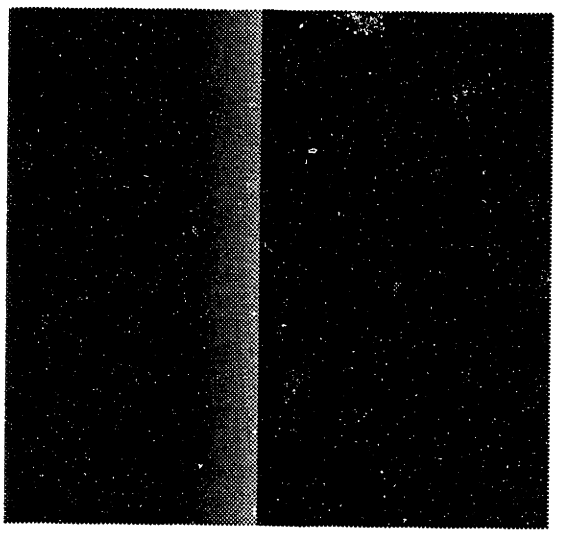


Figure 6.3 (c) Original image: sharp edge.

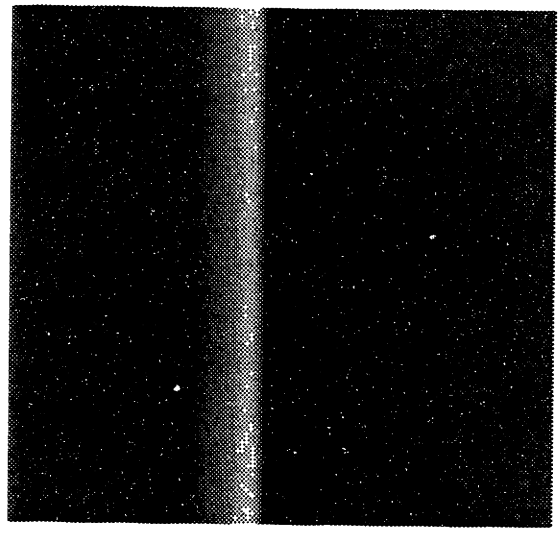


Figure 6.3 (d) Near perfect DCT reconstruction of the smooth edge of Fig. 6.3(a).

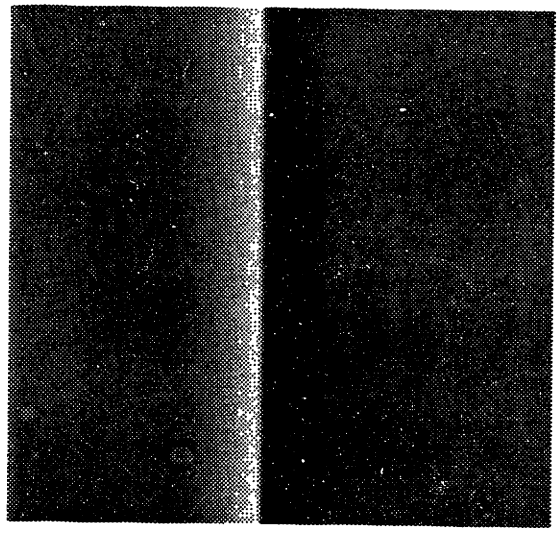


Figure 6.3 (e) DCT reconstruction of the medium-sharp edge of Fig. 6.3(b) at the same bit rate as Fig. 6.3(d).

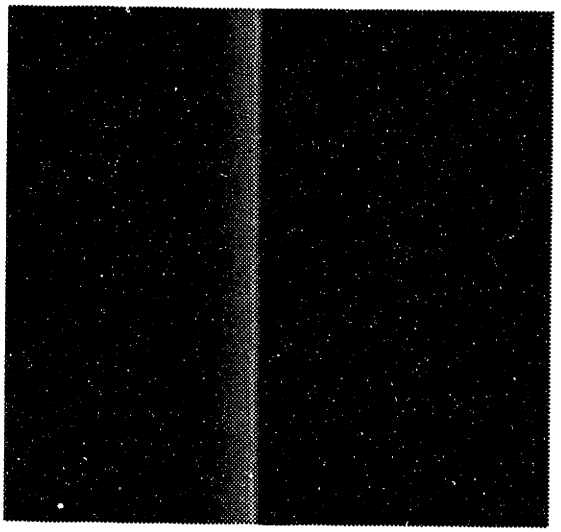


Figure 6.3 (f) DCT reconstruction of the sharp edge of Fig. 6.3(c) at the same bit rate as Fig. 6.3(d).

6.3 The basis functions are attached to image features

While morphological coding pegs its basis functions, i.e. morphemes, to the main image features, transform techniques lack this flexibility. An example of this is shown in fig. 6.4. Fig. 6.4a shows the original image of a set of squares. The stroked, i.e. morphologically coded, image is shown in fig. 6.4b. The only imperfections in the latter image are the slight horizontal streak on both sides of each square. The cause of these streaks was explained in subsection 5.2.4 in chapter 5. If the set of squares happen to coincide with the DCT block doundaries, as shown in fig. 6.4c, DCT will code the image with zero error and with two-thirds the bandwidth of the morphologically coded image. However, when the squares do not coincide with the block doundaries, the bandwidth required for maintaining the same quality significantly increases for DCT while remaining constant for morphological coding. Equivalently, if the bandwidth is held constant, image quality drastically degrades. Figs. 6.4d-6.4g illustrate the latter case where the bandwidth was fixed at that of the morphologically coded image of fig. 6.4b.

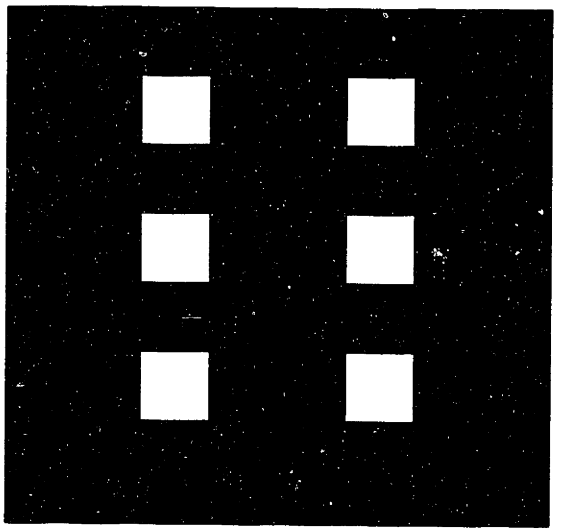


Figure 6.4 (a) Original image.

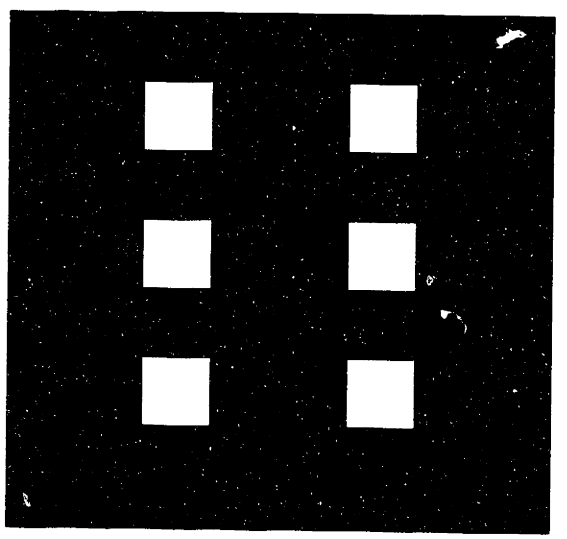


Figure 6.4 (b) Morphologically constructed (stroked) image.

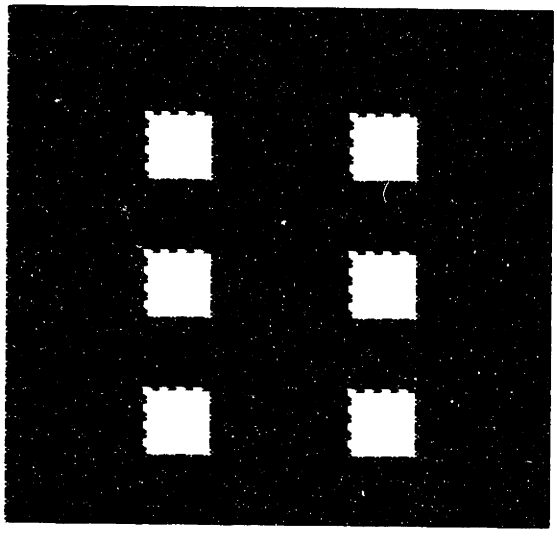


Figure 6.4 (c) When the square boundaries coincide with DCT block boundaries, DCT yields perfect reconstruction at two-thirds the bit rate of the morphologically coded image in Fig. 6.4(b).

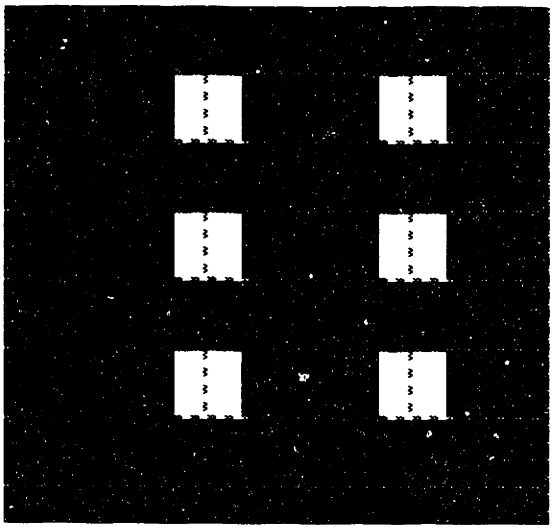


Figure 6.4 (d) Block boundaries bisect squares.

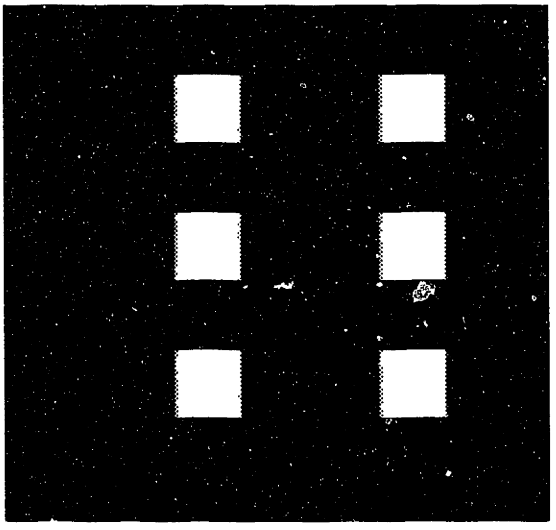


Figure 6.4 (e) DCT reconstruction of Fig. 6.4(d) at the same bit rate as the stroked image in Fig. 6.4(b).

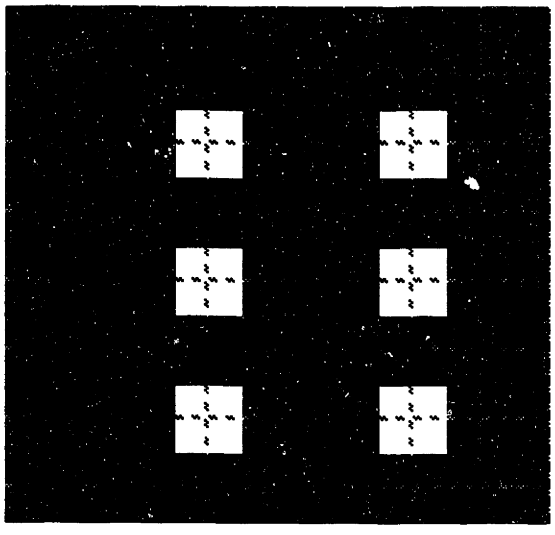


Figure 6.4 (f) Block boundaries quadrasect squares.

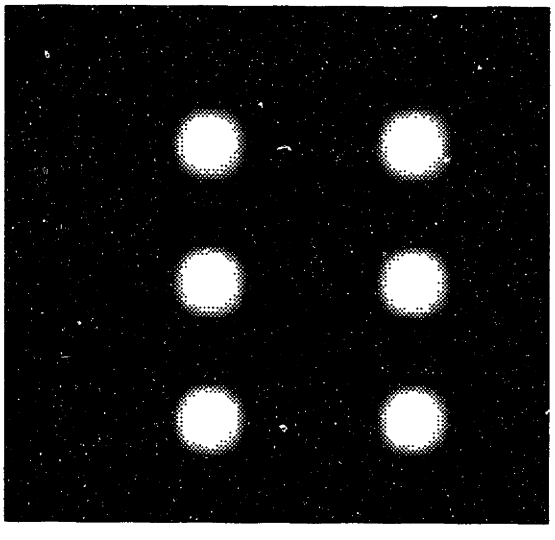


Figure 6.4 (g) DCT reconstruction of Fig. 6.4(f) at the same bit rate as the stroked image in Fig. 6.4(b).

CHAPTER 7

MORPHOLOGICAL CODING IN A DIFFERENT PERSPECTIVE

In the previous chapters, the study of morphological coding presented the notion of the "morpheme" in association with a set of basis functions, each of which is characterized by a number of parameters, namely : position, type, and amplitude. While such spatially distributed functions can epitomize the morpheme notion, the concept of morphological coding is of a wider scope. This chapter offers a view of morphological coding from a different angle; the "morpheme" is proposed to be a local measure of a certain characteristic of the signal. One such local measure is the "direction" along which the signal exhibits the highest correlation locally. For example, near an edge the "morpheme" would be the direction along the edge.

The following sections demonstrate the validity of the direction of maximum local correlation, (henceforth referred to as "the direction"), as a morpheme, via spatial-domain processing. Later, the stage will be set for future work to exploit the direction in transform-domain processing to achieve more significant bandwidth savings.

7.1 Detecting the direction on a pel-by-pel basis

A given pel in an image can be predicted, either perfectly or with a residual error signal, using one or more of the pels that causally precede the given pel, i.e. those that are to the left of the given pel in the same line or are in the lines above it. Such is the principle of the time-honored DPCM. However, the predictor pel, or linear combination of pels, in DPCM is usually fixed throughout the image. Detecting the direction of maximum local correlation, on the other hand, provides each pel, (or group of pels as shown in section 7.3), with its most suitable predictor to significantly reduce the residual error as compared to DPCM. This is demonstrated below.

For the purposes of this section, the direction of prediction for a given pel, x, is sought among pels a thru g which precede pel x causally as shown in fig. 7.1. A search is made for the best linear combination of a pair of consecutive pels in the set a thru g, (i.e. (a,b), (b,c), ...etc.), to predict pel x with the lowest residual error. Once the best pair is determined, the direction is defined as that of a straight line connecting pel x to the point that divides the line joining the pair in the ratio of their linear combination. For example, if the best linear combination of a pair to predict pel x is (1/2)a+(1/2)b, the direction makes a positive 150 degrees with the positive x-axis, (where "positive" here follows the standard convention); if the combination is (0)a+(1)b the angle is 135. Since maximum computational

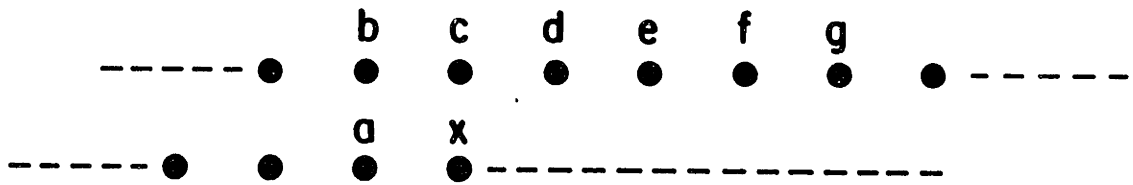


Figure 7.1 Pels on two image lines, showing those used for choosing a predictor for pel x.

efficiency is not of prime concern here, an exhaustive search of the best direction is made among all directions spanning the possible linear combinations of all pairs (a,b) thru (f,g), incrementing the direction one degree at a time from 14.04 degrees, (where the direction connects pel x to pel g), to 180 degrees.

The direction detected as above for each pel may not be unique. This is demonstrated in fig. 7.2b where the direction, or directions, for each pel in the area of the right eye, (fig. 7.2a), of the subject of fig 6.1a, (see chapter 6), is plotted. This ambiguity can be resolved by observing the following criterion : conformity among directions in a neighborhood is desired for two reasons, 1) The global behavior of individual-pel directions should reflect the flow of features in the image; this would confirm the validity of the directions to stand as morphemes representing the image, and, 2) This opens the possibilty of coding the directions using transform techniques to exploit the energy compaction inherently offered by such conformity.

To implement the above criterion, the following method is used. If a pel has more than one direction, each is compared to the directions of the eight immediately surrounding pels. The directions of the surrounding pels "elect" one of the directions in the center pel in the following manner : each surrounding direction casts its "vote" to the center direction which has the least angle difference with it. When a surrounding pel has more than one direction, they equally

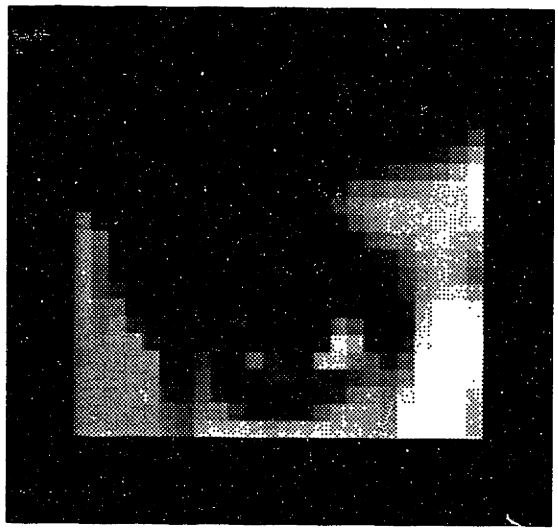


Figure 7.2 (a) The right eye of the subject in Fig. 6.1(a).

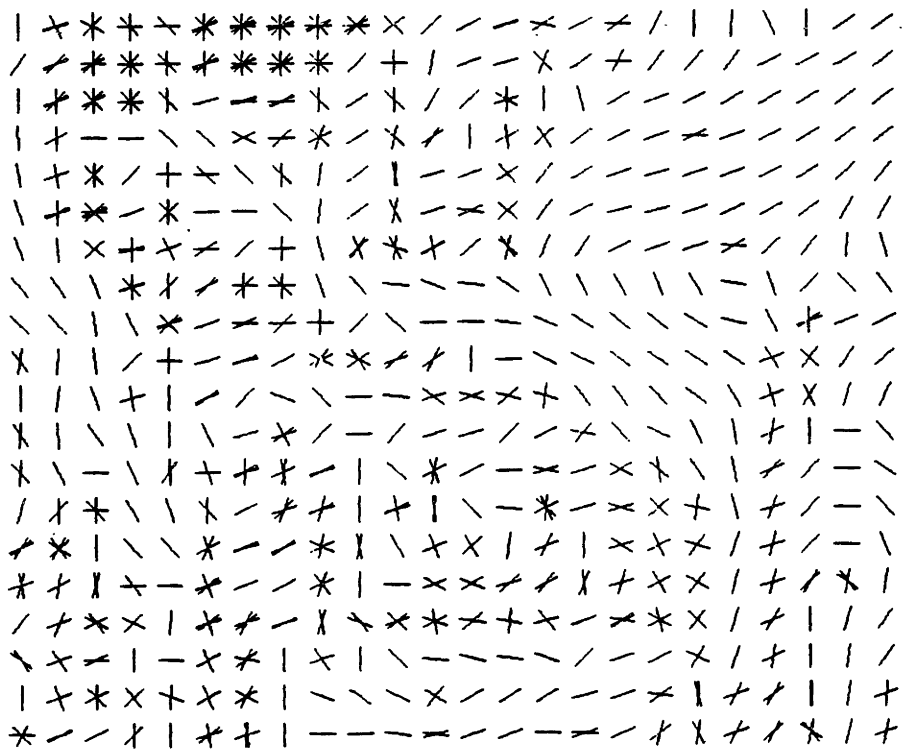


Figure 7.2 (b) All possible prediction directions for each pel in Fig. 7.2(b).

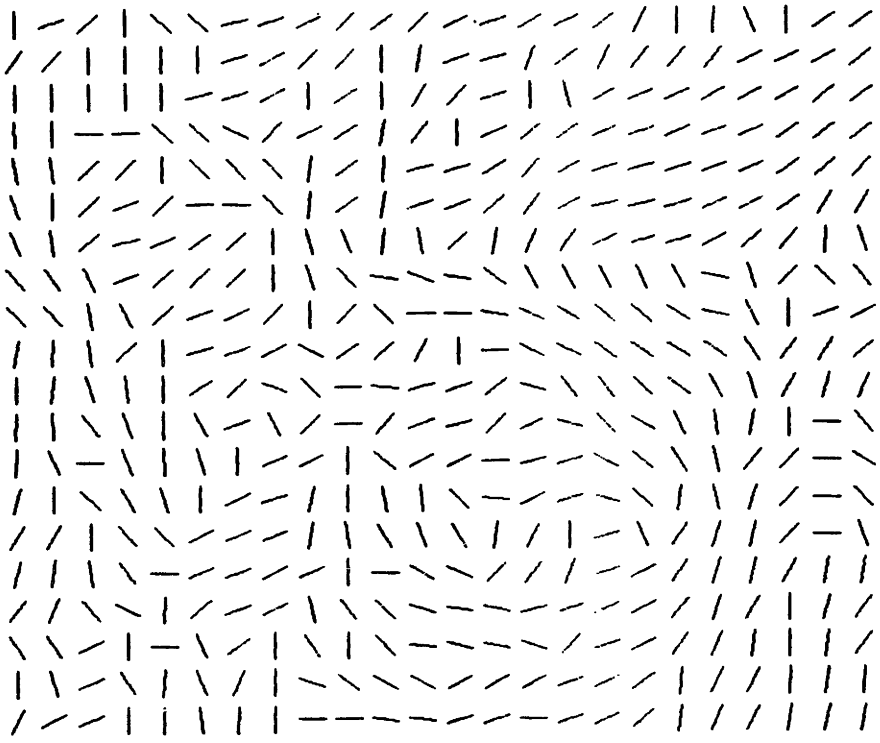


Figure 7.2 (c) "Elected" directions among those of Fig. 7.2(b).

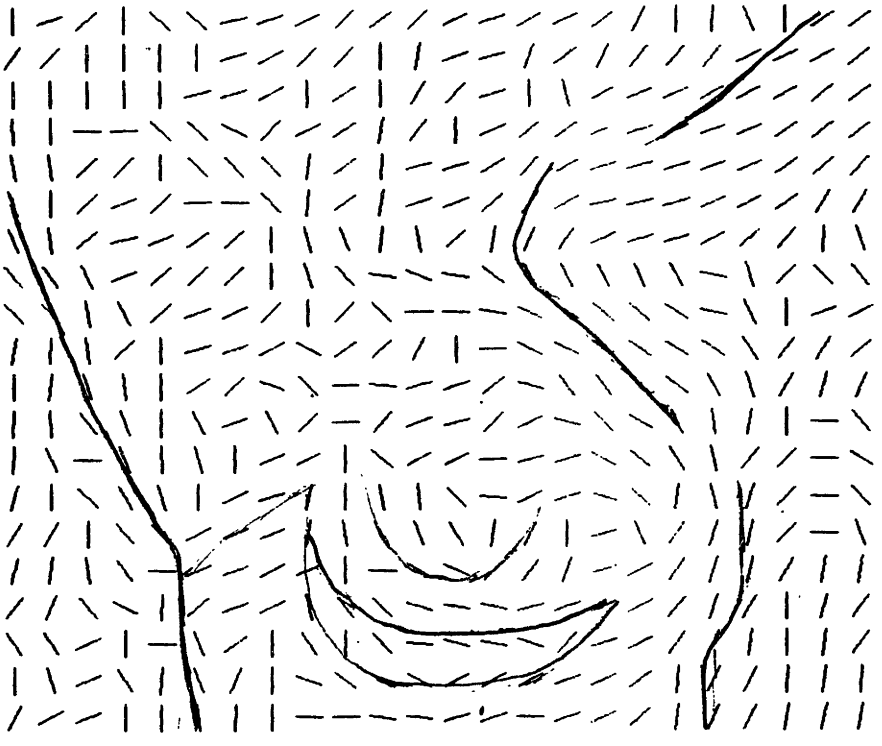


Figure 7.2 (d) The flow of features of the image of Fig. 7.2(a).

share the vote of that pel; however, in case of a pel that causally precedes the center pel, only the "winner" of its election gets to vote. The so-called winners are plotted in figure 7.2c. Figure 7.2d demonstrates how the global behavior of the individual-pel directions of figure 7.2c follows the flow of image features.

7.2 The performance of individual-pel directions

Using the method outlined in the previous section, the directions on a pel-by-pel basis were detected for the entire image of fig. 6.1a. Pels that still showed a residual error given the best possible prediction direction are marked in fig. 7.3 with "X"; those that were perfectly predicted are marked with "-". As the figure suggests, only §15% of the pels required innovations.

Even though individual-pel directions yield a drastic improvement in the percentage of pels that require innovations as compared to DPCM, (§15% vs §85%), the condition that each pel retains its own direction would incur an intolerable overhead in an overall coding scheme. This is true even if transform techniques are used to code the directions since the intrinsic requirement of precision in individual-pel directions to produce the low 15% mentioned above would limit the savings offered by transform coding. Therefore, a method that produces less direction overhead should be used. Such method is presented in the next section.

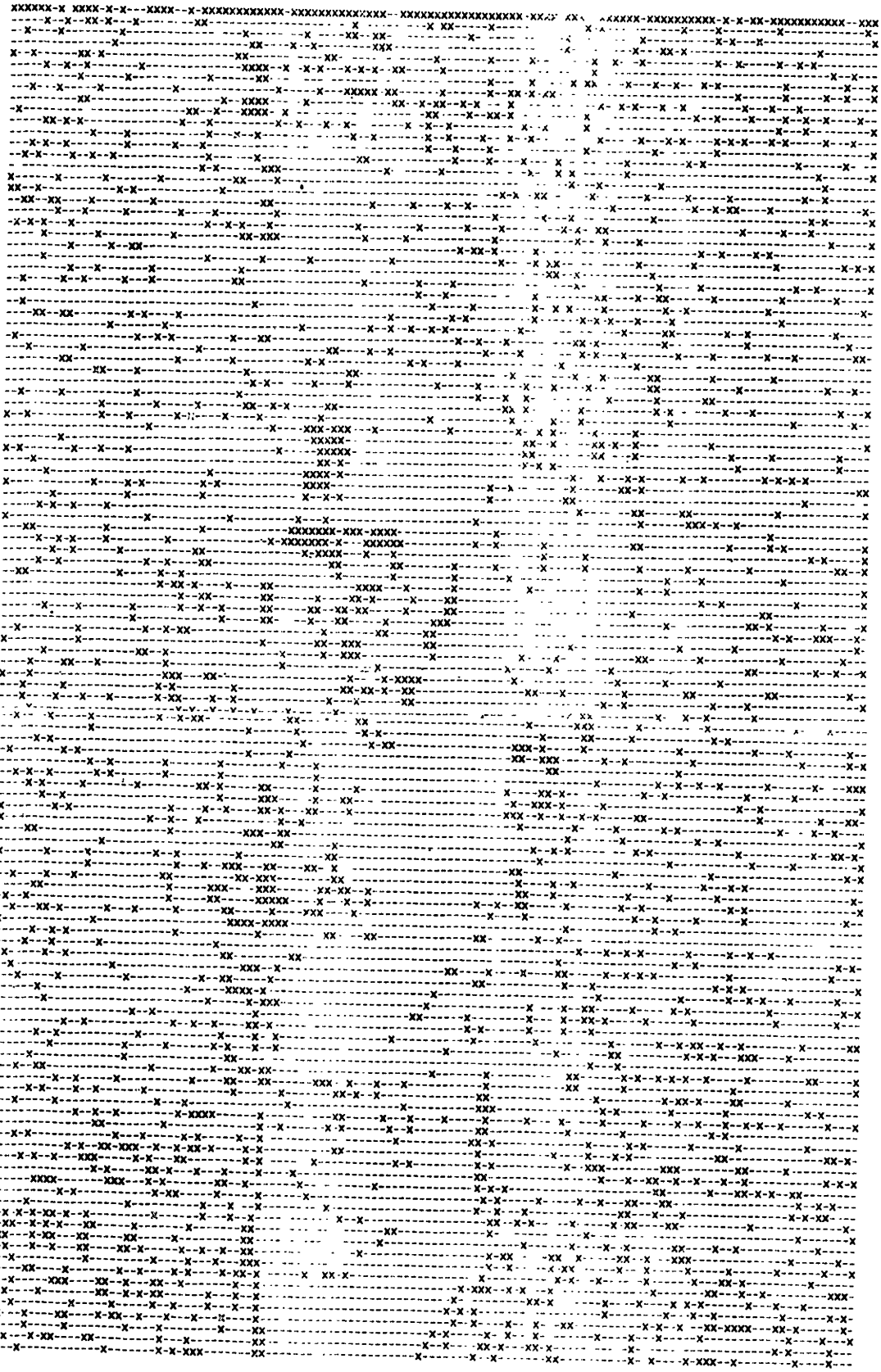


Figure 7.3 Pels of the image of Fig. 6.1(a) requiring innovations after prediction with individual-pel directions.

7.3 Detecting the direction for a block of pels

Instead of determining the direction for each individual pel, the direction which minimizes the error, e.g. the rms error, over a block of pels can be sought. For the purposes of this section, a block size of 4X4 pels was chosen. This would reduce the overhead mentioned at the end of the previous section by a factor of 16. However, the amount of innovations would significantly increase.

The method for detecting block directions is identical to that outlined in section 7.1 except for the error criterion : instead of using the magnitude of the error for an individual pel, the rms error calculated over the 4X4 block is used for determining the optimum block direction. Figure 7.4 shows how block directions follow the flow of image features.

7.4 The performance of block directions

Since the direction overhead in the method of block directions is quite low, the predicate for judging the performance of this method would be the bit-rate needed to convey the amount of information in the innovations. The standard deviation of a given variable represents a measure of the number of bits needed to code the variable for a given accuracy. Hence, comparing the standard deviation of the error signal, i.e. the innovations, produced by the method of block directions with that produced by DPCM serves as a good

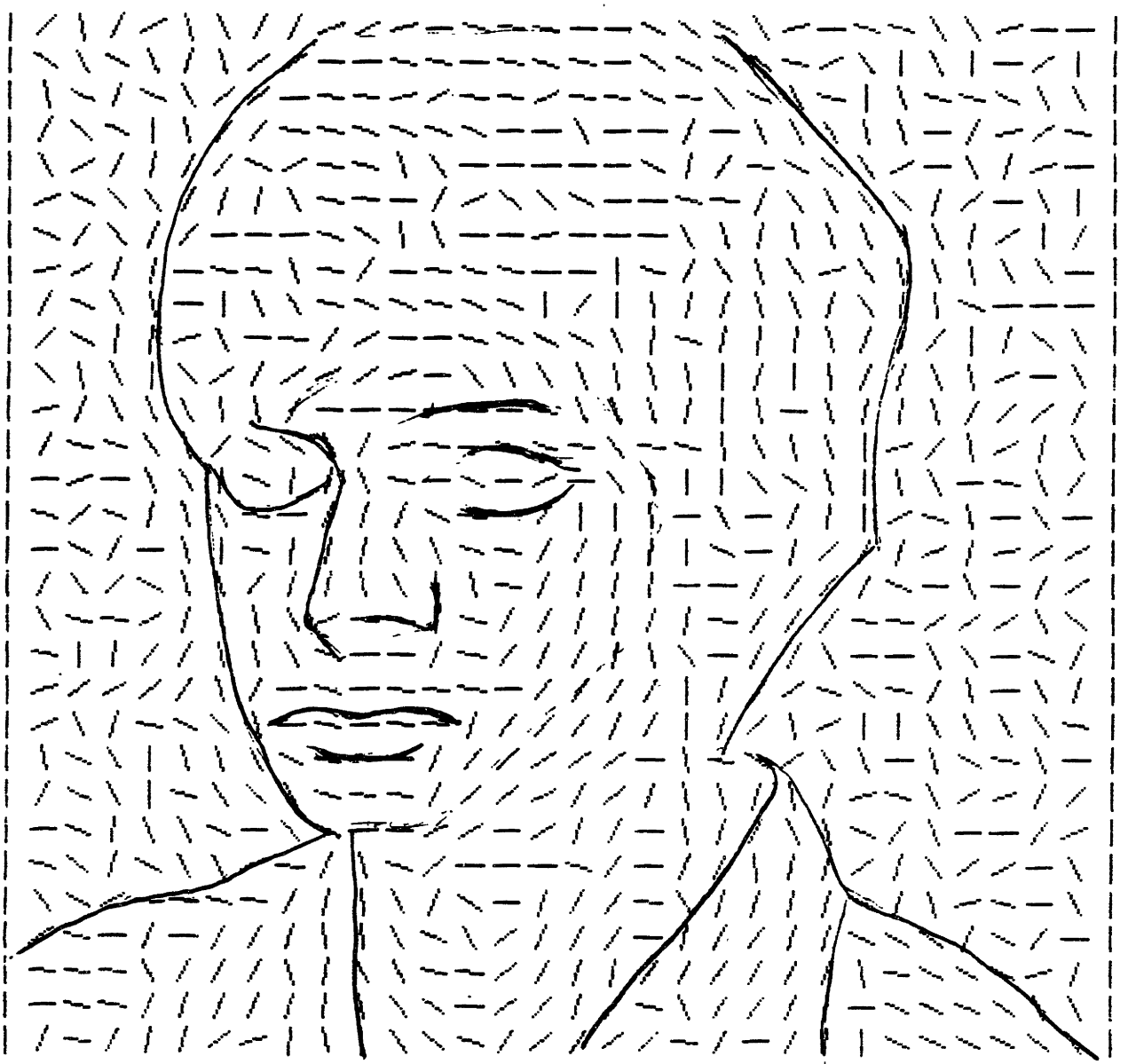


Figure 7.4 Block directions showing the flow of the main features of the image of Fig. 6.1(a).

indication of the amount of bandwidth savings offered by the former over the latter.

The standard deviation of the innovations for the image in fig. 6.1a was calculated once for the block direction method, and again using a fixed predictor throughout the image. The block direction method yielded a standard deviation which is one-half that produced by a fixed predictor DPCM : 10.2 vs 19+-1. The latter number was determined as an average of results obtained using four different cases of a fixed direction for the entire image; namely 180, 135, 90, and 14 degrees. Based on the usual log-variance method of bit-assignment to a random variable, the above results indicate a savings of approximately one bit per pel over DPCM. This amount of savings achieved by the block direction method significantly outweighs its overhead for coding the direction.

7.5 Towards a more efficient use of the direction morpheme

The first four sections of this chapter introduced the direction morpheme and demonstrated its validity for efficient image representation. The sections showed that the global behavior of local directions follows the flow of main image features. In addition, judging by the standard deviation of the innovations produced by each method, the direction morpheme achieved the significant bit-rate reduction, for a comparable image quality, of one bit per pel over DPCM. To achieve both the bandwidth savings and the image quality necessary for low-bandwidth teleconferencing, however, it may

be that the promise of the direction morpheme lies elsewhere; namely, in transform coding techniques.

The main reason for the limitation in the bandwidth savings offered by the direction morpheme in the spatial domain is the following. Although this morphological coding method offered a bit-rate reduction of approximately one bit per pel over that of the DPCM method, it still required error signal transmission for every pel, even if the pel was perfectly predicted. Coding methods with such a requirement are, bar entropy coding, inherently lower bounded by a bit-rate of one bit per pel. Entropy coding techniques may provide some, but not significant, improvement. This is because, contrary to the case of fig. 7.3 where run-length coding might be applied to the runs of dashes, the innovations resulting from block direction coding don't usually offer such exploitable redundancies. Transform coding, on the other hand, has the potential of significantly breaking the one-bit-per-pel barrier owing to its ability to exploit the energy compactness of the image spectrum. Such compactness, with the associated differences in standard deviations among the decorrelated frequency components, allows independent, unequal, bit assignments to these components, thereby leading to an average of less than one bit per pel.

It is beyond the scope of this thesis to discuss the ramifications of exploiting the direction morpheme in the transform domain. Research work in this vein is currently being conducted at RLE by Mr. Philippe Cassereau under the

supervision of Professor David Staelin.

CHAPTER 8
SUMMARY AND CONCLUSIONS

This thesis has developed a procedure for, evaluated the performance of, and indicated the limitations of the use of the morphological method of coding two-dimensional facial images. The first section below summarizes the highlights of morphological coding as implemented in this thesis. The second section states the fundamental issues of morphological coding as identified during the course of this research.

8.1 Summary

1) The lack of rigorous derivation for a set of optimum morphemes prompted the arbitrary choice of a set of "gaussian" and edge-shaped basis functions to code one-dimensional images. The optimum filters theoretically derived to detect a given morpheme of such a set in a background of other such morphemes and white noise closely resembled in shape those deduced for neurological networks in the human visual system [20]. This suggests that one intermediate representation for images in the human visual system may be the sum of gaussians or both gaussian and other morphemes.

2) The morpheme detectors lacked the ability to respond to (i.e. detect) low-frequency structure in the image. A low-frequency estimator, namely the widest gaussian, had to be used to compensate for that. The importance of this estimator was strikingly evident when test patterns were processed once including it, and again without it.

3) The approximations made in implementing the optimum morpheme detectors, and the following estimation procedure which used only first and second-order corrections for inter-symbol interference led to results such as those shown in fig. 8.1 for coding individual scan lines separately.

4) When reconstructed one-dimensional images (scan lines of a head-and-shoulder image) were arranged to form a two-dimensional image, the result was less than satisfactory. The major objection to the quality of images so reconstructed was discontinuities evident in the vertical direction (i.e. horizontal streaks). These were caused mainly by instabilities in this morpheme estimation process due to marginal inclusion or exclusion of morphemes (as they fall above or below a detection threshold), and to changes in the sequence in which they are estimated.

5) To alleviate horizontal streaks in a two-dimensional image, and to achieve the maximum possible bit-rate reduction

this morphological coding method is capable of, processing to exploit the image charateristics in the vertical direction should be employed. This was realized through the grouping of events (i.e. estimated morphemes) on adjacent image lines into strokes (vertical runs) similar to the brush strokes in an artist's painting. These strokes were coded using rms-error interpolation fits of the three stroke parameters: position, width, and amplitude. Such efficient stroke-coding carried an inherent smoothing process which went a long way towards removing the horizontal streaks from the image.

6) The limit of this method of morphological coding to produce acceptable quality images at a low bit rate is epitomized in its performance on a 128 X 128 pel image (fig. 4.2(e)). This image was reconstructed (fully stroked) at a bit rate of 0.5 bit/pel, and with a (raw) rms error of 7.5 (units out of 255).

7) Upon close examination and detailed study, all of the performance limitations of this morphological coding method in one-dimension were directly traceable to the strong inter-symbol interference among the non-orthogonal morphemes. This interference causes apparent shifts in position, width, and amplitude of the estimated morpheme. To realize a more significant correction for this than the first and second-order corrections employed here (for the sake of computational efficiency), a more elaborate iterative or simultaneous-

solution method must be employed. The computational cost of such a method, though, may border on the prohibitive.

8) Limitations encountered in the stroking process were mainly due to change of order of event estimation from line to line, the sequential method of fitting concatenated parabolas to arbitrary-shaped strokes, change in the morpheme set which represents a given signal, and discontinuities in the "fat-g's" (the low-frequency estimator). Solutions to such problems would generally involve a significant increase in the computational cost.

9) The following features of this method of morphological coding, especially as compared to transform coding, were demonstrated 1) Degradation of image quality and recognizability at very low bandwidths (less than 5 events per line) is much more graceful than results achieved with DCT, 2) Bit rate for image coding can be independent of frequency content of the image, and 3) The bit rate of image coding is independent of feature location within the image.

10) The promise of morphclogical coding to achieve higher quality pictures at lower bandwidths may lie in a different perspective. Such a perspective would associate the notion of the "morpheme" with a local measure of a characteristic of the image rather than with a set of basis functions. Research being presently conducted will focus on

utilizing that notion in the transform domain.

8.2 Conclusions

The above section presented a summary of the procedural steps involved in implementing, evaluating the performance of, and indicating the limitations of morphological coding. During the course of this research effort, the following fundamental issues were found to underlie the concept of morphological coding.

1) The choice of morphemes.

The set of morphemes chosen in this thesis for the morphological coding of images was introduced in chapter 1 as an arbitrary choice owing to the lack of a more rigorously derived morpheme set. However, at the conclusion of this chapter, it is possible to say that the following issues further prompt and/or supported this particular choice.

a) Independent research [20] has suggested that gaussians, or gaussians and other morphemes, may be involved in the manner in which the human visual system interprets images. Since the end-user of all image coding is the human, it is appropriate that coding tools that conform with the way the human eye reconstructs images be used.

b) As the number of morphemes included in reconstructing a one-dimensional image increases, the rms reconstruction error decreases as shown in fig. 8.1. The floor of the error shown in this figure is largely

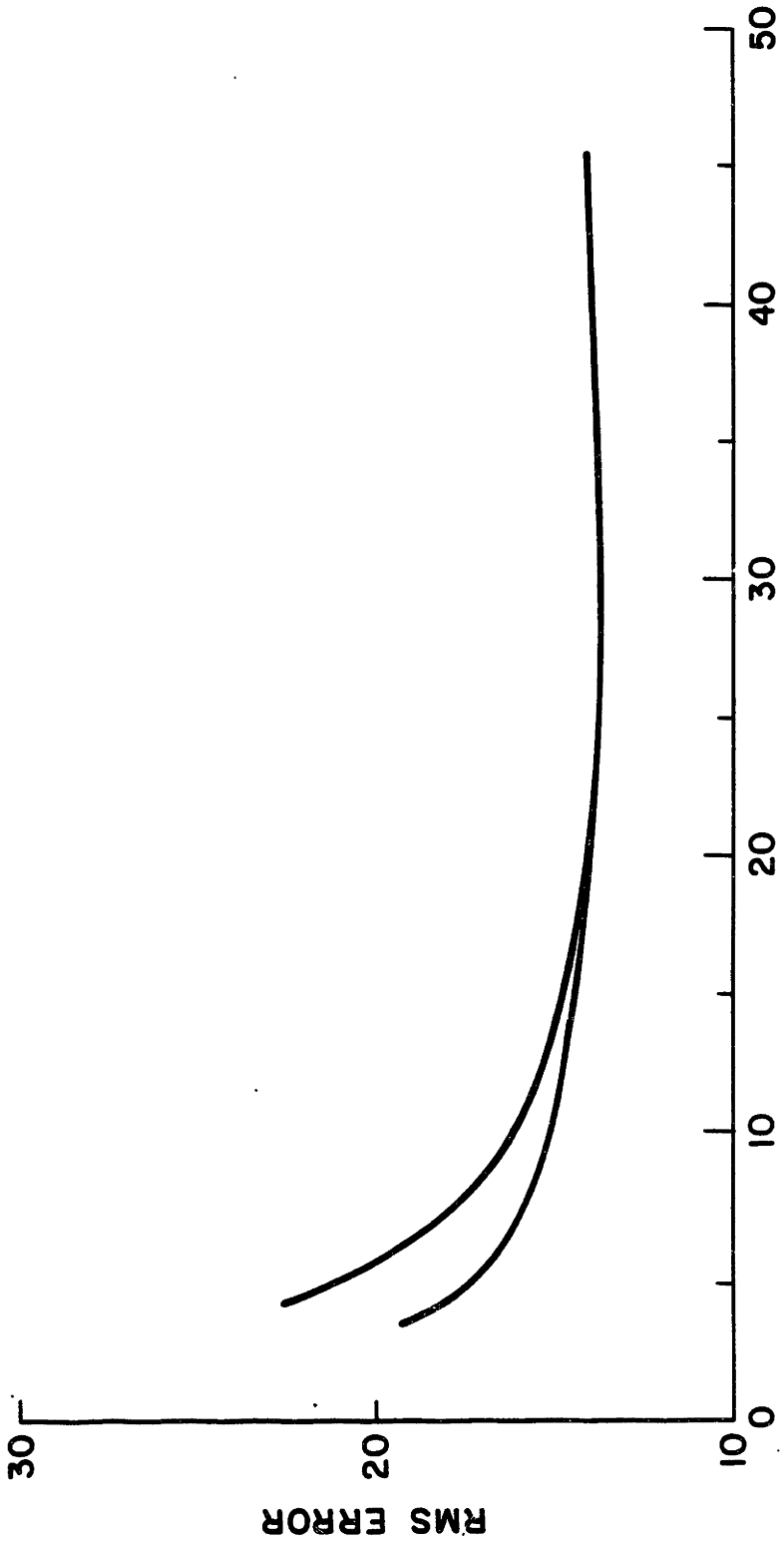


Figure 8.1 rms error vs. average number of events (gaussian and edges) per line for the image shown in Fig. 6.1.

attributable to the approximations made in the process of morpheme detection and estimation, as shown below.

c) The general solution for the problem of rigorous derivation of such a set of non-orthogonal morphemes is not known. Therefore, it is not obvious that a different set from the one chosen here would yield higher quality pictures at lower bandwidths. In principle, it is possible to obtain an indication of that by varying the bell shape used here for the gaussians and, correspondingly, the shape of the edges to see if any improvements at all in the reconstruction of images would result.

2) Practical considerations in the implementation of morphological coding.

The major cause of errors resulting from this aspect is the strong intersymbol interference resulting from the overlap among the non-orthogonal morphemes. As pointed out earlier, to achieve more significant corrections for this intersymbol interference than what is provided by the first and second-order corrections used in this thesis, it is necessary to use iterative or simultaneous methods of solution. Such method would represent the automation of morphologically reconstructing an image scan line "by hand", such as using a mouse on a video display to select, move, and adjust a number of morphemes to fit a given intensity distribution on a scan line. While it is possible to automate such iterative techniques, the associated computational cost

would be a problem to contend with.

That the error resulting from such intersymbol interference is largely responsible for setting the limitation on the one-dimensional reconstruction procedure to achieve higher quality images can be explained as follows:

a) The magnitude of the floor of the error in fig. 8.1 is comparable to the average rms error (for a given detection threshold) obtained for reconstructing controlled test patterns in section 5.1, where the errors due to intersymbol interference were isolated.

b) The error-curve in fig. 8.1 deflects upward towards the end, as the intersymbol interference increases with the increase of the number of events per line.

3) The limitation of the use of simple functions to reconstruct arbitrarily complex structures.

The effect of this limitation was observed most clearly in the stroking process of combining one-dimensional events on successive image lines into vertical runs. Although the three stroke parameters (position, width, and amplitude) were allowed to vary independently along the stroke, there still remained an inherent simplicity in these strokes imposed by the requirement to maintain code efficiency. Such intrinsic simplicity of the basic constructs (i.e. the strokes) used in coding images caused the representation of arbitrarily complex image features to change from one part of the image to another. Such jumps in representation lead to bothersome discontinuities

compared to the original image, and to a loss of track of major features of the original image owing to such breaks in the flow of the code.

This type of shortcoming is universal in all problems of reducing an arbitrarily complex structure into a set of simple, basic ones. An example would be the problem of speech interpretation. When a general text of speech is interpreted, individual phrases can be construed perfectly in and of themselves. However, once the speech is reconstructed by juxtaposing the individual phrases, the overall meaning could be lost, or altogether altered.

Two general approaches may be followed to tackle this problem. First, the domain of structures to be coded can be narrowed down to limit the structure complexity with which the coding algorithm has to deal. For example, in the case of image coding the type of images to be coded can be restricted to front-looking subjects with fixed eye-level (e.g. police mug-shots). The image coding algorithm in this case would possess a fair amount of prior knowledge of, say, the location of the main image features. This would help prevent the multiplicity of representations of any given feature.

Second, to resolve the representational ambiguity which degrades quality and code efficiency of the reconstructed structure, especially if the coding algorithm has to deal with arbitrarily complex structures, a higher level of intelligence is needed to make decisions to resolve such ambiguities. An algorithm utilizing such type of artificial intelligence

would recognize the continuous flow in main features at a broader scale than the narrower view a simpler algorithm would otherwise provide, and would accordingly make the right decisions to resolve any representational ambiguities disrupting this flow.

One should note that the optimum solution to any such problem may consist of some combination of applying both of the above approaches. The details of implementing such a solution in the case of morphological coding of images remains a subject for future research.

REFERENCES

1. P. A. Wintz, "Transform picture coding," Proc. IEEE, 60, No. 7, pp. 809-820, July 1972.
2. W. H. Chen and C. H. Smith, "Adaptive coding of monochrome and color images," IEEE Trans. Commun., COM-25, pp. 1285-1292, Nov. 1977.
3. S. C. Knauer, "Real-time video compression algorithm for hadamard transform processing," Proc. SPIE, vol. 66, pp. 58-69, Aug. 1975.
4. D. E. Troxel, W. F. Schreiber et al., "A two-channel picture coding system : I - real time implementation," IEEE Trans. Commun., COM-29, No. 12, pp. 1841-1849, Dec. 1981.
5. J. K. Yan and D. J. Sakrison, "Encoding of images based on two component source model," IEEE Trans. Commun., COM-25, pp. 1315-1328, Nov. 1977.
6. D. H. Staelin and D. I. Szabo, "Adaptive image coding algorithms incorporating two or three bands," unpublished paper.
7. P. Elias, "Predictive Coding," IRE Trans. Inform. Theory, IT-1, pp. 16-33, Mar. 1955.
8. H. Yasuda and H. Kawanishi, "Predictor adaptive DPCM," in Proc. SPIE Conf. Applications of Digital Image Processing, vol. 149, pp. 189-195, Aug. 1978.

9. F. deJager, "Delta modulation, A method of PCM transmission using a 7-unit code," Philips Res. Rep., pp. 442-446, Dec. 1952.

10. D. Marr, "Early processing of visual information," Philos. Trans. Roy. Soc. London Ser. B, vol. 257, pp. 483-524, 1976.

11. R. Nevatia, "Locating object boundaries in textured environments," IEEE Trans. Comput., vol. 25, pp. 1170-1175, 1976.

12. T. Pavlidis, "Structural Pattern Recognition," Springer, N. Y., 1977.

13. A. Rosenfeld and A. C. Kak, "Digital Picture Processing," vol. 1 & 2, Academic Press, N. Y., 1982.

14. A. N. Netraveli and J. O. Limb, "Picture coding : A review," Proc. IEEE, 68, No. 3, pp. 366-406, Mar. 1980.

15. W. K. Pratt, "Bibliography on digital image processing and related topics," Univ. of Southern California, USCEE Rep. 453, Sept. 1973.

16. A. Rosenfeld, "Image Modeling," Academic Press, N. Y., 1981.

17. D. Marr and E. Hildreth, "Theory of edge detection," M.I.T., Art. Intel. Lab., A. I. Memo No. 518, 1979.

18. H. L. VanTrees, "Detection, Estimation, and Modulation Theory, Part 1," pp. 24-26, John Wiley & Sons, N. Y., 1968.

19. ibid, pp. 254ff.

20. H. R. Wilson and J. R. Bergen, "A four mechanism model for threshold spatial vision," Vision Research, 19, pp. 19-32, 1979.

21. G. Strang, "Linear Algebra and its Applications," Academic Press, N. Y., 1980.

22. S. A. Hovanessian, "Radar Detection and Tracking Systems," pp. 9.1-9.8, Artech House, M. A., 1973.

23. W. H. Beyer, "CRC Standard Mathematical Tables," 25th ed., pp. 308-309, CRC Press, Florida, 1980.

APPENDIX 1

COMPUTER PROGRAMS ASSOCIATED WITH CHAPTERS 1 & 2

```
;                          **********PROGRAM : BENCH**********
;
;
;       GENERATES THE OUTPUT OF UPTO 21 G-FILTERS FOR EACH IMAGE LINE.
;       SPECIFICATIONS FOR ANY GIVEN RUN OF THIS PROGRAM ARE READ FROM
;       FILE "ASFL".

        .TITL BENCH
        .TXTM 1

        .ZREL

C0:     0
C1:     1
C2:     2
C5:     5
C6:     6
CN6:    -6
CN7:    -7
CN8:    -10
C10:    12
K10:    0
CCR:    15
C60:    60
CN11:   -13
CSP:    40
CCOM:   ",
C512:   1000
CLB:    377B7
ACBUFF: .+1*2+1             ;THE ADDRESS OF THE RIGHT BYTE OF CBUFF.
CBUFF:  0
SUM:    0
INFL:   .+1*2
        .BLK 3
DUMY:   .+1*2+1
        .BLK 1
OUTF:   .+1*2
        .BLK 3
ASFL:   .+1*2
        .TXT "ASFL"
NPEL:   0
NPEL2:  0
NPPEL:  0
OM64:   0
M64:    0
T64:    0
NM64:   0
NCL:    0
OFSTC:  0
FSTC:   0
SCNC:   0
```

```
THDC:     0
NSL:      0
TNSL:     0
ASL:      .
SL:       .BLK 6
SCHK:     0
ONPAIR:   0
NPAIR:    0
TNPAIR:   0
ALPR:     .
LPR:      .BLK 25
ASKPR:    .
SKPR:     .BLK 25
RAN:      .RAN
ER:       .ER
AOL:      OL-1
ABOL:     OL*2
ROL:      .ROL
ARLN:     .
RLN:      LN1
          LN2
          LN3
          LN4
          LN5
          LN6
ATLN:     TLN-1
.TL1:     TL1
.TL2:     TL2
LSTO:     0
STL:      .STL
BOL:      0
EOL:      0
IOL:      .IOL
CPOL:     .COL
CPL:      .CL
IOSL:     .IOS
CPSL:     .CSL
PAIRC:    0
LE1C:     0
LE2C:     0
FIL:      0
LIL:      0
.STP:     STOP

          .NREL

STRT:     LDA       0,ASFL      ;READ INPUT FROM FILE ASFL.
          SUB       1,1
          .SYSTM
          .OPEN 5
          JMP       @ER
```

```
LDA             0,INFL          ; READ INPUT IMAGE NAME.
LDA             1,C5
.SYSTM
.RDS 5
JMP             @ER
LDA             0,DUMY
LDA             1,C1
.SYSTM
.RDS 5
JMP             @ER
LDA             0,OUTF          ; READ FILE NAME FOR STORING THE OUTPUTS OF THE
LDA             1,C5            ; G-FILTERS FOR EACH IMAGE LINE.
.SYSTM
.RDS 5
JMP             @ER
LDA             0,DUMY
LDA             1,C1
.SYSTM
.RDS 5
JMP             @ER
LDA             0,ACBUFF
JSR             @RAN
STA             2,NPEL          ;READ NO. OF PELS PER LINE.
LDA             3,CN6
MOVZR           2,2
INC             3,3,SZR
JMP             .-2
STA             2,0M64          ;THIS IS THE RESULT: NPEL/64.
 STA             2,M6
STA             2,T64
NEG             2,2
STA             2,NM64
JSR             @RAN
STA             2,NCL           ;NO. OF G-FILTERS TO COMPUTE , MAX.=24.
JSR             @RAN
STA             2,NSL           ;NO. OF G-FILTERS TO STORE, MAX.=6.
 STA             2,TNS
LDA             2,ASL
STA             2,26
JSR             @RAN
STA             2,@26
DSZ             TNSL
JMP             .-3
JSR             @RAN
STA             2,ONPAIR        ;READ VALUES OF DELAY ELEMENTS FOR
STA             2,NPAIR                     ;COMPUTING THE G-FILTERS HIGHER THAN
STA             2,TNPAIR        ;G8.
LDA             2,ALPR
STA             2,23
LDA             2,ASKPR
STA             2,24
```

```
        JSR     @RAN
        STA     2,@23
        JSR     @RAN
        STA     2,@24
        DSZ     TNPAIR
        JMP     .-5
        JSR     @RAN
        STA     2,FIL       ;READ NO. OF FIRST IMAGE LINE TO PROCESS.
        JSR     @RAN
        STA     2,LIL       ;READ NO. OF LAST IMAGE LINE TO PROCESS.

        JMP     SETC

.RAN:   STA     3,RANR      ;SUBROUTINE FOR READING AN ASCII NO. FROM FILE ASFL,
        LDA     2,C0        ;SPACES ARE IGNORED AND THE NO. IS TERMINATED EITHER
        STA     2,SUM       ;BY A COMMA XOR A CR.
.RN1:   .SYSTM
        .RDS 5
        JMP     @ER
        LDB     0,3
        LDA     2,CSP
        SUB     3,2,SNR
        JMP     .RN1
        LDA     2,CCOM
        SUB     3,2,SNR
        JMP     RNR1
        LDA     2,CCR
        SUB     3,2,SNR
        JMP     RNR1
        LDA     2,C60
        SUB     2,3
        STB     0,3
        LDA     2,C10
        STA     2,K10
        LDA     2,C0
        LDA     3,SUM
        ADD     3,2
        DSZ     K10
        JMP     .-2
        LDB     0,3
        ADD     3,2
        STA     2,SUM
        JMP     .RN1
RNR1:   LDA     2,SUM
        JMP     @RANR
RANR:   0


SETC:   LDA     3,NCL       ;SET COUNTERS OF NO. OF G-FILTERS TO BE COMPUTED.
        LDA     2,C2        ;NCL IS INITIALIZED TO SAY: "THE SECOND G-FILTER HAS
        STA     2,NCL       ;JUST STARTED CALCULATION",(SEE: .IOL ).
```

```
        LDA     2,C6        ;IN ADDITION TO THE FIRST G-FILTER, 6 MORE
                            ;G-FILTERS CAN BE
        STA     2,FSTC      ;COMPUTED WITHOUT SHIFT IN THE ADD OPERATION.
        LDA     2,CN8
        ADDZL#  2,3,SZC
        JMP     .+6
        ADD     3,2         ;NCL )= 8.
        STA     2,THDC
        LDA     2,C1
        STA     2,SCNC
        JMP     OPNS
        LDA     0,C0        ;NCL ( 8.
        STA     0,THDC
        STA     0,SCNC
        LDA     2,CN7
        ADDZL#  2,3,SZC
        JMP     .+2
        JMP     OPNS        ;NCL = 7.
        LDA     2,C1
        SUB     2,3,SNR
        JMP     .+3
        STA     3,FSTC      ;7 ) NCL ) 1.
        JMP     OPNS
        STA     0,FSTC      ;NCL = 1.

OPNS:   LDA     0,FSTC      ;OPEN OUTPUT FILE AND SET UP THE WRITE OPERATIONS.
        STA     0,OFSTC
        LDA     0,INFL
        SUB     1,1
        .SYSTM
        .OPEN 3
        JMP     @ER
        LDA     0,C0
        LDA     3,C1
        LDA     2,FIL
        SUB     3,2
        LDA     1,NPEL
        MUL
        .SYSTM
        .SPOS 3
        JMP     @ER

        LDA     0,OUTF
        .SYSTM
        .OPEN 4
        JMP     @ER
        JMP     @STL

OL:     .BLK 1000
```

```
TLN:    .BLK 27                     ;27 OCTAL IS 23 (DECIMAL) LOCATIONS.
TL1:    .BLK 1000
TL2:    .BLK 1000
LN1:    .BLK 1000
LN2:    .BLK 1000
LN3:    .BLK 1000
LN4:    .BLK 1000
LN5:    .BLK 1000
LN6:    .BLK 1000

.STL:   LDA         2,ATLN          ;STORE ADDRESSES (TL1,TL2) THROUGHOUT ATLN.
        STA         2,25
        LDA         1,.TL1
        LDA         2,.TL2
        LDA         3,CN11
TLLP:   STA         1,@25
        STA         2,@25
        INC         3,3,SZR
        JMP         TLLP
        STA         1,@25

.ROL:   LDA         0,ABOL          ;READ IMAGE LINE.
        LDA         1,NPEL
        .SYSTM
        .RDS 3
        JMP         @ER


.SAU:   LDA         0,CLB           ;INITIALIZE ACCUMULATOR 0 AND AUTO-INCREMENT REGISTERS.
        LDA         2,ASL
        STA         2,26
        LDA         2,ARLN
        STA         2,27
        LDA         2,ATLN
        STA         2,25
        LDA         2,ALPR
        STA         2,23
        LDA         2,ASKPR
        STA         2,24

IOOL:   LDA         2,AOL           ;I/O FOR PROCESSING ORIGINAL LINE.
        STA         2,21
        LDA         1,@26           ;IS G-FILTER #0, I.E. ORIGINAL LINE, TO BE STORED ?
        MOV         1,1,SZR
        JMP         STR1            ;NO.
        LDA         2,@27           ;YES.
        LDA         3,C1
        SUB         3,2
        STA         2,22
        LDA         2,NPEL
        MOVZR       2,2
```

```
        STA     2,NPEL2
STR0:   LDA     2,@21       ;STORE ORIGINAL IMAGE LINE IN OUTPUT FILE.
        MOV     2,1
        ANDS    0,2
        STA     2,@22
        COM     0,0
        AND     0,1
        STA     1,@22
        COM     0,0
        DSZ     NPEL2
        JMP     STR0        ;THIS CAN BE SPREAD OUT TO ACHIEVE GREATER SPEED.
        LDA     2,AOL
        STA     2,21
        LDA     1,@26

STR1:   LDA     3,C1        ;IS G-FILTER #1 TO BE STORED ?
        SUB#    3,1,SZR
        JMP     .+4
        LDA     2,@27       ;YES.
        LDA     1,@26
        JMP     .+2
        LDA     2,@25       ;NO.
        STA     1,SCHK
        SUB     3,2
        STA     2,22
        STA     2,LSTO

POL:    LDA     2,@21       ;START PROCESSING ORIGINAL LINE.
        MOV     2,1
        ANDS    0,2
        MOVZL   2,3
        STA     3,BOL
        COM     0,0
        AND     0,1
        ADD     1,2
        STA     2,@22
.COL:   LDA     2,@21       ;NO DO-LOOP IS USED HERE TO ACHIEVE MAX. SPEED.
        MOV     2,3
        COM     0,0
        ANDS    0,2
        ADD     2,1
        STA     1,@22
        COM     0,0
        AND     0,3
        ADD     3,2
        STA     2,@22
        LDA     2,@21
        MOV     2,1
        COM     0,0
        ANDS    0,2
```

```
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
```

```
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
```

```
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
```

```
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
```

```
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
STA     2,@22
LDA     2,@21
MOV     2,1
COM     0,0
ANDS    0,2
ADD     2,3
STA     3,@22
COM     0,0
AND     0,1
ADD     1,2
STA     2,@22
LDA     2,@21
MOV     2,3
COM     0,0
ANDS    0,2
ADD     2,1
STA     1,@22
COM     0,0
AND     0,3
ADD     3,2
```

```
        STA     2,@22
        LDA     2,@21
        MOV     2,1
        COM     0,0
        ANDS    0,2
        ADD     2,3
        STA     3,@22
        COM     0,0
        AND     0,1
        ADD     1,2
        STA     2,@22
        LDA     2,@21
        MOV     2,3
        COM     0,0
        ANDS    0,2
        ADD     2,1
        STA     1,@22
        COM     0,0
        AND     0,3
        ADD     3,2
        STA     2,@22
        LDA     2,@21
        MOV     2,1
        COM     0,0
        ANDS    0,2
        ADD     2,3
        STA     3,@22
        COM     0,0
        AND     0,1
        ADD     1,2
        STA     2,@22
        DSZ     M64
        JMP     @CPOL

.FPO:   LDA     2,@21
        MOV     2,3
        COM     0,0
        ANDS    0,2
        ADD     2,1
        STA     1,@22
        COM     0,0
        AND     0,3
        ADD     3,2
        STA     2,@22
        DSZ     T64
        JMP     .+2
        JMP     EOLO
        LDA     2,@21
        MOV     2,1
        COM     0,0
        ANDS    0,2
```

```
        ADD     2,3
        STA     3,@22
        COM     0,0
        AND     0,1
        ADD     1,2
        STA     2,@22
        JMP     .FPO            ;THIS CAN BE MADE FASTER BY SPREADING IT OUT.

EOLO:   MOVZL   3,3             ;LINE-END REFELECTION.
        STA     3,EOL

        LDA     2,FSTC
        MOV     2,2,SNR
        JMP     @.STP

        LDA     0,C0
.IOL:   LDA     1,NM64          ; SET-UP FOR NEXT G-FILTER.
        LDA     2,LSTO
        STA     2,21
        LDA     2,NCL
        LDA     3,SCHK
        SUB     3,2,SZR
        JMP     .+5
        LDA     2,@27
         LDA     3,@2
        STA     3,SCHK
        JMP     .+2
        LDA     2,@25
        LDA     3,C1
        SUB     3,2
        STA     2,22
        STA     2,LSTO
PL:     LDA     2,@21           ;START PROCESSING NEXT G-FILTER.
        COM     0,0,SNR
        JMP     .+5
        LDA     3,BOL
        ADD     2,3
        STA     3,@22
        JMP     .+3
        MOVZL   2,3
        STA     3,BOL
        LDA     3,@21
        ADD     3,2
        STA     2,@22
.CL:    LDA     2,@21
        ADD     2,3
        STA     3,@22
        LDA     3,@21
        ADD     3,2
        STA     2,@22
        LDA     2,@21
```

```
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
```

```
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
```

```
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
ADD     3,2
STA     2,@22
LDA     2,@21
ADD     2,3
STA     3,@22
LDA     3,@21
```

```
        ADD     3,2
        STA     2,@22
        LDA     2,@21
        ADD     2,3
        STA     3,@22
        LDA     3,@21
        ADD     3,2
        STA     2,@22
        LDA     2,@21
        ADD     2,3
        STA     3,@22
        LDA     3,@21
        ADD     3,2
        STA     2,@22
        LDA     2,@21
        ADD     2,3
        STA     3,@22
        LDA     3,@21
        ADD     3,2
        STA     2,@22
        INC     1,1,SZR
        JMP     @CPL

.FP:    LDA     1,NM64
        LDA     2,@21
        ADD     2,3
        STA     3,@22
        INC     1,1,SZR
        JMP     .+2
        JMP     ELO
        LDA     3,@21
        ADD     3,2
        STA     2,@22
        LDA     2,@21
        ADD     2,3
        STA     3,@22
        LDA     3,@21
        ADD     3,2
        STA     2,@22
        JMP     .FP+1           ;THIS CAN BE MADE FASTER BY SPREADING IT OUT AS IN
                                ;THE OLDER VERSION OF BENCH.

ELO:    MOV     0,0,SNR
        JMP     .+5
        LDA     3,EOL
        ADD     3,2
        STA     2,@22
        JMP     .+6
        LDA     3,@21
        ADD     3,2
        STA     2,@22
```

```
        MOVZL   3,3
        STA     3,EOL

        LDA     2,NCL
        INC     2,2
        STA     2,NCL
        DSZ     FSTC
        JMP     @IOL

        LDA     2,SCNC
        MOV     2,2,SNR
        JMP     @.STP

.IOS:   LDA     1,NM64
        LDA     2,LSTO
        STA     2,21
        LDA     2,NCL
        LDA     3,SCHK
        SUB     3,2,SZR
        JMP     .+5
        LDA     2,@27
        LDA     3,@26
        STA     3,SCHK
        JMP     .+2
        LDA     2,@25
        LDA     3,C1
        SUB     3,2
        STA     2,22
        STA     2,LSTO
PSL:    LDA     2,@21           ;START PROCESSING NEXT SHIFTED (TO PREVENT
                                ;OVERFLOW) G-FILTER.
        COM     0,0,SNR
        JMP     .+5
        LDA     3,BOL
        ADDZR   2,3
        STA     3,@22
        JMP     .+2
        STA     2,BOL
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
.CSL:   LDA     2,@21
        ADDZR   2,3
        STA     3,@22
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
        LDA     2,@21
        ADDZR   2,3
        STA     3,@22
        LDA     3,@21
```

```
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
ADDZR          2,3
STA            3,@22
LDA            3,@21
ADDZR          3,2
STA            2,@22
LDA            2,@21
```

```
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
```

```
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
ADDZR    2,3
STA      3,@22
LDA      3,@21
ADDZR    3,2
STA      2,@22
LDA      2,@21
```

```
        ADDZR   2,3
        STA     3,@22
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
        LDA     2,@21
        ADDZR   2,3
        STA     3,@22
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
        LDA     2,@21
        ADDZR   2,3
        STA     3,@22
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
        INC     1,1,SZR
        JMP     @CPSL

.FPS:   LDA     1,HM64
        LDA     2,@21
        ADDZR   2,3
        STA     3,@22
        INC     1,1,SZR
        JMP     .+2
        JMP     ESLO
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
        LDA     2,@21
        ADDZR   2,3
        STA     3,@22
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
        JMP     .FPS+1      ;THIS CAN BE MADE FASTER BY SPREADING IT OUT AS IN
                            ;THE OLDER VERSION OF BENCH.

ESLO:   MOV     0,0,SNR
        JMP     .+5
        LDA     3,EOL
        ADDZR   3,2
        STA     2,@22
        JMP     .+5
        LDA     3,@21
        ADDZR   3,2
        STA     2,@22
        STA     3,EOL

        LDA     2,NCL
```

```
        INC     2,2
        STA     2,NCL

        LDA     2,THDC
        MOV     2,2,SNR
        JMP     @.STP

.IOE:   LDA     2,@23       ;START PROCESSING G-FILTERS WITH DELAYS GREATER
                            ;THAN ONE.
        STA     2,PAIRC
        LDA     0,@24
PEXL:   MOV     0,2
        LDA     3,NPEL
        SUB     2,3
        STA     3,NPPEL
        MOVZR   2,2
        STA     2,LE1C
        STA     2,LE2C
        LDA     1,LSTO
        STA     1,21
        ADD     2,1
        STA     1,20
        ADD     1,3
        ADD     2,3
        INC     3,3
        STA     3,31
        INC     1,1
        STA     1,30
        LDA     2,NCL
        LDA     3,SCHK
        SUB     3,2,SZR
        JMP     .+5
        LDA     2,@27
         LDA     3,@2
        STA     3,SCHK
        JMP     .+2
        LDA     2,@25
        LDA     3,C1
        SUB     3,2
        STA     2,22
        STA     2,LSTO
BEXL:   LDA     2,@20
        LDA     3,@30
        ADDZR   2,3
        STA     3,@22
        DSZ     LE1C
        JMP     BEXL
.CEXL:  LDA     2,@20
        LDA     3,@21
        ADDZR   2,3
        STA     3,@22
```

```
        DSZ     NPPEL
        JMP     .CEXL
EEXL:   LDA     2,@31
        LDA     3,@21
        ADDZR   2,3
        STA     3,@22
        DSZ     LE2C
        JMP     EEXL

        LDA     2,NCL
        INC     2,2
        STA     2,NCL
        DSZ     PAIRC
        JMP     PEXL
        DSZ     NPAIR
        JMP     .IOE

WRIT:   LDA     2,NSL           ;STORE OUTPUTS IN OUTPUT FILE.
        NEG     2,2
        LDA     1,NPEL
        MOVZL   1,1
        LDA     0,RLN
        MOVZL   0,0
WL:     .SYSTM
        .WRS 4
        JMP     @ER
        INC     2,2,SNR
        JMP     NEXT
        MOVZR   0,0
        LDA     3,C512
        ADD     3,0
        MOVZL   0,0
        JMP     WL

NEXT:   LDA     0,FIL           ; ANY MORE IMAGE LINES TO PROCESS ?.
        LDA     1,LIL
        SUB     0,1,SNR
        JMP     STOP
        INC     0,0
        STA     0,FIL
        LDA     0,OM64
        STA     0,M64
        STA     0,T64
        LDA     0,C2
        STA     0,NCL
        LDA     0,OFSTC
        STA     0,FSTC
        LDA     0,ONPAIR
        STA     0,NPAIR
        JMP     @ROL
```

```
STOP:   .SYSTM
        .RTN

.ER:    .SYSTM
        .ERTN

.END    STRT
```

APPENDIX 2

COMPUTER PROGRAMS ASSOCIATED WITH CHAPTER 3

```
C       ******************PROGRAM : BEDET******************

C CONVERTS THE OUTPUTS OF THE G-FILTERS PRODUCED BY "BENCH"
C INTO OUTPUTS OF THE H-FILTERS AND DETECTS ALL PEAKS IN THESE
C OUTPUTS THAT ARE ABOVE A GIVEN THRESHOLD.  AMONG THESE PEAKS,
C IT SELECTS PAIRS OF ADJACENT, OPPOSITE PEAKS WHOSE AMPLITUDES
C ARE WITHIN A GIVEN PERCENTAGE OF EACH OTHER; THIS OPERATION
C IS FOR EDGE DETECTION.



      COMMON/DUM1/NPEL,FATS(7),IBUFF(2560),ROW(512,4)
      COMMON/DUM2/VBEAK(512,4),IPBEAK(512,4),NBD(4)
      COMMON/DUM3/IFL(15),IPW(4),NDPR(25)
      COMMON/A/NED(4),IPE(50,4),IPBE(2,50,4),VBE(2,50,4),WH(3),IWH(3)
      DIMENSION VBEM(2,50),IPEM(2,50),IHE(50)
      EQUIVALENCE (VBEAK(1,1),VBEM(1,1)),(IPBEAK(1,1),IPEM(1,1)),
     *            (IBUFF(1),IHE(1))
    8 FORMAT(1X,I2)
    6 FORMAT(4X,I2,',   ',I4,',   ',F8.3,',    ',I4,',   ',F8.3)
    3 FORMAT(2X,I3,',   ',F8.3)
   15 FORMAT(S29)
      CALL OPEN(17,"BEDID",3,IER)

C READ NO. OF PELS PER LINE, NAME OF INPUT FILE (OUTPUT OF BENCH), NOS. OF
C FIRST AND LAST IMAGE LINE TO PROCESS, PEL WINDOWS FOR SELECTING EDGE PEAKS,
C PERCENTAGE FOR SELECTING EDGE PEAKS, PEL WINDOWS FOR PAIRING PEAKS IN
C ADJACENT FILTERS, THRESHOLD FOR RETAINING H-FILTER PEAKS, AND THRESHOLD
C FOR RETAINING EDGE PEAKS.
      READ(17)NPEL
      READ(17,15)IFL(1)
      READ(17)IFIL,ILIL,IPW,PCTHR,WH,THR,THR2
      DO 10 I=1,3
   10 IWH(I)=WH(I)
      TYPE 'BDID READ'

C OPEN FILE FOR STORING ALL DETECTED PEAKS.
      CALL DFILW("BDOD",IER)
      CALL CFILW("BDOD",2,IER)
      CALL OPEN(4,"BDOD",2,IER)
C OPEN FILE FOR STORING EDGE PEAKS.
      CALL DFILW("PAIREOD",IER)
      CALL CFILW("PAIREOD",2,IER)
      CALL OPEN(6,"PAIREOD",2,IER)

C INITIALIZE THE ADDRESSES FOR THE DETECTION PROCESS.
      CALL BBEDET(IFL,NPEL,IFIL,IBUFF(1),FATS(1),ROW(1,1),VBEAK(1,1),
     *            IPBEAK(1,1),NBD(1),VBE(1,1,1),IPBE(1,1,1),IPE(1,1),NED(1))

      DO 100 L=IFIL,ILIL
C STORE THE POSITIONS OF THE FAT-G'S AND GENERATE THE H-FILTER OUTPUTS.
```

```
        CALL BDS

C DETECT ALL H-FILTER PEAKS ABOVE THRESHOLD "THR2"
        CALL BDET(THR2)

C STORE POSITIONS AND VALUES OF DETECTED PEAKS AND THE 7 FAT-G'S IN FILE "BDOD"
        DO 30 J=1,4
        WRITE(4)NBD(J)
        IF(NBD(J).EQ.0)GO TO 30
        NBDT=NBD(J)
        WRITE(4,3)((IPBEAK(I,J),VBEAK(I,J)),I=1,NBDT)
   30   CONTINUE
        DO 140 I=1,7
  140   WRITE(4)FATS(I)

C DETECT EDGE PEAKS AMONG THE ABOVE PEAKS DETECTED.
        CALL EDET(THR,PCTHR,IPW(1))

C PAIR THE EDGE PEAKS.
        CALL PRES(NDPR(1),IWH(1))

C STORE THE DETECTED EDGE PEAKS IN FILE "PAIREOD".
        I=0
        DO 2000 M=1,25
        IF(NDPR(M).LE.1)GO TO 0100
        NP=NDPR(M)
        WRITE(6,8)NP
        DO 2000 K=1,NP
        I=I+1
        WRITE(6,6)IHE(I),IPEM(1,I),VBEM(1,I),IPEM(2,I),VBEM(2,I)
 2000   CONTINUE
 0100   IZ=0
        WRITE(6)IZ
        TYPE 'BEDET  ',L
  100   CONTINUE

        STOP
```

```
; **********************SUBROUTINE : BBEDET; CALLED BY BEDET*****************


; INITIALIZES THE ADDRESSES OF THE ARRAYS AND THE VALUES OF THE VARIBLES
; INVOLVED IN THE DETECTION PROCESS. ALSO, OPENS THE IO  CHANNEL TO THE
; OUTPUT OF "BENCH".



          .TITL               BBEDET
          .ENT                BBEDET
          .EXTD               .CPYL,.FRET
          .ENT                CHGS,NPEL,NPL4,NPL5,AIBUF,AFATS,AROW,AVBK,AIPBK,ANBD
          .ENT                AVBE,AIPBE,AIPE,ANED
ARG1=-167
ARG2=ARG1+1
ARG3=ARG2+1
ARG4=ARG3+1
ARG5=ARG4+1
ARG6=ARG5+1
ARG7=ARG6+1
ARG8=ARG7+1
ARG9=ARG8+1
ARG10=ARG9+1
ARG11=ARG10+1
ARG12=ARG11+1
ARG13=ARG12+1
          .ZREL
NPEL:     0
NPL4:     0
NPL5:     0
AIBUF:    0
AFATS:    0
AROW:     0
CHGS:     0
AVBK:     0
AIPBK:    0
ANBD:     0
AVBE:     0
AIPBE:    0
AIPE:     0
ANED:     0
          .NREL
          13.
BBEDET:   JSR                 @.CPYL
          LDA                 3,FSP
          .SYSTM                                ;GET FREE CHANNEL
          .GCHN
          JMP ER
          STA                 2,CHGS
          LDA                 0,ARG1,3
```

```
        MOVZL           0,0
        SUB             1,1
        .SYSTM
        .OPEN 77
        JMP             ER

        LDA             2,@ARG2,3
        STA             2,NPEL
        MOVZL           2,1
        MOVZL           1,1
        STA             1,NPL4
        ADD             2,1
        STA             1,NPL5

SEEK:   SUB             0,0
        LDA             1,NPL5
        MOVZL           1,1
        LDA             2,@ARG3,3
        LDA             3,CN1
        ADD             3,2
        MUL
        LDA             2,CHGS
        .SYSTM
        .SPOS 77
        JMP             ER

        LDA             3,FSP
        LDA             2,ARG4,3
        STA             2,AIBUF
        LDA             2,ARG5,3
        STA             2,AFATS
        LDA             2,ARG6,3
        STA             2,AROW
        LDA             2,ARG7,3
        STA             2,AVBK
        LDA             2,ARG8,3
        STA             2,AIPBK
        LDA             2,ARG9,3
        STA             2,ANBD
        LDA             2,ARG10,3
        STA             2,AVBE
        LDA             2,ARG11,3
        STA             2,AIPBE
        LDA             2,ARG12,3
        STA             2,AIPE
        LDA             2,ARG13,3
        STA             2,ANED
```

```
        JSR             @.FRET
CN1:    -1

ER:     .SYSTM
        .ERTN
.END
```

```
; ********************SUBROUTINE : BDS; CALLED BY BEDET*****************

; STORES THE POSITIONS AND VALUES OF THE FAT-G'S, AND GENERATES THE OUTPUTS
; OF THE H-FILTERS.


          .TITL          BDS
          .ENT           BDS
          .TXTM          1
          .EXTD          .CPYL,.FRET
          .EXTN          GLFX1,GXFL1
          .EXTD          CHGS,NPEL,NPL4,NPL5,AIBUF,AFATS,AROW

          .NREL
          0
BDS:      JSR            @.CPYL
          LDA            0,AIBUF
          MOVZL          0,0
          LDA            1,NPL5
          MOVZL          1,1
          LDA            2,CHGS
          .SYSTM
          .RDS 77                        ; READ OUTPUTS OF THE G-FILTERS.
          JMP ER
          JMP BFATS

ER:       .SYSTM
          .ERTN


BFATS:    LDA            2,C7            ; STORE POSITIONS AND VALUES OF THE 7 FAT-G'S.
          STA            2,TC7
          LDA            2,AIPF
          STA            2,21
          LDA            2,AIBUF
          LDA            3,NPL4
          ADD            2,3
          STA            3,SAS
          LDA            2,AFATS
          STA            2,RAFAT
GFATS:    LDA            2,@21
          ADD            2,3
          LDA            2,0,3
          STA            2,ITEMP
          GXFL1
          ITEMP
          LDA            2,RAFAT
          .FSRS          2
          DSZ            TC7
```

```
          JMP             .+2
          JMP             B31
          INC             2,2
          INC             2,2
          STA             2,RAFAT
          LDA             3,SAS
          JMP             GFATS

TC7:      0
C7:       7
RAFAT:    0
SAS:      0
AIPF:     .
IPF:      0
          21.
          42.
          63.
          84.
          105.
          127.
ITEMP:    0
AIFAC:    .
IFAC:     128.
          8.
          1
          1
          1
IFAC1:    0
IFAC2:    0
C2:       2
TC:       0
TNPEL:    0
C1:       1
AG:       0
C512:     512.
DROW:     0

;
; GENERATE THE OUTPUTS OF THE H-FILTERS.
;

B31:      LDA             3,C512
          LDA             1,NPEL
          SUB             1,3
          MOVZL           3,3
          STA             3,DROW
          LDA             2,AROW
          LDA             1,C1
          SUB             1,2
          SUB             3,2
          STA             2,24
```

```
        LDA     2,AIBUF
        SUB     1,2
        STA     2,AG
        LDA     2,AIFAC
        STA     2,21
        LDA     2,@21
        STA     2,IFAC2
        SUB     0,0

        LDA     2,C2
        STA     2,TC
D1J31:  LDA     3,DROW
        LDA     2,24
        ADD     3,2
        STA     2,24
        LDA     2,AG
        STA     2,23
        LDA     1,NPEL
        STA     1,TNPEL
        ADD     1,2
        STA     2,25
        STA     2,AG
        LDA     2,IFAC2
        STA     2,IFAC1
        LDA     2,@21
        STA     2,IFAC2
D1I31:  LDA     1,@23
        LDA     2,IFAC1
        MUL
        MOV     1,3
        LDA     1,@25
        LDA     2,IFAC2
        MUL
        SUB     1,3
        STA     3,@24
        STA     3,@24
        DSZ     TNPEL
        JMP     D1I31
        DSZ     TC
        JMP     D1J31

        LDA     2,C2
        STA     2,TC
D2J31:  LDA     3,DROW
        LDA     2,24
        ADD     3,2
        STA     2,24
        LDA     2,AG
        STA     2,23
        LDA     1,NPEL
        STA     1,TNPEL
```

```
            ADD             1,2
            STA             2,25
            STA             2,AG
D2I31:      LDA             3,@23
            LDA             1,@25
            SUB             1,3
            STA             3,@24
            STA             3,@24
            DSZ             TNPEL
            JMP             D2I31
            DSZ             TC
            JMP             D2J31
            JMP             B32
TNPL5:      0
AF128:      .+1
F128:       041200
            0
RAROW:      0

B32:        LDA             3,DROW
            LDA             2,AROW
            SUB             3,2
            STA             2,RAROW
            LDA             1,C1
            SUB             1,2
            STA             2,23
            LDA             2,AF128
            .FLDS           2
            .FMFT
            LDA             2,C2
            MOVZL           2,2
            STA             2,TC

DOJ32:      LDA             3,DROW
            LDA             2,RAROW
            ADD             3,2
            STA             2,RAROW
            LDA             2,23
            ADD             3,2
            STA             2,23
            LDA             2,NPEL
            STA             2,TNPEL

DOI32:      LDA             2,@23
            STA             2,@23       ;THIS IS A DUMMY INSTRUCTION TO INCREMENT @23.
            STA             2,ITEMP
            GXFL1
            ITEMP
            .FDTS
            LDA             2,RAROW
            .FSRS           2
```

```
        INC     2,2
        INC     2,2
        STA     2,RAROW
        DSZ     TMPEL
        JMP     DOI32
        DSZ     TC
        JMP     DOJ32

BACK:   JSR     @.FRET

.END
```

```
; ********************SUBROUTINE : BDET; CALLED BY BEDET*****************


; DETECTS ALL PEAKS IN THE OUTPUTS OF THE H-FILTERS WHOSE ABSOLUTE VALUE
; IS GREATER THAN "THR2".

          .TITL               BDET
          .ENT                BDET
          .EXTD               .CPYL,.FRET
          .EXTD               NPEL,AROW,AVBK,AIPBK,ANBD
          .ENT                RANBD,RAVBK,FGLZ,FGTZ,C4,TC4,TC,SGN
ARG1=-167
          .ZREL
ATHR2:    0
FGLZ:     02400
FGTZ:     02000
RANBD:    0
RAVBK:    0
RAROW:    0
AJROW:    0
TC:       0
C4:       4
TC4:      0
SGN:      0
DORS:     .DORS
L030:     .L030
BACK:     .BACK

          .NREL
          1
BDET:     JSR                 @.CPYL
          LDA                 3,FSP
          LDA                 2,ARG1,3
          STA                 2,ATHR2
          LDA                 2,AROW
          STA                 2,RAROW
          STA                 2,AJROW
          LDA                 2,ANBD
          STA                 2,RANBD
          LDA                 2,AVBK
          STA                 2,RAVBK
          LDA                 2,AIPBK
          SUB                 1,1
          INC                 1,1             ;LDA          1,C1
          SUB                 1,2
          STA                 2,20
          LDA                 2,C4
          STA                 2,TC4
          JMP                 .+2

TNPEL:    0
```

```
.DORS:  SUB     0,0
        STA     0,@RANBD
        LDA     2,NPEL
        STA     2,TNPEL
        DSZ     TNPEL
        LDA     1,RAROW
        MOV     1,2
        LDA     0,FGLZ
D020:   INC     2,2             ; GET THE FIRST SLOPE YOU ENCOUNTER STARTING AT
                                ; THE LEFT END OF THE FILTER OUTPUT.
        INC     2,2
        .FLDS   2
        .FSS    1
        .FRST   3
        AND     0,3,SZR
        JMP     XEB
        MOV     2,1
        DSZ     TNPEL
        JMP     D020
        JMP     NJROW

XEB:    STA     2,RAROW
        STA     3,SGN           ;GET THE SIGN OF THIS FIRST SLOPE.
        LDA     2,AJROW
        SUB     2,1
        MOVZR   1,1             ;=IS-1
        MOVZR   1,1             ;=(IS-1)/2
        INC     1,0             ;=(IS-1)/2+1
        STA     0,@20
        MOVZL   1,1
        ADD     2,1
        .FCLR
        .FAS    1
        .FRST   2
        LDA     0,FGLZ
        AND     0,2             ;GET THE SIGN OF THE MID-FLAT PEAK.
        AND     2,3,SZR
        JMP     XPK             ;IF THE SIGNS ARE EQUAL,REMOVE PEAK POSITION FROM 20.
        .FABS
        LDA     2,ATHR2
        .FSS    2
        .FRST   2
        LDA     0,FGTZ
        AND     0,2,SNR
        JMP     XPK             ;IF ABS(ROW(PEAK)) < THR2, REMOVE POSITION FROM 20.
        .FLDS   1
        LDA     2,RAVBK
        .FSRS   2
        INC     2,2
        INC     2,2
        STA     2,RAVBK
```

```
        ISZ     @RANBD
        JMP     L30
XPK:    DSZ     20

L30:    LDA     1,RAROW                 ; PROCEED WITH THE REST OF THE FILTER OUTPUT.
        INC     1,2
        INC     2,2
        STA     2,RAROW
        .FLDS   2
        .FSS    1
        .FRST   3
        LDA     0,FGLZ
        AND     0,3,SNR
        JMP     FLATP
        LDA     2,SGN
        AND#    2,3,SNR
        JMP     L31
        JMP     @L030

L31:    STA     3,SGN
        .FCLR
        .FAS    1
        .FRST   2
        AND     0,2
        AND     2,3,SZR
        JMP     @L030
        .FABS
        LDA     2,ATHR2
        .FSS    2
        .FRST   2
        LDA     0,FGTZ
        AND     0,2,SNR
        JMP     @L030

        LDA     2,RAROW
        NEG     2,1
        LDA     3,AJROW
        SUBOR   3,2
        STA     2,@20
        INC     1,1
        INC     1,1
        NEG     1,1
        .FLDS   1
        LDA     2,RAVBX
        .FSRS   2
        INC     2,2
        INC     2,2
        STA     2,RAVBX
        ISZ     @RANBD

.L030:  LDA     2,RAROW
```

```
            MOV         2,3
            LDA         1,AJROW
            SUBOR       1,2
            LDA         1,NPEL
            NEG         1,1
            INC         1,1
            NEG         1,1
            SUB         1,2,SZR
            JMP         L30

            MOV         3,1
            .FCLR
            .FAS        1
            .FRST       3
            LDA         2,SGN
            AND         2,3,SNR
            JMP         NJROW
            .FABS
            LDA         2,ATHR2
            .FSS        2
            .FRST       2
            LDA         0,FGTZ
            AND         0,2,SNR
            JMP         NJROW
            LDA         2,NPEL
            STA         2,@20
            .FLDS       1
            LDA         2,RAVBK
            .FSRS       2
            INC         2,2
            INC         2,2
            STA         2,RAVBK
            ISZ         @RANBD
            JMP         NJROW


C512:       512.
NJROW:      DSZ         TC4             ; GET NEXT FILTER.
            JMP         .+2
            JMP         @BACK
            LDA         1,C512
            LDA         2,C4
            LDA         3,TC4
            SUB         3,2
            SUB         0,0
            MUL
            LDA         2,AIPBK
            ADD         1,2
            NEG         2,2
            INC         2,2
            NEG         2,2
```

```
          STA     2,20
          MOVZL   1,1
          LDA     2,AROW
          ADD     1,2
          STA     2,RAROW
          STA     2,AJROW
          LDA     2,AVBK
          ADD     1,2
          STA     2,RAVBK
          ISZ     RANBD
          JMP     @DORS

ICSM1:    0
FLATP:    LDA     0,RAROW                 ; SUBROUTINE FOR HANDLING FLAT PEAKS.
          MOV     0,1
          MOV     0,2
          LDA     3,AJROW
          SUBOR   3,0
          STA     0,ICSM1
          INC     0,0
          LDA     3,NPEL
          SUB     0,3,SNR
          JMP     @L030
          STA     3,TC
          LDA     0,FGLZ
DOF10:    INC     2,2
          INC     2,2
          .FLDS   2
          .FSS    1
          .FRST   3
          AND#    0,3,SZR
          JMP     LF010
          MOV     2,1
          DSZ     TC
          JMP     DOF10

          LDA     2,NPEL
          LDA     1,ICSM1
          SUB     1,2
          MOVZR   2,2
          ADD     1,2
          STA     2,@20
          LDA     1,AJROW
          NEG     2,2             ;
          INC     2,2             ;LDA        0,C1
          NEG     2,2             ;
          MOVZL   2,2
          ADD     2,1
          .FCLR
          .FAS    1
          .FRST   3
```

```
        LDA     0,FGLZ
        AND     0,3
        LDA     2,SGN
        AND     2,3,SNR
        JMP     GOBE
        .FABS
        LDA     2,ATHR2
        .FSS    2
        .FRST   2
        LDA     0,FGTZ
        AND     0,2,SNR
        JMP     GOBE
        .FLDS   1
        LDA     2,RAVBK
        .FSRS   2
        INC     2,2
        INC     2,2
        STA     2,RAVBK
        ISZ     @RANBD
        JMP     .+2
GOBE:   DSZ     20
        JMP     NJROW

LF010:  STA     2,RAROW
        LDA     2,AJROW
        SUB     2,1
        MOVZR   1,1
        INC     1,1             ;=ICE
        LDA     2,ICSM1
        SUB     2,1
        MOVZR   1,1
        ADD     2,1
        STA     1,@20

        LDA     2,SGN
        LDA     0,FGLZ
        AND     0,3
        STA     3,SGN
        AND     2,3,SZR         ;IF THE SLOPE SIGNS BEFORE AND AFTER THE FLAT PEAK ARE
        JMP     FRTE            ;EQUAL, DISREGARD THE FLAT PEAK.

        NEG     1,1
        INC     1,1
        NEG     1,1
        MOVZL   1,1
        LDA     2,AJROW
        ADD     2,1
        .FCLR
        .FAS    1
        .FRST   3
        LDA     0,FGLZ
```

```
        AND             0,3
        LDA             2,SGN
        AND             2,3,SZR         ;IF THE SIGNS OF THE FLAT PEAK AND THE SLOPE AFTER IT
        JMP             FRTE            ;ARE EQUAL, DISREGARD THE FLAT PEAK.

        .FABS
        LDA             2,ATHR2
        .FSS            2
        .FRST           3
        LDA             0,FGTZ
        AND             0,3,SNR
        JMP             FRTE            ;IF ABS(ROW(PEAK)) ( THR2, DIREGARD THE PEAK.

        .FLDS           1
        LDA             2,RAVBK
        .FSRS           2
        INC             2,2
        INC             2,2
        STA             2,RAVBK
        ISZ             @RANBD
        JMP             .+2
FRTE:   DSZ             20
        JMP             @L030

.BACK:  JSR             @.FRET
.END
```

```
; ********************SUBROUTINE : EDET; CALLED BY BEDET*****************

; DETECTS EDGE PEAKS SATISFYING THE "THR", THE "PCTHR", AND THE "IPW"
; PREDICATES.

          .TITL     EDET
          .ENT      EDET
          .EXTD     .CPYL,.FRET
          .EXTD     AVBK,AIPBK,ANBD,AVBE,AIPBE,AIPE,ANED
          .EXTD     C4,TC4,RANBD,RAVBK,FGLZ,FGTZ
          .ENT      RANED,RAVBE
ARG1=-167
ARG2=ARG1+1
ARG3=ARG2+1
          .ZREL
ATHR:     0
APCTH:    0
RANED:    0
RAVBE:    0
NBD1:     0
D0101:    .D0101

          .NREL
          3
EDET:     JSR       @.CPYL                    ; GET THE VALUES OF THE THREE PREDICATES.
          LDA       3,FSP
          LDA       2,ARG1,3
          STA       2,ATHR
          LDA       2,ARG2,3
          STA       2,APCTH
          LDA       2,ARG3,3
          NEG       2,2
          INC       2,2
          NEG       2,2
          STA       2,21


          LDA       2,ANBD                    ; INITIALIZE ARRAY ADDRESSES.
          STA       2,RANBD
          LDA       2,AVBK
          STA       2,RAVBK
          LDA       2,ANED
          STA       2,RANED
          LDA       2,AVBE
          STA       2,RAVBE
          LDA       2,AIPBK
          SUB       1,1
          INC       1,1             ;LDA      1,C1
```

```
            SUB             1,2
            STA             2,20
            LDA             2,AIPBE
            SUB             1,2
            STA             2,22
            LDA             2,AIPE
            SUB             1,2
            STA             2,24
            LDA             2,C4
            STA             2,TC4
            JMP             .+2

IPWJ:       0
.D0101:     SUB             0,0                     ; START CHECKING THE PEAKS FOR THE THREE
                                                    ; PREDICATES.
            STA             0,@RANED
            LDA             2,@21
            STA             2,IPWJ
            LDA             2,@RANBD
            SUB             1,1
            INC             1,1             ;LDA     1,C1
            INC             1,3
            SUBZ#           3,2,SNC
            JMP             NEXTJ
            SUB             1,2
            STA             2,NBD1
            LDA             1,@20
            LDA             0,RAVBK
            JMP             .+4
IPBK1:      0
SIPBK:      0
SAVBK:      0
D0102:      LDA             2,@20
            STA             2,SIPBK
            SUB             1,2
            STA             1,IPBK1
            INC             0,1
            INC             1,1
            LDA             3,IPWJ
            SUBZ            2,3,SNC
            JMP             NXTBK
            .FCLR
            .FAS            0
            .FRST           2
            .FMFT
            .FCLR
            .FAS            1
            .FRST           3
            AND             2,3
            LDA             2,FGLZ
            AND             2,3,SZR
```

```
        JMP     NXTBK
        LDA     3,FGTZ
        .FABS
        LDA     2,ATHR
        .FSS    2
        .FRST   2
        AND     3,2,SZR
        JMP     .+10
        .FMTF
        .FABS
        LDA     2,ATHR
        .FSS    2
        .FRST   2
        AND     3,2,SNR
        JMP     NXTBK
        .FLDS   0
        .FSS    1
        .FMFT
        .FLDS   0
        .FAS    1
        .FDTS
        .FABS
        LDA     2,APCTH
        .FSS    2
        .FNEG
        .FRST   2
        AND     3,2,SNR
        JMP     NXTBK
        ISZ     @RANED
        .FLDS   0
        LDA     2,RAVBE
        .FSRS   2
        .FLDS   1
        INC     2,2
        INC     2,2
        .FSRS   2
        INC     2,2
        INC     2,2
        STA     2,RAVBE
        LDA     2,IPBK1
        STA     2,@22
        LDA     3,SIPBK
        STA     3,@22
        SUBOR   2,3
        ADD     2,3
        STA     3,@24
NXTBK:  MOV     1,0                     ; GET NEXT PEAK.
        LDA     1,SIPBK
        DSZ     NBD1
        JMP     DO102
        JMP     .+3
```

```
C512:   512.
C50:    50.
NEXTJ:  DSZ     TC4                     ; GET THE PEAKS OF THE NEXT H-FILTER.
        JMP     .+2
        JMP     BACK
        LDA     1,C512
        LDA     3,C4
        LDA     2,TC4
        SUB     2,3
        MOV     3,2
        SUB     0,0
        MUL
        LDA     2,AIPBK
        ADD     1,2
        NEG     2,2
        INC     2,2
        NEG     2,2
        STA     2,20
        MOVZL   1,1
        LDA     2,AVBK
        ADD     1,2
        STA     2,RAVBK
        LDA     1,C50
        MOV     3,2
        MUL
        LDA     2,AIPE
        ADD     1,2
        NEG     2,2
        INC     2,2
        NEG     2,2
        STA     2,24
        MOVZL   1,1
        LDA     2,AIPBE
        ADD     1,2
        NEG     2,2
        INC     2,2
        NEG     2,2
        STA     2,22
        MOVZL   1,1
        LDA     2,AVBE
        ADD     1,2
        STA     2,RAVBE
        ISZ     RANED
        ISZ     RANBD
        JMP     @D0101

BACK:   JSR     @.FRET
.END
```

```
; **********************SUBROUTINE : PRES; CALLED BY BEDET*****************

; PAIRS THE EDGE PEAKS THAT SATISFY THE "WH" CRITERIA.

          .TITL               PRES
          .ENT                PRES
          .EXTD               .CPYL,.FRET
          .EXTD               AVBE,AIPBE,AIPE,ANED,AVBK,AIPBK,AIBUF
          .EXTD               C4,TC,FGLZ,FGTZ,RAVBE,RANED,SGN
ARG1=-167
ARG2=     ARG1+1
          .ZREL
RANDP:    0
AJVBE:    0
AJIPB:    0
AJIPE:    0
RAVBM:    0
SJ1VB:    0
AJ1VB:    0
RAJ1V:    0
SJ1PE:    0
AJ1PE:    0
SJ1PB:    0
AJ1PB:    0
SJ:       0
J1:       0
C3:       3
SNED:     0
AIWHJ:    0
TTC:      0
D0202:    .D202
BCK:      GOBCK
          .NREL
          2
PRES:     JSR                 @.CPYL
          LDA                 3,FSP
          LDA                 2,ARG1,3
          STA                 2,RANDP
          SUB                 1,1
          INC                 1,1
          SUB                 1,2
          STA                 2,20
          LDA                 2,C25
          STA                 2,TC
          SUB                 0,0
          JMP                 .+2
C25:      25.
ZNDPR:    STA                 0,@20         ; INITIALIZE THE ADDRESSES OF THE ARRAYS.
          DSZ                 TC
          JMP                 ZNDPR
```

```
        LDA     2,ARG2,3
        SUB     1,2
        STA     2,AIWHJ
        LDA     2,AVBE
        STA     2,AJVBE
        STA     2,SJ1VB
        LDA     2,AIPBE
        SUB     1,2
        STA     2,AJIPB
        STA     2,SJ1PB
        LDA     2,AIPE
        SUB     1,2
        STA     2,AJIPE
        STA     2,SJ1PE
        LDA     2,AVBX
        STA     2,RAVBM
        LDA     2,AIPBX
        SUB     1,2
        STA     2,22
        LDA     2,AIBUF
        SUB     1,2
        STA     2,26
        STA     1,SJ
        LDA     2,ANED
        STA     2,SNED

.D202:  LDA     2,AJVBE
        STA     2,RAVBE
        LDA     2,AJIPB
        STA     2,27
        LDA     2,AJIPE
        STA     2,23
        LDA     2,@SNED
        MOV     2,2,SNR
        JMP     NEXTJ
        STA     2,TC

D2021:  LDA     2,AIWHJ         ; START CHECKING THE PEAKS FOR THE
                                ; PAIRING CRITERION.
        STA     2,21
        LDA     2,SNED
        STA     2,RANED
        ISZ     RANED
        LDA     2,@RANED
        MOV     2,2,SNR
        JMP     NEXTJ
        SUB     1,1
        INC     1,1
        STA     1,@RANDP
        LDA     2,SJ
        INC     2,2
```

```
        STA     2,J1
        LDA     2,SJ1VB
        STA     2,AJ1VB
        LDA     2,SJ1PB
        STA     2,AJ1PB
        LDA     2,SJ1PE
        STA     2,AJ1PE
        LDA     1,RAVBE
        .FCLR
        .FAS    1
        .FRST   3
        LDA     0,FGLZ
        AND     0,3
        STA     3,SGN
        LDA     2,RAVBM
        .FSRS   2
        INC     1,1
        INC     1,1
        .FLDS   1
        INC     2,2
        INC     2,2
        .FSRS   2
        INC     1,1
        INC     1,1
        STA     1,RAVBE
        SUB     0,0             ; NOTE THAT THE UPDATE OF RAVBM IS LEFT TO PRE ONLY
        INC     0,0             ; IF IT CONFIRMS A MATCH.
        STA     0,PI
        LDA     2,@27
        STA     2,IPBE1
        LDA     2,@27
        STA     2,IPBE2         ; SIMILARLY, STORING IPBE1 AND IPBE2 IS LEFT TO PRE.
        LDA     0,@23
        JSR     PRE
        LDA     2,IFLAG
        MOV     2,2,SNR
        JMP     NEXTI
        LDA     3,C3
        LDA     2,SJ
        SUB     2,3,SNR
        JMP     NEXTI

CM1:    ISZ     J1
        SUB     0,0
        STA     0,PI
        ISZ     RANED
        LDA     2,@RANED
        MOV     2,2,SNR
        JMP     NEXTI
        LDA     1,RAVBM
```

```
        LDA     2,C4
        SUB     2,1
        .FCLR
        .FAS    1
        .FRST   3
        LDA     0,FGLZ
        AND     0,3
        STA     3,SGN
        LDA     0,IPET
        JSR     PRE
        LDA     2,IFLAG
        MOV     2,2,SNR
        JMP     NEXTI
        SUB     3,3
        INC     3,3
        INC     3,3
        LDA     2,SJ
        SUB     2,3,SNR
        JMP     NEXTI

CM2:    ISZ     J1
        ISZ     RANED
        LDA     2,@RANED
        MOV     2,2,SNR
        JMP     NEXTI
        LDA     1,RAVBM
        LDA     2,C4
        SUB     2,1
        .FCLR
        .FAS    1
        .FRST   3
        LDA     0,FGLZ
        AND     0,3
        STA     3,SGN
        LDA     0,IPET
        JSR     PRE

NEXTI:  LDA     2,@RANDP        ; GET NEXT PEAK.
        NEG     2,2
        INC     2,2,SZR
        ISZ     RANDP
        DSZ     TC
        JMP     D2021

NEXTJ:  LDA     3,C3                    ; GET PEAKS OF NEXT H-FILTER.
        LDA     2,SJ
        SUB     2,3,SNR
        JMP     @BCK
        ISZ     SJ
```

```
        ISZ     AIWHJ
        ISZ     SNED
        LDA     3,C200
        LDA     2,AJVBE
        ADD     3,2
        STA     2,AJVBE
        STA     2,SJ1VB
        MOVZR   3,3
        LDA     2,AJIPB
        ADD     3,2
        STA     2,AJIPB
        STA     2,SJ1PB
        MOVZR   3,3
        LDA     2,AJIPE
        ADD     3,2
        STA     2,AJIPE
        STA     2,SJ1PE
        JMP     @DO202

PI:     0
C200:   200.
IPET:   0
IFLAG:  0
IPBE1:  0
IPBE2:  0
IWHJ:   0
PRE:    STA     3,PRET          ; PERFORM THE ACTUAL PAIRING FOR THE PEAKS
                                ; THAT MEET THE CRITERIA.
                                ; THE POSITION OF THE EDGE TO PAIRE IS IN AC0.
        SUB     2,2
        STA     2,IFLAG
        LDA     2,@21
        STA     2,IWHJ
        LDA     2,AJ1VB
        LDA     3,C200
        ADD     3,2
        STA     2,AJ1VB
        STA     2,RAJ1V
        MOVZR   3,3
        LDA     2,AJ1PB
        ADD     3,2
        STA     2,AJ1PB
        MOVZR   3,3
        LDA     2,AJ1PE
        ADD     3,2
        STA     2,AJ1PE
        STA     2,25
```

```
        LDA     1,SGN
        LDA     2,@RANED
        STA     2,TTC
PRD20:  LDA     2,@25
        STA     2,IPET
        SUBZ    0,2,SNC
        NEG     2,2
        LDA     3,IWHJ
        SUBZ    2,3,SNC
        JMP     PNXTI
        LDA     2,RAJ1V
        .FCLR
        .FAS    2
        .FRST   3
        AND     1,3,SNR
        JMP     PNXTI
        JMP     PRM
PNXTI:  LDA     2,RAJ1V
        INC     2,2
        INC     2,2
        INC     2,2
        INC     2,2
        STA     2,RAJ1V
        DSZ     TTC
        JMP     PRD20
        JMP     @PRET

PRET:   0

PRM:    SUB     2,2             ; SEE IF MULTIPLE PAIRINGS CAN BE MADE.
        INC     2,2
        STA     2,IFLAG

        LDA     2,PI
        MOV     2,2,SNR
        JMP     PSTJ1
        LDA     2,SJ
        STA     2,@26
        LDA     2,IPBE1
        STA     2,@22
        LDA     2,IPBE2
        STA     2,@22
        LDA     2,RAVBM
        LDA     1,C4
        ADD     1,2
        STA     2,RAVBM

PSTJ1:  LDA     2,J1
        STA     2,@26
        LDA     3,@RANED
        LDA     2,TTC
```

```
        SUB     2,3
        MOVZL   3,3
        LDA     2,AJ1PB
        INC     2,2
        ADD     3,2
        LDA     3,0,2
        STA     3,@22
        INC     2,2
        STA     2,20
        LDA     3,0,2
        STA     3,@22
        LDA     1,RAJ1V
        .FLDS   1
        LDA     2,RAVBM
        .FSRS   2
        INC     1,1
        INC     1,1
        .FLDS   1
        INC     2,2
        INC     2,2
        .FSRS   2
        INC     2,2
        INC     2,2
        STA     2,RAVBM
        ISZ     @RANDP
        DSZ     @RANED
        JMP     .+1

        LDA     2,TTC
        SUB     0,0
        INC     0,0
        SUB     0,2,SNR
        JMP     @PRET

        STA     2,TTC
        NEG     1,2
        INC     2,2
        INC     2,2
        INC     2,2
        INC     2,2
        NEG     2,2
        LDA     3,25
        SUB     0,3
        STA     3,24

PSHFT:  LDA     3,@25           ; ADJUST ARRAY POSITIONS ACCORDINGLY.
        STA     3,@24
        LDA     0,@20
        LDA     3,20
        STA     0,-2,3
        LDA     0,@20
```

```
        LDA     3,20
        STA     0,-2,3
        INC     1,1
        INC     1,1
        .FLDS   1
        INC     2,2
        INC     2,2
        .FSRS   2
        INC     1,1
        INC     1,1
        .FLDS   1
        INC     2,2
        INC     2,2
        .FSRS   2
        DSZ     TTC
        JMP     PSHFT
        JMP     @PRET

GOBCK:  JSR     @.FRET
.END
```

```
C ******************************PROGRAM : ES************************

C ESTIMATES EDGES FROM THE OUTPUT OF "BEDET" : "PAIREOD".  PERFORMS CORRECTIONS
C AMONG EDGS, AS WELL AS CORRECTING THE GAUSSIAN PEAKS FOR EDGES.


      COMMON/AE/NGE,NPE(20),IPBE(2,50),VBE(2,50),IHP(50),ISFP(2,50)
      COMMON/FM/FME(20),IOFME(20)
      COMMON/ER/IER(3584)
      COMMON/FR/IFR(896)
      COMMON/ED/IED(4)
      COMMON/NP/NPEL,NPL4
      COMMON/DUM/NB(4),IPB(100,4),VB(100,4),IPF(7),FATS(7)
    5 FORMAT(2X,I3,',  ',F8.3)
   11 FORMAT(A1)
    4 FORMAT(2X,F6.1,',  ',I3,',  ',F7.3',  ',F7.3)

C OPEN OUTPUT FILE FOR ESTIMATED EDGES.
      CALL DFILW("ESOD",MER)
      CALL CFILW("ESOD",2,MER)
      CALL FOPEN(18,'ESOD')
C OPEN OUTPUT FILE FOR CORRECTED GAUSSIAN PEAKS.
      CALL DFILW("CBOD",MER)
      CALL CFILW("CBOD",2,MER)
      CALL FOPEN(16,'CBOD')
C OPEN TABLE LOOK-UP FILE FOR CORRECTING H-FILTER PEAKS FOR  EDGES.
      CALL FOPEN(13,'HERS')
      READ BINARY(13)IER
      CALL FCLOS(13)
      TYPE 'IER READ'
      CALL FOPEN(15,"ESID")
C READ DEFAULT EDGES.
      READ(15)IED
      CALL FCLOS(15)
      CALL FOPEN(15,'BEDID')
      READ(15)NPEL
      NPL4=4*NPEL
      READ(15,11)IDUM
      READ(15)IFIL,ILIL
      READ(15)IDUM
      READ(15)PCTHR
      READ(15)DUMMY
      READ(15)BETH,THR2
C OPEN TABLE LOOK-UP FILE FOR CORRECTING THE FAT-G'S FOR EDGES.
      CALL FOPEN(9,'FERS')
      READ BINARY(9)IFR
      TYPE 'IFR READ'

      IPF(1)=1
```

```
      IPF(2)=22
      IPF(3)=43
      IPF(4)=64
      IPF(5)=85
      IPF(6)=106
      IPF(7)=128
      CALL FOPEN(13,"BDOD")
      CALL FOPEN(19,"PAIREOD")

      DO 100 L=IFIL,ILIL
      DO 20 J=1,4
      READ(13)NB(J)
      IF(NB(J).EQ.0)GO TO 20
      N=NB(J)
      DO 10 I=1,N
   10 READ(13)IPB(I,J),VB(I,J)
   20 CONTINUE
      READ(13)FATS
      NGE=0
      ID=0
  010 READ(19)NMP
      IF(NMP.EQ.0)GO TO 041
      NGE=NGE+1
      NPE(NGE)=NMP
      DO 40 I=1,NMP
      ID=ID+1
      READ(19)IHP(ID),IPBE(1,ID),VBE(1,ID),IPBE(2,ID),VBE(2,ID)
      ISFP(1,ID)=VBE(1,ID)/ABS(VBE(1,ID))
   40 ISFP(2,ID)=VBE(2,ID)/ABS(VBE(2,ID))
      GO TO 010
  041 IF(NGE.EQ.0)GO TO 50
      IBOE=0
      ICE=1

C NGE = NO. OF ALREADY ESTIMATED ES + NO. OF GROUPS TO BE PROCESSED

C RESORT EDGES TO START WITH THE LARGEST, NARROWEST PEAK.
  050 CALL ERST(IBOE,ICE)

C ESTIMATE AN EDGE FROM THE FILTER PEAKS.
 0050 CALL EEST(IBOE,ICE,PE,IE,A,AMF,MER)
      IBOE=IBOE+NPE(ICE)
      IF(MER.EQ.2)GO TO 0070
C STORE THE ESTIMATED EDGE.
      WRITE(18,4)PE,IE,A,AMF
      IF(ICE.EQ.NGE)GO TO 030

C CORRECT THE REST OF THE EDGE PEAKS FOR THE JUST ESTIMATED EDGE.
      CALL AECORR(IBOE,ICE,PE,IE,A,AMF)
```

```
C REMOVE THE EDGE PEAKS THAT NO LONGER SATISFY THE CRITERIA AFTER CORRECTION.
      CALL ECLN(BETH,IBOE,ICE,PCTHR)

  030 DO 30 J=1,4
      IF(NB(J).EQ.0)GO TO 30
      N=NB(J)
      DO 31 I=1,N
C CORRECT ALL H-FILTER PEAKS FOR THE JUST ESTIMATED EDGE.
      CALL ESCORR(J,IPB(I,J),VB(I,J),IE,PE,A,AMF)
   31 CONTINUE
   30 CONTINUE

      DO 70 I=1,7
C CORRECT ALL FAT-G'S FOR THE JUST ESTIMATED EDGE.
      CALL EFCORR(FATS(I),IPF(I),IE,PE,A,AMF)
   70 CONTINUE
 0070 IF(NGE.EQ.ICE)GO TO 0140
      ICE=ICE+1
      GO TO 050


C RETAIN ONLY THE H-FILTER PEAKS THAT ARE STILL ABOVE THE DETECTION THRESHOLD.
 0140 DO 140 J=1,4
      IF(NB(J).EQ.0)GO TO 140
      N=NB(J)
      IU=0
      DO 141 I=1,N
      IF(ABS(VB(I,J)).GE.THR2)GO TO 0141
      IU=IU+1
      NB(J)=NB(J)-1
      GO TO 141
 0141 IPB(I-IU,J)=IPB(I,J)
      VB(I-IU,J)=VB(I,J)
  141 CONTINUE
  140 CONTINUE

   50 IZ=0
      FZ=0
      WRITE(18,4)FZ,IZ,FZ,FZ

      DO 150 J=1,4
      WRITE(16)NB(J)
      IF(NB(J).EQ.0)GO TO 150
      N=NB(J)
      DO 151 I=1,N
  151 WRITE(16,5)IPB(I,J),VB(I,J)
  150 CONTINUE
      DO 160 I=1,7
  160 WRITE(16)FATS(I)

      TYPE 'ES ',L
```

```
100   CONTINUE

      STOP
      END
```

```
C **********************SUBROUTINE : ERST; CALLED BY BEDET*****************

C RESORTS EDGE PEAKS FORM THE LARGEST, NARROWEST TO THE WEAKEST, BROADEST.
C ALSO, FAVORS MULTIPLE OVER SINGLE EDGE-PEAK PAIRES.

      SUBROUTINE ERST(IBOE,ICE)
      COMMON/AE/NRGE,NPE(20),IPBE(2,50),VBE(2,50),IHP(50),ISFP(2,50)
      COMMON/FM/FME(20),IOFME(20)
      COMMON/CLNDUM/IH1(4),IP1(2,4),VB1(2,4),ISF1(2,4)


      ID=IBOE
      DO 20 N=ICE,NRGE
      NM=NPE(N)
      IOFME(N)=1
      IDT=ID+IOFME(N)
      FME(N)=ABS((VBE(1,IDT)+VBE(2,IDT))/(VBE(1,IDT)-VBE(2,IDT)))
      IF(NM.EQ.1) GO TO 020
      IDT=IDT+1
      FME(N)=(FME(N)+ABS((VBE(1,IDT)+VBE(2,IDT))/(VBE(1,IDT)-VBE(2,IDT))))/2.
020   ID=ID+NM
 20   CONTINUE

      I1I=ICE
      IHP1=IHP(IBOE+1)
      ID=IBOE+NPE(ICE)+1
      ICE1=ICE+1
      DO 30 I=ICE1,NRGE
      IF((IHP1.LT.IHP(ID)).OR.((IHP1.EQ.IHP(ID))
     *                         .AND.(FME(I).GE.FME(I1I))))GO TO 030
      ID1=ID
      IHP1=IHP(ID)
      I1I=I
030   ID=ID+NPE(I)
 30   CONTINUE

      IF(I1I.EQ.ICE)GO TO 100
      FM1=FME(I1I)
      IOF1=IOFME(I1I)
      NMP=NPE(I1I)
      ID1=ID1-1
      DO 50 I=1,NMP
      IH1(I)=IHP(ID1+I)
      IP1(1,I)=IPBE(1,ID1+I)
      IP1(2,I)=IPBE(2,ID1+I)
      VB1(1,I)=VBE(1,ID1+I)
      VB1(2,I)=VBE(2,ID1+I)
      ISF1(1,I)=ISFP(1,ID1+I)
 50   ISF1(2,I)=ISFP(2,ID1+I)
```

```
      IBOE1=IBOE+1
      DO 60 I=IBOE1,ID1
      IR=ID1-I+IBOE1
      IHP(IR+NMP)=IHP(IR)
      IPBE(1,IR+NMP)=IPBE(1,IR)
      IPBE(2,IR+NMP)=IPBE(2,IR)
      VBE(1,IR+NMP)=VBE(1,IR)
      VBE(2,IR+NMP)=VBE(2,IR)
      ISFP(1,IR+NMP)=ISFP(1,IR)
60    ISFP(2,IR+NMP)=ISFP(2,IR)

      DO 70 I=1,NMP
      IHP(IBOE+I)=IH1(I)
      IPBE(1,IBOE+I)=IP1(1,I)
      IPBE(2,IBOE+I)=IP1(2,I)
      VBE(1,IBOE+I)=VB1(1,I)
      VBE(2,IBOE+I)=VB1(2,I)
      ISFP(1,IBOE+I)=ISF1(1,I)
70    ISFP(2,IBOE+I)=ISF1(2,I)

      I1I1=I1I-1
      DO 80 I=ICE,I1I1
      IR=I1I1-I+ICE
      FME(IR+1)=FME(IR)
      IOFME(IR+1)=IOFME(IR)
80    NPE(IR+1)=NPE(IR)
      NPE(ICE)=NMP
      FME(ICE)=FM1
      IOFME(ICE)=IOF1

100   RETURN
      END
```

```
C **********************SUBROUTINE : EEST; CALLED BY ES*****************

C ESTIMATES AN EDGE FORM ONE OR MORE EDGE-PEAK PAIRS USING A LOOK-UP OF THE
C TABLE SHOWN BELOW.

      SUBROUTINE EEST(IBOE,ICE,PE,IE,A,AMF,IER)
      COMMON/AE/NE,NPE(20),IPBE(2,50),VBE(2,50),IHP(50),ISFP(2,50)
      COMMON/FM/FME(20),IOFME(20)
      COMMON/ED/IED(4)
      COMMON/ESTDUM/ IH(2),IP(2,2),VBT(2,2)
      COMMON/DUM1/BH(4,7)

      DATA BH/
     *    165.93,  111.91,  113.54,   48.91,
     *     63.63,  100.11,  120.26,   53.31,
     *     53.31,   92.72,  128.87,   58.46,
     *     42.19,   85.15,  129.34,   60.30,
     *     35.12,   82.06,  132.02,   62.97,
     *     24.44,   73.02,  134.53,   66.14,
     *     20.56,   65.45,  135.66,   68.92/

      IER=0
      IF(NPE(ICE).NE.1)GO TO 01
      IER=1
      IHD=IHP(IBOE+1)
      PE=IPBE(1,IBOE+1)+(IPBE(2,IBOE+1)-IPBE(1,IBOE+1))/2.0
      BD=ABS(VBE(1,IBOE+1)-VBE(2,IBOE+1))*(VBE(2,IBOE+1)/ABS(VBE(2,IBOE+1)))
      GO TO 02
 01   ID=IOFME(ICE)-1
      DO 010 I=1,2
      IH(I)=IHP(IBOE+ID+I)
      IP(1,I)=IPBE(1,IBOE+ID+I)
      IP(2,I)=IPBE(2,IBOE+ID+I)
      VBT(1,I)=VBE(1,IBOE+ID+I)
010   VBT(2,I)=VBE(2,IBOE+ID+I)

      PE=IP(1,1)+(IP(2,1)-IP(1,1))/2.0
      B1=ABS(VBT(1,1)-VBT(2,1))*(VBT(2,1)/ABS(VBT(2,1)))
      B2=ABS(VBT(1,2)-VBT(2,2))*(VBT(2,2)/ABS(VBT(2,2)))

      DO 10 I=1,7
      IF(B1/B2.GT.BH(IH(1),I)/BH(IH(2),I))GO TO 21
 10   CONTINUE
      IER=1
      IE=6
      A=0
      AMF=B2/BH(IH(2),7)
      IF(IH(2).EQ.4)AMF=B1/BH(3,7)
```

```
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)

      GO TO 300

 21   IF(I.NE.1)GO TO 23
      IER=1
      IE=1
      A=1
      IF(IH(1).EQ.2)AMF=B1/BH(2,1)
      IF(IH(2).EQ.2)AMF=B2/BH(2,1)
      IF(IH(1).EQ.3)AMF=B1/BH(3,1)
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)
      GO TO 300


 23   IE=I-1
      DET=B1*(BH(IH(2),I)-BH(IH(2),IE))-(BH(IH(1),I)-BH(IH(1),IE))*B2
      A=(B1*BH(IH(2),I)-BH(IH(1),I)*B2)/DET
      AMF=DET/(BH(IH(1),I)*(BH(IH(2),I)-BH(IH(2),IE))
     *          -(BH(IH(1),I)-BH(IH(1),IE))*BH(IH(2),I))
      IF(ABS(AMF).GT.1.0)IER=1
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)
      GO TO 300

 02   A=1
      IE=IED(IHD)
      AMF=BD/BH(IHD,IE)
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)
      IF(IE.EQ.7)A=0
      IF(IE.EQ.7)IE=6

300   RETURN
      END
```

```
C **********************SUBROUTINE : AECORR; CALLED BY ES*****************

C CORRECTS ALL REMAINING EDGE PEAKS FOR THE JUST ESTIMATED EDGE.

      SUBROUTINE AECORR(IBOE,ICE,PE,IE,A,AMF)
      COMMON/AE/NRGE,NPE(20),IPBE(2,50),VBE(2,50),IHP(50)
      ICE1=ICE+1
      IC=IBOE
      DO 53 N=ICE1,NRGE
      NMP=NPE(N)
      DO 53 I=1,NMP
      IC=IC+1
      DO 53 J=1,2
      CALL ESCORR(IHP(IC),IPBE(J,IC),VBE(J,IC),IE,PE,A,AMF)
   53 CONTINUE
      RETURN
      END
```

```
C **********************SUBROUTINE : ESCORR; CALLED BY ES*****************


C CORRECTS AN H-FILTER PEAK FOR THE JUST ESTIMATED EDGE.


        SUBROUTINE ESCORR(IHS,IPS,BS,IE,PE,A,AMF)
        COMMON/ER/IER(3584)
        COMMON/NP/NPEL,NPL4
C THER IS NO CHECK HERE TO SEE IF THE CORRECTION IS ZERO WHEN THE EDGE AND THE
C SINGLE ARE TOO FAR APART. THAT MAY BE ADDED IN THE FUTURE.
        PD=ABS(FLOAT(IPS)-PE)
        PDA=0.5
        IF((INT(PD)-PD).EQ.0.0)PDA=0
        IF(IPS.EQ.0)TYPE 'NPEL =',NPEL,' PE=',PE,'  PD=',PD
        IF(IPS.EQ.0)TYPE 'BS=',BS
        ID=(4*(IE-1)+(IHS-1))*NPEL+PD+PDA
        IF(PD.EQ.0.0)ID=ID+1
        CR=AMF*(A*IER(ID)+(1.0-A)*IER(ID+NPL4))/128.0
        IF(FLOAT(IPS).LT.PE)CR=-1*CR
        IF(PD.EQ.0.0)CR=ABS(CR)*(BS/ABS(BS))
        IDN4=ID+NPL4
        IF(IPS.EQ.0)TYPE 'ID=',ID,IER(ID),IER(IDN4)
        IF(IPS.EQ.0)TYPE 'CR=',CR
        BS=BS-CR
        IF(IPS.EQ.0)TYPE 'BS=',BS


        RETURN
        END
```

```
C **********************SUBROUTINE : ECLN; CALLED BY ES*****************

C REMOVES ALL EDGE PEAKS THAT NO LONGER MEET THE CRITERIA AFTER BEING
C CORRECTED.

      SUBROUTINE ECLN(BETH,IBOE,ICE,PCTHR)
      COMMON/AE/NRGE,NPE(20),IPBE(2,50),VBE(2,50),IHP(50),ISFP(2,50)
      COMMON/FM/FME(20),IOFME(20)
      IDU=0
      IC=IBOE
      ICE1=ICE+1
      DO 20 N=ICE1,NRGE
      NMP=NPE(N)

      DO 21 I=1,NMP
      IC=IC+1
      B1=VBE(1,IC)
      B2=VBE(2,IC)
      IF(((ABS(B1).LE.BETH).AND.(ABS(B2).LE.BETH))
     *   .OR.(B1/ABS(B1).NE.ISFP(1,IC))
     *   .OR.(B2/ABS(B2).NE.ISFP(2,IC))
     *   .OR.((N.EQ.ICE1).AND.(ABS((B1+B2)/(B1-B2)).GT.PCTHR)))
     * GO TO 021
      GO TO 0021
  021 IDU=IDU+1
      NPE(N)=NPE(N)-1
      GO TO 21
 0021 IF(IDU.EQ.0)GO TO 21
      IHP(IC-IDU)=IHP(IC)
      IPBE(1,IC-IDU)=IPBE(1,IC)
      IPBE(2,IC-IDU)=IPBE(2,IC)
      VBE(1,IC-IDU)=VBE(1,IC)
      VBE(2,IC-IDU)=VBE(2,IC)
      ISFP(1,IC-IDU)=ISFP(1,IC)
      ISFP(2,IC-IDU)=ISFP(2,IC)
   21 CONTINUE
   20 CONTINUE


      IDU=0
      NEC=NRGE
      DO 30 N=ICE1,NRGE
      IF(NPE(N).LT.0)GO TO 40
      IF(NPE(N).EQ.0)GO TO 030
      NPE(N-IDU)=NPE(N)
      IOFME(N-IDU)=IOFME(N)
      GO TO 30
  030 IDU=IDU+1
      NEC=NEC-1
```

```
 30   CONTINUE
      NRGE=NEC
      GO TO 100

 40   TYPE 'NPE(N) IS LESS THAN 0 FOR NPE(N) =',NPE(N),' I =',I
      STOP
100   RETURN
      END
```

```
C **********************SUBROUTINE : EFCORR; CALLED BY ES*****************

C CORRECTS THE FAT-G'S FOR THE JUST ESTIMATED EDGE.

      SUBROUTINE EFCORR(SAMTC,IPF,IE,PE,A,AMF)
      COMMON/FR/IFR(896)
      COMMON/NP/NPEL
C THER IS NO CHECK HERE TO SEE IF THE CORRECTION IS ZERO WHEN THE EDGE AND THE
C FATG SAMPLE  ARE TOO FAR APART. THAT MAY BE ADDED IN THE FUTURE.
      PD=ABS(FLOAT(IPF)-PE)
      IF(PD.EQ.0.0)GO TO 100
      PDA=0.5
      IF((INT(PD)-PD).EQ.0.0)PDA=0
      ID=(IE-1)*NPEL+PD+PDA
      CR=AMF*(A*IFR(ID)+(1-A)*IFR(ID+NPEL))
      IF(FLOAT(IPF).LT.PE)CR=-1*CR
      SAMTC=SAMTC-CR
C

  100 RETURN
      END
```

```
C       ******************PROGRAM : PRG*******************

C MAKES PAIRS OF THE PEAKS IN THE OUTPUT FILE OF "ES" : "CBOD" WHICH
C ARE OF THE SAME SIGN, FALL IN ADJACENT H-FILTERS, AND ARE WITHIN
C THE CORRESPONDING PEL WINDOW : "WH".  THE REST OF THE PEAKS ARE PUT
C IN A SINGLES' LIST.



      COMMON/A/NB(4),IPB(4,100),VB(4,100),WH(3)
      DIMENSION IPM(3),VBM(3),FATS(7),THP(4),THS(4),IFILE(15)
    8 FORMAT(1X,I2,/,4X,I2,',   ',I4,',   ',F8.3,',   ',F8.3)
    6 FORMAT(4X,I2,',   ',I4,',   ',F8.3,',   ',F8.3)
    4 FORMAT(4X,I2,',   ',I4,',   ',F8.3)
    3 FORMAT(S1)
    5 FORMAT(S29)

C OPEN FILE TO OUTPUT PAIRS AND SINGLES TO.
      CALL DFILW("PRGOD",IER)
      CALL CFILW("PRGOD",2,IER)
      CALL FOPEN(2,"PRGOD")

      CALL FOPEN(15,"BEDID")
      READ(15)IDUM
      READ(15,3)IDUM
      READ(15)IFIL,ILIL
      READ(15)IDUM
      READ(15)DUM,WH

      CALL FOPEN(3,'LESTID')
C READ THRESHOLDS FOR RETAINING PAIRS AND SINGLES.
      READ(3)THP,THS
      CALL FCLOS(3)

      CALL FOPEN(17,"PRGID")
C READ NAME OF INPUT FILE (USUALLY : CBOD).
      READ(17,5)IFILE(1)
      CALL FOPEN(13,IFILE)

      DO 100 L=IFIL,ILIL
      DO 101 J=1,4
      READ(13)NB(J)
      IF(NB(J).EQ.0)GO TO 101
      N=NB(J)
      DO 103 I=1,N
  103 READ(13)IPB(J,I),VB(J,I)
  101 CONTINUE
      READ(13)FATS
      ITNB=NB(1)+NB(2)+NB(3)+NB(4)
```

```
      DO 102 J=1,4
      IF(NB(J).EQ.0)GO TO 102
      N=NB(J)
      DO 1021 I=1,N
      NDPR=0
      IF(J.EQ.4)GO TO 01021
      SF=VB(J,I)/ABS(VB(J,I))
C CHECK FOR ESTABLISHING THE FIRST PAIR.
      CALL PAIRG(I,IPB(J,I),VB(J,I),SF,J,IFLAG,IPM(1),VBM(1),IX)
      NDPR=NDPR+IFLAG
      IF((IFLAG.EQ.0).OR.(J.EQ.3))GO TO 01021
      J1=J+1
      SF=VBM(1)/ABS(VBM(1))

C ANOTHER PEAK ADDED TO THE PAIR ?
      CALL PAIRG(IX,IPM(1),VBM(1),SF,J1,IFLAG,IPM(2),VBM(2),IY)
      NDPR=NDPR+IFLAG
      IF((IFLAG.EQ.0).OR.(J.EQ.2))GO TO 01021
      J2=J+2
      SF=VBM(2)/ABS(VBM(2))

C A FORTH PEAK ADDED TO THE PAIR ?
      CALL PAIRG(IY,IPM(2),VBM(2),SF,J2,IFLAG,IPM(3),VBM(3),IDUM)
      NDPR=NDPR+IFLAG
01021 IF(NDPR.EQ.0)GO TO 1021
      ITNB=ITNB-(NDPR+1)
      IPI=IPB(J,I)
      IPB(J,I)=0
      IF((NDPR.EQ.1).AND.(ABS(VB(J,I)).LT.THP(J))
     *              .AND.(ABS(VBM(1)).LT.THP(J+1)))GO TO 1021
      IF(J.EQ.3)GO TO 021
      IF((NDPR.EQ.2).AND.(ABS(VB(J,I)).LT.THP(J))
     *              .AND.(ABS(VBM(1)).LT.THP(J+1))
     *              .AND.(ABS(VBM(2)).LT.THP(J+2)))GO TO 1021
      IF((NDPR.EQ.3).AND.(ABS(VB(J,I)).LT.THP(1))
     *              .AND.(ABS(VBM(1)).LT.THP(2))
     *              .AND.(ABS(VBM(2)).LT.THP(3))
     *              .AND.(ABS(VBM(3)).LT.THP(4)))GO TO 1021

C IF THE PAIR MEETS THE THRESHOLD CRITERION, SAVE IT.
 021  WRITE(2,8)NDPR,J,IPI,VB(J,I),VBM(1)
      IF(NDPR.EQ.1)GO TO 1021
      DO 21 K=2,NDPR
      K1=K-1
      IH=J+K1
      WRITE(2,6)IH,IPM(K1),VBM(K1),VBM(K)
  21  CONTINUE
1021  CONTINUE
 102  CONTINUE
      IZ=0
      WRITE(2)IZ
```

```
C STORE THE REST OF THE PEAKS AS SINGLES.
      DO 210 J=1,3
      IF(NB(J).EQ.0)GO TO 210
      N=NB(J)
      DO 211 I=1,N
      IF((IPB(J,I).NE.0).AND.(ABS(VB(J,I)).LT.THS(J)))ITNB=ITNB-1
  211 CONTINUE
  210 CONTINUE
      ITNB=ITNB-NB(4) ;**********************************
      WRITE(2)ITNB

      DO 110 J=1,3    ;**********************************
      IF(NB(J).EQ.0)GO TO 110
      N=NB(J)
      DO 111 I=1,N
      IF((IPB(J,I).EQ.0).OR.(ABS(VB(J,I)).LT.THS(J)))GO TO 111
      WRITE(2,4)J,IPB(J,I),VB(J,I)
  111 CONTINUE
  110 CONTINUE
      DO 112 I=1,7
  112 WRITE(2)FATS(I)

      TYPE 'PRG',L
  100 CONTINUE

      STOP
      END
```

```
C       ******************SUBROUTINE : PAIRG; CALLED BY PRG*******************

C CHECKS TWO PEAKS FOR THE CRITERIA OF MAKING A PAIR.

        SUBROUTINE PAIRG(I1,IPI,VBI,SF,J,IFLAG,IPM,VBM,I2)
        COMMON/A/NB(4),IPB(4,100),VB(4,100),WH(3)
        IFLAG=0
        J1=J+1
        IF(NB(J1).EQ.0)GO TO 200
        NBH=NB(J1)
        DO 20 I=1,NBH
        ID=IABS(IPI-IPB(J1,I))
        IF((ID.LE.WH(J)).AND.
     *     ((VB(J1,I)/ABS(VB(J1,I))).EQ.SF))GO TO 020
   20   CONTINUE
        GO TO 200

  020   IF(I1.EQ.NB(J))GO TO 0020
        I11=I1+1
        ID1=IABS(IPB(J,I11)-IPB(J1,I))
      IF((ID1.GT.ID).OR.((ID1.EQ.ID).AND.((ABS(VBI).GE.ABS(VB(J,I11)))
     *                                    .OR.(VB(J,I11)/ABS(VB(J,I11)).NE.SF))
     *  ))GO TO 0020
        IF((ID1.LE.ID).AND.((VB(J,I11)/ABS(VB(J,I11))).EQ.SF))GO TO 200
        ;AT THIS POINT VB(J,I11) HAS TO HAVE AN OPPOSITE SIGN TO SF AND ID1<ID.
        ;SO WE LOOK AT THE NEXT CONSECUTIVE PEAK IN J.
        IF(I11.EQ.NB(J))GO TO 0020
        I12=I1+2
        ID2=IABS(IPB(J,I12)-IPB(J1,I))
      IF((ID2.GT.ID).OR.((ID2.EQ.ID).AND.((ABS(VBI).GE.ABS(VB(J,I12)))
     *                                    .OR.(VB(J,I12)/ABS(VB(J,I12)).NE.SF))
     *  ))GO TO 0020
        IF((ID2.LE.ID).AND.((VB(J,I12)/ABS(VB(J,I12))).EQ.SF))GO TO 200
 0020   IFLAG=1
        I2=I-1
        IPM=IPB(J1,I)
        VBM=VB(J1,I)
        IF(I.EQ.NBH)GO TO 0200
        IN=I+1
        DO 21 I=IN,NBH
        IPB(J1,I-1)=IPB(J1,I)
   21   VB(J1,I-1)=VB(J1,I)

 0200   NB(J1)=NB(J1)-1
  200   RETURN
        END
```

```
C       ******************PROGRAM : LEST******************

C ESTIMATES GAUSSIANS USING THE OUTPUT OF "PRG" : "PRGOD". PERFORMS THE
C CORRECTIONS AMONG THE PAIRS, SINGLES AND FAT-G'S.



        COMMON/HR/IHR(4096)
        COMMON/FR/IFR(2688)
        COMMON/PAIRS/IHP(100),IPP(100),VP(100,2),ISFP(100)
        COMMON/NMPG/NPG(100),IH1G(100),ABBG(100)
        COMMON/SINGLES/IHS(100),IPS(100),VS(100),ISFS(100)
        COMMON/IMCORR/NPEL,IBW(19),I19GS(2432)
        COMMON/GD/IGD(4)
        COMMON/CLN/THP(4),THS(4),THR2
        DIMENSION IPF(7),SAMP(7)
        DATA IBW/ 2, 4, 6, 8, 10, 12, 14, 16, 18, 22, 26, 30, 36, 44, 52, 64,
     *           78,, 98, 122/
    4   FORMAT(2X,I3,'   ',I3,'   ',F7.3'   ',F7.3)
    6   FORMAT(S1)

C READ THE TABLE LOOK-UP FILE FOR H-FILTER PEAK CORRECTIONS.
        CALL MAPDF(IHR,11)
        CALL GCHN(ICHN)
        CALL EOPEN(ICHN,"HSRS",0,MYER)
        TYPE 'MYER',MYER
        CALL ERDB(ICHN,0,42,0)
C READ THE TABLE LOOK-UP FILE FOR THE FAT-G'S CORRECTIONS.
        CALL FOPEN(7,"FTGRS")
        READ BINARY(7)IFR
        TYPE "IFR READ"
        CALL FCLOS(7)
C READ THE GAUSSIANS WHICH THE ESTIMATED G'S ARE LINEAR COMBINATIONS OF.
        CALL FOPEN(7,"BN21GS")
        READ BINARY(7)I19GS
        CALL FCLOS(7)
        TYPE "I19GS READ"
        CALL FOPEN(15,'BEDID')
        READ(15)NPEL
        READ(15,6)IDUM
        READ(15)IFIL,ILIL
        TYPE "BEDID READ"
        CALL FOPEN(21,'LESTID')
        READ(21)THP,THS,THR2,IGD,IED
        TYPE "LESTID READ"
C OPEN FILE FOR OUTPUTTING THE ESTIMATED G'S.
        CALL DFILW("LEOD",IER)
        CALL CFILW("LEOD",2,IER)
        CALL FOPEN(18,'LEOD')
        TYPE "LEOD OPENED"
```

```
      IZ=0
      FZ=0
      IPF(1)=1
      IPF(2)=22
      IPF(3)=43
      IPF(4)=64
      IPF(5)=85
      IPF(6)=106
      IPF(7)=128
      CALL FOPEN(19,'PRGOD')
      TYPE "PRGOD OPENED"

      DO 300 LINE=IFIL,ILIL
      IBO=0
      NG=0
      ID=0
 10   READ(19)NMP
      IF(NMP.EQ.0)GO TO 041
      NG=NG+1
      NPG(NG)=NMP
      DO 40 I=1,NMP
      ID=ID+1
      READ(19)IHP(ID),IPP(ID),VP(ID,1),VP(ID,2)
      ISFP(ID)=VP(ID,1)/ABS(VP(ID,1))
 40   CONTINUE
      GO TO 10
041   READ(19)NS
      IF(NS.EQ.0)GO TO 42
      DO 41 I=1,NS
      READ(19)IHS(I),IPS(I),VS(I)
 41   ISFS(I)=VS(I)/ABS(VS(I))
 42   READ(19)SAMP


C FORWARD CORRECTION FOR PAIRS.
      IF(NG.EQ.0)GO TO 100
      NGC=0
      IF(NG.EQ.1)GO TO 50
C RESORT PAIRS TO START WITH THE LARGEST, NARROWEST ONE.
      CALL PRST(NGC,IBO,NG)
 50   NGC=NGC+1

C ESTIMATE G POSITION WHEN THERE IS A MULTIPLE PAIR.
      CALL PPEST(NPG(NGC),IBO,IPG)
C ESTIMATE G FROM THE MULTIPLE PAIR
      CALL MPGEST(NPG(NGC),IBO,IG,A,AMF,IER)

      IF((NG.EQ.NGC).AND.(IER.EQ.2))GO TO 060
      IF(NG.EQ.NGC)GO TO 055
      IF(IER.EQ.2)GO TO 50
      CALL APCORR(NGC,IBO,NG,IG,IPG,A,AMF,IED)
```

```
                                        ;FOR'D CORR. PAIRS FOR ONE CORR'ING PAIR.
  055   IF(NS.EQ.0)GO TO 057
        CALL ASCORR(0,NS,IG,IPG,A,AMF,IED)
                                    ;FOR'D CORR. ALL SINGLES FOR ONE CORR'ING PAIR.
  057   CALL PCLN(NGC,IBO,NG,NS)
                                      ;CLEAN UP THE PAIRS AFTER A FOR'D CORRECTION.
        IF(NG.EQ.NGC)GO TO 060
        IF(NG-NGC.EQ.1)GO TO 50
C RESORT PAIRS.
        CALL PRST(NGC,IBO,NG)
        GO TO 50

C FORWARD CORRECTION FOR SINGLES.

  060   IF(NS.EQ.0)GO TO 0163
        CALL SCLN(0,NS)
        TYPE 'PAIRS FORWARD DONE'


  100   IF(NS.EQ.0)GO TO 0163
        IF(NS.EQ.1)GO TO 0160
        NSC=0
C RESORT SINGLES.
        CALL SRST(NSC,NS)
  150   NSC=NSC+1
C ESTIMATE A G FROM A SINGLE.
        CALL DGEST(IHS(NSC),VS(NSC),IG,A,AMF)

        IF(NS.EQ.NSC)GO TO 0160
        CALL ASCORR(NSC,NS,IG,IPS(NSC),A,AMF,IED)
                                    ;FOR'D CORR. SINGLES FOR ONE CORR'ING SINGLE.
        CALL SCLN(NSC,NS)

        IF(NS.EQ.NSC)GO TO 0160
        IF(NS-NSC.EQ.1)GO TO 150
        CALL SRST(NSC,NS)
        GO TO 150

C BACKWARD CORRECTION FOR SINGLES.

 0160   TYPE 'SINGLES FORWARD DONE'
        NSCB=-1
  160   NSCB=NSCB+1
        NSC=NS-NSCB

        CALL DGEST(IHS(NSC),VS(NSC),IG,A,AMF)

        IF(NSC.EQ.1)GO TO 0161
        CALL ASCORR(0,(NSC-1),IG,IPS(NSC),A,AMF,IED)
                                    ;BACK'D CORR. SINGLES FOR ONE CORR'ING SINGLE.
        CALL SCLN(0,NS)
```

```
              ;FOR THE SINGLES BETWEEN NSC AND NS INCLUSIVE, SCLN WILL JUST
              ;PACK THEM IN.
 0161    IF(NG.EQ.0)GO TO 161
         CALL APCORR(0,0,NG,IG,IPS(NSC),A,AMF,IED)
                                    ;BACK'D CORR. PAIRS FOR ONE CORR'ING SINGLE.
  161    IF(NS.EQ.NSCB+1)GO TO 162
         GO TO 160

  162    IF(NG.EQ.0)TYPE 'SINGLES BACKWARD DONE'
         IF(NG.EQ.0)GO TO 200
         IBO=0
         CALL PCLN(0,IBO,NG,NS)
                          ;CLEAN UP THE PAIRS AFTER ALL CORRECTION FOR SINGLES.
         TYPE 'SINGLES BACKWARD DONE'

C BACKWARD CORRECTION FOR PAIRS.

 0163    IF(NG.EQ.0)GO TO 200
         NGCB=-1
  163    NGCB=NGCB+1
         NGC=NG-NGCB

         IBOB=0
         DO 164 N=0,NGCB
  164    IBOB=IBOB+NPG(NG-N)
         IBOC=IBO-IBOB

         CALL PPEST(NPG(NGC),IBOC,IPG)
         CALL MPGEST(NPG(NGC),IBOC,IG,A,AMF,IER)

         IF(NGC.EQ.1)TYPE 'PAIRS BACKWARD DONE'
         IF(NGC.EQ.1)GO TO 200
         IF(IER.EQ.2)GO TO 163
         CALL APCORR(0,0,(NGC-1),IG,IPG,A,AMF,IED)
                                    ;BACK'D CORR. PAIRS FOR ONE CORR'ING PAIR.
         IBO=0
         CALL PCLN(0,IBO,NG,NS)
               ;FOR THE PAIRS BETWEEN NGC AND NG INCLUSIVE, PCLN WILL JUST
               ;PACK THEM IN.
         IF(NG.EQ.NGCB+1)TYPE 'PAIRS BACKWARD DONE'
         IF(NG.EQ.NGCB+1)GO TO 200
         GO TO 163

C FAT-G SAMPLES CORRECTION AND ESTIMATION.

  200    IF((NG.EQ.0).AND.(NS.EQ.0))GO TO 02100

         IF(NG.EQ.0)GO TO 080
         IBO=0
         DO 70 N=1,NG
         CALL PPEST(NPG(N),IBO,IPG)
```

```
      CALL MPGEST(NPG(N),IBO,IG,A,AMF,IER)
                                          ;PERFORM THE FINAL GEST ESTIMATIONS.
      IF(IER.EQ.2)GO TO 70
      DO 71 I=1,7
   71 CALL FCORR(SAMP(I),IPF(I),IG,IPG,A,AMF,IED)
      WRITE(18,4)IPG,IG,A,AMF
   70 CONTINUE
  080 IF(NS.EQ.0)GO TO 02100
      DO 80 J=1,NS
      CALL DGEST(IHS(J),VS(J),IG,A,AMF)
                                         ;PERFORM THE FINAL DGEST ESTIMATIONS.
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)
      DO 81 I=1,7
   81 CALL FCORR(SAMP(I),IPF(I),IG,IPS(J),A,AMF,IED)
   80 WRITE(18,4)IPS(J),IG,A,AMF

02100 WRITE(18,4)IZ,IZ,FZ,FZ
      DO 2100 I=1,7
      SAMP(I)=SAMP(I)/128.0
 2100 WRITE(18)SAMP(I)

      TYPE 'LEST',LINE
  300 CONTINUE

      STOP
      END
```

```
C  ********************SUBROUTINE : PPEST; CALLED BY LEST*************

C ESTIMATES PAIR POSITION IN CASE OF MULTIPLE PAIRS.

      SUBROUTINE PPEST(NMP,IBO,IPG)
      COMMON/PAIRS/IHP(100),IPP(100),VP(100,2)
      IPG=IPP(IBO+1)
      IF((NMP.EQ.1).OR.(VP(IBO+1,2).LT.VP(IBO+1,1)))GO TO 51
      IF(VP(IBO+2,2).LE.VP(IBO+1,1))GO TO 51
      IPG=IPP(IBO+2)
      IF((NMP.EQ.2).OR.(VP(IBO+2,2).LT.VP(IBO+1,2)))GO TO 51
      IF(VP(IBO+3,2).LE.VP(IBO+1,2))GO TO 51
      IPG=IPP(IBO+3)

   51 RETURN
      END
```

```
C  ********************SUBROUTINE : MPGEST; CALLED BY LEST*************

C ESTIMATES A G FROM ONE OR MORE PAIRS.


      SUBROUTINE MPGEST(NMP,IBO,IG,A,AMF,IER)
      COMMON/PAIRS/IHP(100),IPP(100),VP(100,2)
      COMMON/HAW/WD(21)
      DIMENSION IGA(3),AA(3),AMFA(3),IERA(3)
      WOF(IGI,AI)=AI*WD(IGI)+(1-AI)*WD(IGI+1)
      DATA WD/ 1.0, 2.01, 2.67, 3.11, 3.50, 4.46, 5.06, 5.59, 6.05,
     *         8.36, 9.54, 12.57, 14.42, 17.86, 21.94, 26.85, 34.25, 43.00,
     *         55.00, 68.60, 80.60/

      DO 010 I=1,3
  010 IERA(I)=0
      DO 10 I=1,NMP
      IBO=IBO+1
      CALL GEST(IHP(IBO),VP(IBO,1),VP(IBO,2),IGA(I),AA(I),AMFA(I),IERA(I))
   10 CONTINUE
      IF(((NMP.EQ.1).AND.(IERA(1).LT.2)).OR.
     *   ((NMP.EQ.2).AND.(IERA(1).EQ.IERA(2)).AND.(IERA(1).LT.2)).OR.
     *   ((NMP.EQ.3).AND.(IERA(1).EQ.IERA(2)).AND.(IERA(2).EQ.IERA(3))
     *              .AND.(IERA(1).LT.2)))GO TO 01
      GO TO 020

   01 IER=IERA(1)
      IF(IER.EQ.1)TYPE 'IER=',IER,' IPP=',IPP(IBO)
      W=0
      AMF=0
      DO 20 I=1,NMP
      W=W+WOF(IGA(I),AA(I))
      AMF=AMF+AMFA(I)
   20 CONTINUE
      W=W/NMP
      CALL WTOIG(W,IG,A)
      AMF=AMF/NMP
      GO TO 100


  020 GO TO (21,22,23),NMP

   21 IER=2
      TYPE 'IER=',IER,' IPP=',IPP(IBO)
      GO TO 100

   22 IF(IERA(1).NE.IERA(2))GO TO 221
      IER=2
      TYPE 'IER=',IER,' IPP=',IPP(IBO)
      GO TO 100
```

```
221   J=2
      IF(IERA(1).LT.IERA(2))J=1
      IG=IGA(J)
      A=AA(J)
      AMF=AMFA(J)
      IER=IERA(J)
      IF(IER.EQ.1)TYPE 'IER=',IER,' IPP=',IPP(IBO)
      GO TO 100

 23   IF(((IERA(1).LT.IERA(2)).AND.(IERA(1).LT.IERA(3))).OR.
     *    ((IERA(2).LT.IERA(1)).AND.(IERA(2).LT.IERA(3))).OR.
     *    ((IERA(3).LT.IERA(1)).AND.(IERA(3).LT.IERA(2))))GO TO 231
      IF(((IERA(1).EQ.IERA(2)).AND.(IERA(1).LT.IERA(3))).OR.
     *    ((IERA(1).EQ.IERA(3)).AND.(IERA(1).LT.IERA(2))).OR.
     *    ((IERA(2).EQ.IERA(3)).AND.(IERA(2).LT.IERA(1))))GO TO 232
      IER=2
      TYPE 'IER=',IER,' IPP=',IPP(IBO)
      GO TO 100
231   IER=MIN0(IERA(1),IERA(2),IERA(3))
      IF(IER.EQ.1)TYPE 'IER=',IER,' IPP=',IPP(IBO)
      J=1
      IF(IER.EQ.IERA(2))J=2
      IF(IER.EQ.IERA(3))J=3
      IG=IGA(J)
      A=AA(J)
      AMF=AMFA(J)
      GO TO 100
232   MAXER=MAX0(IERA(1),IERA(2),IERA(3))
      J1=1
      J2=2
      IF(MAXER.EQ.IERA(2))J1=1
      IF(MAXER.EQ.IERA(2))J2=3
      IF(MAXER.EQ.IERA(1))J1=2
      IF(MAXER.EQ.IERA(1))J2=3
      IER=IERA(J1)
      IF(IER.EQ.1)TYPE 'IER=',IER,' IPP=',IPP(IBO)
      W=(WOF(IGA(J1),AA(J1))+WOF(IGA(J2),AA(J2)))/2.0
      CALL WTOIG(W,IG,A)
      AMF=(AMFA(J1)+AMFA(J2))/2.0

100   RETURN
      END
```

```
C  ********************SUBROUTINE : WTOIG; CALLED BY MPGEST*************

C CONVERTS A WIDTH INTO A COEFFICIENT AND A G NO. FOR THE LINEAR COMBINATION
C OF ESTIMATING A G.


      SUBROUTINE WTOIG(W,IG,A)
      COMMON/HAW/WD(21)

      DO 10 I=1,21
      IF(WD(I).GT.W)GO TO 010
   10 CONTINUE

  010 IG=I-1
      A=(W-WD(IG+1))/(WD(IG)-WD(IG+1))
      RETURN
      END
```

```
C  ********************SUBROUTINE : GEST; CALLED BY MPGEST*************

C ESTIMATES A G FROM A SINGLE PAIR USING A LOOK-UP OF THE TABLE BELOW.

      SUBROUTINE GEST(IH,B1,B2,IG,A,AMF,IER)
      COMMON/A1/BH(4,21)
C THE FOLLOWING TABLE HAS THE RESPONSES OF THE FOUR H-FILTRS, ONE IN EACH
C COLOMN, FOR 21 GAUSSIANS WHEN EACH GAUSSIAN HAS A NORMALIZED AMPLITUDE
C OF 255.
      DATA BH/
     *     159.37,   55.41,   26.41,    8.04,
     *      95.37,   81.91,   50.12,   16.02,
     *      69.00,   85.91,   63.46,   21.22,
     *      54.25,   84.45,   72.49,   25.32,
     *      44.62,   81.48,   79.04,   28.77,
     *      29.62,   73.30,   90.26,   35.50,
     *      24.25,   65.70,   94.95,   40.08,
     *      19.87,   59.55,   97.97,   44.16,
     *      17.25,   54.64,   99.32,   47.45,
     *       8.12,   35.78,  101.56,   60.86,
     *       7.75,   30.08,   94.79,   66.69,
     *       3.25,   17.54,   82.46,   79.13,
     *       3.87,   15.52,   71.46,   82.09,
     *       1.87,    9.13,   57.37,   86.74,
     *       1.75,    6.82,   43.12,   84.80,
     *       0.87,    4.06,   31.86,   79.05,
     *       0.75,    2.68,   21.07,   66.70,
     *       0.12,    1.72,   14.21,   52.53,
     *       0.12,    1.20,    8.74,   37.54,
     *       0.12,    0.70,    5.88,   26.67,
     *       0.00,    0.41,    4.04,   20.43/
      IER=0
      IH1=IH
      IH2=IH+1

      DO 10 I=1,15    ;************************************
      IF(B1/B2.GT.BH(IH1,I)/BH(IH2,I))GO TO 21
10    CONTINUE
      IER=2
      GO TO 200

21    IF(I.NE.1)GO TO 22
      IF(IH1.EQ.3) GO TO 022
      IER=1
      IG=1
      A=1
      AMF=B1/BH(IH1,1)
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)
      GO TO 200

22    IF(.NOT.(((IH1.EQ.1).AND.(I.GE.10)).OR.((IH2.EQ.2).AND.(I.GE.14))
```

```
   *          .OR.((IH1.EQ.3).AND.(I.LE.5))))GO TO 23
022   TYPE "NOISEY RATIOS FIX  FOR :","<15>",
   *  "IH = ",IH1," B1 = ",B1," B2 = ",B2
      IER=1
      BD=B1
      IHD=IH1
      IF(ABS(B2).GT.ABS(B1))BD=B2
      IF(ABS(B2).GT.ABS(B1))IHD=IH2
      CALL DGEST(IHD,BD,IG,A,AMF)
      IF(IG.GT.15)IER=2          ;************************************
      GO TO 200


 23   IG=I-1
      DET=B1*(BH(IH2,I)-BH(IH2,IG))-(BH(IH1,I)-BH(IH1,IG))*B2
      A=(B1*BH(IH2,I)-BH(IH1,I)*B2)/DET
      AMF=DET/(BH(IH1,I)*(BH(IH2,I)-BH(IH2,IG))
   *          -(BH(IH1,I)-BH(IH1,IG))*BH(IH2,I))
      IF(ABS(AMF).GT.1.0)IER=1
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)


200   RETURN
      END
```

```
C  ********************SUBROUTINE : DGEST; CALLED BY LEST*************

C ESTIAMTES A G FROM A SINGLE PEAK USING DEFAULT G'S.

      SUBROUTINE DGEST(IH,VB,IG,A,AMF)
      COMMON/A1/DBH(4,21)
      COMMON/GD/IGD(4)
      A=1
      IG=IGD(IH)
      AMF=VB/DBH(IH,IG)
      IF(ABS(AMF).GT.1.0)AMF=AMF/ABS(AMF)
      IF(IG.EQ.21)A=0
      IF(IG.EQ.21)IG=20
      RETURN
      END
```

```
C  ********************SUBROUTINE : APCORR; CALLED BY LEST*************

C CORRECTS ALL REMAINING PAIRS FOR THE JUST ESTIMATED G.

      SUBROUTINE APCORR(NGC,IBO,NG,IG,IPG,A,AMF,IED)
      COMMON/PAIRS/IHP(100),IPP(100),VP(100,2)
      COMMON/NMPG/NPG(100)
      NGC1=NGC+1
      ID=IBO
      DO 53 N=NGC1,NG
      NMP=NPG(N)
      DO 53 I=1,NMP
      ID=ID+1
   53 CALL PCORR(IHP(ID),IPP(ID),VP(ID,1),VP(ID,2),IG,IPG,A,AMF,
     *            VP(ID,1),VP(ID,2),IED)
      RETURN
      END
```

```
C  ********************SUBROUTINE : ASCORR; CALLED BY LEST*************

C CORRECTS ALL REMAINING SINGLES FOR THE JUST ESTIMATED G.

      SUBROUTINE ASCORR(NSC,NS,IG,IPG,A,AMF,IED)
      COMMON/SINGLES/IHS(100),IPS(100),VS(100),ISFS(100)
      NSC1=NSC+1
      ID=NSC
      DO 55 I=NSC1,NS
      ID=ID+1
      CALL SCORR(IHS(ID),IPS(ID),VS(ID),IG,IPG,A,AMF,VS(ID),IED)
55    CONTINUE

      RETURN
      END
```

```
C  ********************SUBROUTINE : PCORR; CALLED BY APCORR*************

C CORRECTS A PAIR FOR THE JUST ESTIMATED G.


        SUBROUTINE PCORR(IHP,IPP,B1TC,B2TC,IG,IPG,A,AMF,B1CD,B2CD,IED)
        COMMON/HR/IHR(4096)
        COMMON/IMCORR/NPEL,IBW(19),I19GS(2432)
        NP2=(NPEL/2.0)+0.5
        NPL4=4*NPEL
        NPL5=5*NPEL
        IPD=IABS(IPP-IPG)

        IF(IPP.EQ.0)TYPE 'IPP =',IPP,' IPG =',IPG,'  IPD=',IPD
      IF(IPD.GE.NPEL)TYPE "@@@@@@@@@@@@@@@@IPD ERROR;IPG=",IPG," IPP=",IPP
C CORRECT THE PAIR FOR THE ORIGINAL(I.E. NOT THE MIROOR IMAGE OF) G.
        ID=(4*(IG-1)+(IHP-1))*NPEL+IPD+1
        CR1=AMF*(A*IHR(ID)+(1.0-A)*IHR(ID+NPL4))/128.0
        CR2=AMF*(A*IHR(ID+NPEL)+(1.0-A)*IHR(ID+NPL5))/128.0
        IDN=ID+NPEL
        IDN4=ID+NPL4
        IDN5=ID+NPL5
        IF(IPP.EQ.0)TYPE 'ID=',ID,IHR(ID),IHR(IDN),IHR(IDN4),IHR(IDN5)
        IF(IPP.EQ.0)TYPE 'B1TC=',B1TC,' B2TC=',B2TC
        IF(IPP.EQ.0)TYPE 'CR1=',CR1,' CR2=',CR2
        B1CD=B1TC-CR1
        B2CD=B2TC-CR2
        IF(IPP.EQ.0)TYPE 'B1CD=',B1CD,' B2CD=',B2CD

  100   RETURN
        END
```

```
C  ********************SUBROUTINE : SCORR; CALLED BY ASCORR*************

C CORRECTS A SINGLE FOR THE JUST ESTIMATED G.


      SUBROUTINE SCORR(IHS,IPS,BTC,IG,IPG,A,AMF,BCD,IED)
      COMMON/HR/IHR(4096)
      COMMON/IMCORR/NPEL,IBW(19),I19GS(2432)
      NP2=(NPEL/2.0)+0.5
      NPL4=4*NPEL
      NPL5=5*NPEL
      IPD=IABS(IPS-IPG)
      IF(IPS.EQ.0)TYPE 'IPS =',IPS,' IPG=',IPG,'  IPD=',IPD
      IF(IPS.EQ.0)TYPE 'BTC=',BTC
    IF(IPD.GE.NPEL)TYPE "@@@@@@@@@@@@@@@@IPD ERROR;IPG=",IPG," IPS=",IPS

      ID=(4*(IG-1)+(IHS-1))*NPEL+IPD+1
      CR=AMF*(A*IHR(ID)+(1.0-A)*IHR(ID+NPL4))/128.0
      IDN4=ID+NPL4
      IF(IPS.EQ.0)TYPE 'ID=',ID,IHR(ID),IHR(IDN4)
      IF(IPS.EQ.0)TYPE 'CR=',CR
      BCD=BTC-CR
      IF(IPS.EQ.0)TYPE 'BCD=',BCD

      RETURN
      END
```

```
C  ********************SUBROUTINE : FCORR; CALLED BY LEST*************

C CORRECTS A FAT-G FOR THE JUST ESTIMATED G.


        SUBROUTINE FCORR(SAMTC,IPF,IG,IPG,A,AMF,IED)
        COMMON/FR/IFR(2688)
        COMMON/IMCORR/NPEL,IBW(19),I19GS(2432)
        NP2=(NPEL/2.0)+0.5

        IPD=IABS(IPF-IPG)
        IF(IPF.EQ.0)TYPE 'NP2 =',NP2,' IPG=',IPG,'  IPD=',IPD
      IF(IPD.GE.NPEL)TYPE "@@@@@@@@@@@@@@@@IPD ERROR;IPG=",IPG," IPF=",IPF

        ID=(IG-1)*NPEL+IPD+1
        CR=AMF*(A*IFR(ID)+(1-A)*IFR(ID+NPEL))
        IDN=ID+NPEL
        IF(IPF.EQ.0)TYPE 'ID=',ID,IFR(ID),IFR(IDN)
        IF(IPF.EQ.0)TYPE 'SAMTC=',SAMTC
        IF(IPF.EQ.0)TYPE 'CR=',CR
        SAMTC=SAMTC-CR
        IF(IPF.EQ.0)TYPE 'SAMTC=',SAMTC

  100   RETURN
        END
```

```
C  ********************SUBROUTINE : PCLN; CALLED BY LEST*************

C REMOVES, OR TURNS INTO SINGLES, ALL PAIRS THAT NO LONGER MEET THE
C PAIR CRITERIA AFTER CORRECTIONS.


      SUBROUTINE PCLN(NGC,IBO,NG,NS)
      COMMON/PAIRS/IHP(100),IPP(100),VP(100,2),ISFP(100)
      COMMON/NMPG/NPG(100)
      COMMON/CLN/THP(4),THS(4),THR2
      NGC1=NGC+1
      ID=IBO
      IDU=0
      DO 20 N=NGC1,NG
      NMP=NPG(N)
      IB2=0
      DO 21 J=1,NMP
      ID=ID+1
      B1=VP(ID,1)
      B2=VP(ID,2)
      THP1=THP(IHP(ID))
      THP2=THP(IHP(ID)+1)
      IF((ABS(B1).LT.THP1).AND.(ABS(B2).LT.THP2)) IB2=0
      IF((ABS(B1).LT.THP1).AND.(ABS(B2).LT.THP2)) GO TO 021
      IF((ABS(B1).GE.THR2).AND.(ABS(B2).GE.THR2)
     *                    .AND.(B1/ABS(B1).EQ.B2/ABS(B2)))GO TO 0021
      IF((ABS(B1).LT.THP1).OR.(IB2.EQ.1))GO TO 23
      CALL SAPP(IHP(ID),IPP(ID),B1,NS)
   23 IF(((NMP.GT.1).AND.(J.LT.NMP)).OR.(ABS(B2).LT.THP2).OR.((IHP(ID)+1).EQ.4)
     *  )IB2=0
      IF(((NMP.GT.1).AND.(J.LT.NMP)).OR.(ABS(B2).LT.THP2).OR.((IHP(ID)+1).EQ.4)
     *  )GO TO 021
      ;****************************************
      IB2=1
      IHSI=IHP(ID)+1
      CALL SAPP(IHSI,IPP(ID),B2,NS)
  021 IDU=IDU+1
      NPG(N)=NPG(N)-1
      GO TO 21
 0021 IF(B1/ABS(B1).NE.ISFP(ID))ISFP(ID)=B1/ABS(B1)
      IHP(ID-IDU)=IHP(ID)
      IPP(ID-IDU)=IPP(ID)
      VP(ID-IDU,1)=VP(ID,1)
      VP(ID-IDU,2)=VP(ID,2)
      ISFP(ID-IDU)=ISFP(ID)
      IB2=1
   21 CONTINUE
   20 CONTINUE


      IDU=0
```

```
      NGT=NG
      DO 30 N=NGC1,NG
      IF(NPG(N).LT.0)GO TO 40
      IF(NPG(N).EQ.0)GO TO 030
      NPG(N-IDU)=NPG(N)
      GO TO 30
030   IDU=IDU+1
      NGT=NGT-1
 30   CONTINUE
      NG=NGT

      IF(NG.LT.NGC)TYPE 'NG < NGC',NGC,IBO,NG
      IF(NG.LT.NGC)STOP

      GO TO 0100
 40   TYPE 'NPG(N) IS LESS THAN 0 FOR NPG(N) =',NPG(N),' N =',N
      TYPE NGC,IBO,NG
      STOP
0100  IF(IBO.NE.0)GO TO 100
      DO 50 N=1,NG
 50   IBO=IBO+NPG(N)

100   RETURN
      END
```

```
C  ********************SUBROUTINE : SAPP; CALLED BY PCLN*************

C APPENDS THE SINGLES THAT CAME FORM PAIRS TO THE LIST OF SINGLES.

      SUBROUTINE SAPP(IHSI,IPSI,VSI,NS)
      COMMON/SINGLES/IHS(100),IPS(100),VS(100),ISFS(100)
      IF(NS.NE.0)GO TO 010
      IHS(1)=IHSI
      IPS(1)=IPSI
      VS(1)=VSI
      ISFS(1)=VSI/ABS(VSI)
      GO TO 100
010   DO 10 I=1,NS
      IF((IHSI.LT.IHS(I)).OR.((IHSI.EQ.IHS(I)).AND.(ABS(VSI).GT.ABS(VS(I)))))
     * GO TO 020
 10   CONTINUE
      IHS(NS+1)=IHSI
      IPS(NS+1)=IPSI
      VS(NS+1)=VSI
      ISFS(NS+1)=VSI/ABS(VSI)
      GO TO 100
020   DO 20 J=I,NS
      JP=NS-J+I
      IHS(JP+1)=IHS(JP)
      IPS(JP+1)=IPS(JP)
      VS(JP+1)=VS(JP)
 20   ISFS(JP+1)=ISFS(JP)
      IHS(I)=IHSI
      IPS(I)=IPSI
      VS(I)=VSI
      ISFS(I)=VSI/ABS(VSI)
100   NS=NS+1
      RETURN
      END
```

```
C  ********************SUBROUTINE : SCLN; CALLED BY LEST*************

C REMOVES ALL SINGLES THAT NO LONGER MEET THE SINGLE CRITERIA AFTER
C CORRECTIONS.


      SUBROUTINE SCLN(NSC,NS)
      COMMON/SINGLES/IHS(100),IPS(100),VS(100),ISFS(100)
      COMMON/CLN/THP(4),THS(4)
      NSC1=NSC+1
      NST=NS
      IDU=0
      DO 21 I=NSC1,NS
      IF(ABS(VS(I)).GT.THS(IHS(I)))GO TO 0021
      IDU=IDU+1
      NST=NST-1
      GO TO 21
0021  IHS(I-IDU)=IHS(I)
      IPS(I-IDU)=IPS(I)
      VS(I-IDU)=VS(I)
      ISFS(I-IDU)=ISFS(I)
 21   CONTINUE
      NS=NST

      RETURN
      END
```

```
C  ********************SUBROUTINE : PRST; CALLED BY LEST*************

C RESORTS PAIRS TO POSITION THE LARGEST, NARROWEST ON TOP.

      SUBROUTINE PRST(NGC,IBO,NG)
      COMMON/PAIRS/IHP(100),IPP(100),VP(100,2),ISFP(100)
      COMMON/NMPG/NPG(100),IH1G(100),ABBG(100)
      DIMENSION IHP1(3),IPP1(3),VP1(3,2),ISFP1(3)
      NGC1=NGC+1

      ID=IBO+1
      DO 20 I=NGC1,NG
      IH1G(I)=IHP(ID)
      ABBG(I)=AMAX1(ABS(VP(ID,1)),ABS(VP(ID,2)))
20    ID=ID+NPG(I)

      I1I=NGC1
      NGC2=NGC1+1
      DO 30 I=NGC2,NG
      IF((IH1G(I).GT.IH1G(I1I))
     *   .OR.((IH1G(I).EQ.IH1G(I1I)).AND.(ABBG(I).LT.ABBG(I1I))))GO TO 30
      I1I=I
30    CONTINUE
      IF(I1I.EQ.NGC1)GO TO 100

      I1I1=I1I-1
      ID1=IBO
      DO 40 I=NGC1,I1I1
40    ID1=ID1+NPG(I)
      NMP=NPG(I1I)
      DO 50 I=1,NMP
      IHP1(I)=IHP(ID1+I)
      IPP1(I)=IPP(ID1+I)
      VP1(I,1)=VP(ID1+I,1)
      VP1(I,2)=VP(ID1+I,2)
      ISFP1(I)=ISFP(ID1+I)
50    CONTINUE
      IBO1=IBO+1
      DO 60 I=IBO1,ID1
      IR=ID1-I+IBO1
      IHP(IR+NMP)=IHP(IR)
      IPP(IR+NMP)=IPP(IR)
      VP(IR+NMP,1)=VP(IR,1)
      VP(IR+NMP,2)=VP(IR,2)
      ISFP(IR+NMP)=ISFP(IR)
60    CONTINUE
      DO 70 I=1,NMP
      IHP(IBO+I)=IHP1(I)
      IPP(IBO+I)=IPP1(I)
      VP(IBO+I,1)=VP1(I,1)
```

```
      VP(IBO+I,2)=VP1(I,2)
      ISFP(IBO+I)=ISFP1(I)
 70   CONTINUE

      DO 80 I=NGC1,I1I1
      IR=I1I1-I+NGC1
 80   NPG(IR+1)=NPG(IR)
      NPG(NGC1)=NMP

100   RETURN
      END
```

```
C  ********************SUBROUTINE : SRST; CALLED BY LEST*************

C RESORTS SINGLES TO POSITION THE LARGEST, NARROWEST ON TOP.

      SUBROUTINE SRST(NSC,NS)
      COMMON/SINGLES/IHS(100),IPS(100),VS(100),ISFS(100)
      NSC1=NSC+1
      NSC2=NSC+2
      I1I=NSC1
      DO 30 I=NSC2,NS
      IF((IHS(I).GT.IHS(I1I))
     *   .OR.((IHS(I).EQ.IHS(I1I))
     *        .AND.(ABS(VS(I)).LT.ABS(VS(I1I))))))GO TO 30
      I1I=I
   30 CONTINUE
      IF(I1I.EQ.NSC1)GO TO 100

      IHS1=IHS(I1I)
      IPS1=IPS(I1I)
      VS1=VS(I1I)
      ISFS1=ISFS(I1I)
      I1I1=I1I-1
      DO 60 I=NSC1,I1I1
      IR=I1I1-I+NSC1
      IHS(IR+1)=IHS(IR)
      IPS(IR+1)=IPS(IR)
      VS(IR+1)=VS(IR)
   60 ISFS(IR+1)=ISFS(IR)
      IHS(NSC1)=IHS1
      IPS(NSC1)=IPS1
      VS(NSC1)=VS1
      ISFS(NSC1)=ISFS1

  100 RETURN
      END
```

```
C       FCORR2.FR
C
C       FIT THE FAT GAUSSIANS TO (ORIGINAL IMAGE - (G'S AND E'S))
C
        REAL F(128,7),PHI(7,7),PSI(7),XK(7)
        INTEGER IH(128)
        INTEGER IGP(64),IG(128),IFILE(15)
    3   FORMAT(S29)
50      FORMAT(2X,I3,',  ',I3,',  ',F7.3,',  ',F7.3)
        CALL FOPEN(17,'FCORR2ID')
        READ(17,3)IFILE(1)
        CALL FOPEN (20,IFILE)
        CALL FOPEN (21,"IMROD")
        CALL FOPEN (22,"INVPHI")
        CALL FOPEN (23,"IMAGES")
        CALL FOPEN (30,"LEOD")
        CALL FOPEN (31,"LEOD1")
        DO 2 J=1,7
        DO 1 I=1,7
1       READ (22) PHI(I,J)
        DO 2 I=1,128
2       READ (23) F(I,J)
        TYPE "BEGIN CORRECTION"
        DO 1000 K=1,128
        TYPE "FCORR2",K
        READ BINARY (20) IGP
        DO 010 I=1,64
        IG(2*I-1)=ISHFT(IGP(I),-8)
  010   IG(2*I)=IGP(I).AND.255
        READ BINARY (21) IH
        DO 5 I=1,7
        XK(I)=0
5       PSI(I)=0
        DO 10 J=1,7
        DO 10 I=1,128
10      PSI(J)=PSI(J)+(IG(I)-IH(I)+128)*F(I,J)
        DO 20 J=1,7
        DO 20 I=1,7
20      XK(J)=XK(J)+PHI(I,J)*PSI(I)*255
30      READ (30) IX,N,W,A
        WRITE (31,50) IX,N,W,A
        IF (N.NE.0) GO TO 30
        DO 40 I=1,7
        READ (30) DUMMY
        GO TO 40
        IF (XK(I).GT.255) XK(I)=255
        IF (XK(I).LT.0) XK(I)=0
40      WRITE (31) XK(I)
 1000    CONTINU
        STOP
        END
```

```
C       *****************PROGRAM : IMRECON******************

C READS THE G'S AND THE FAT-G'S FROM THE OUTPUT OF "LEST" AND "FCORR2": "LEOD",
C AND THE E'S FORM THE OUTPUT OF "ES" : "ESOD", AND FORMS THE RECONSTRUCTED
C IMAGE AS THEIR SUM.


        COMMON/DUM/ IALLG(2688),IBUF(512),ALLE(896)
        COMMON/FAT/BUF(512),SAMP(7)
        CALL FOPEN(19,"BN21GS")
        READ BINARY(19)IALLG
        CALL FCLOS(19)
        CALL FOPEN(19,'BNFES')
        READ BINARY(19)ALLE
        DO 10 I=1,896
   10   ALLE(I)=ALLE(I)/2
        CALL FOPEN(15,"LEOD")
        CALL FOPEN(13,"ESOD")
        CALL FOPEN(21,"IMRECONID")
        READ(21)NPEL,IFL,ILL
        CALL DFILW("IMROD",IER)
        CALL CFILW("IMROD",2,IER)

C INITIALIZE ADDRESSES FOR THE RECONSTRUCTION PROCESS.
        CALL BIMR(NPEL,IBUF(1),IALLG(1),BUF(1),ALLE(1))

        DO 200 L=IFL,ILL
C INITIALIZE CUMULATIVE BUFFER.
        CALL INBUF

 0300   READ(13)PE,IE1,A,AMF
        IF(PE.EQ.0.0)GO TO 0100

C INSERT AN E IN THE RECONSTRUCTED IMAGE LINE.
        CALL EINS(PE,IE1,A,AMF)
        GO TO 0300

 0100   READ(15)IPG,IR1,A,AMF
        IF(IPG.EQ.0)GO TO 100
C INSERT AN G IN THE RECONSTRUCTED IMAGE LINE.
        CALL GINS(IPG,IR1,A,AMF)
        GO TO 0100

  100   READ(15)SAMP
        IPG=-20
        DO 110 I=1,6
        IPG=IPG+21
        AMF=SAMP(I)/255.0
C INSERT A FAT-G IN THE RECONSTRUCTED IMAGE LINE.
  110   CALL GINS(IPG,18,1.0,AMF)
        IPG=128
```

```
        AMF=SAMP(7)/255.0
        CALL GINS(IPG,18,1.0,AMF)
C WRITE THE RECONSTRUCTED IMAGE LINE.
        CALL BFXW
  200   TYPE "IMRECON",L

        STOP
        END
```

```
; ********************SUBROUTINE : BIMR; CALLED BY IMRECON*****************

; INITIALIZES THE INVOLVED ADDRESSES AND VARIABLES FOR THE RECONSTRUCTION
; PROCESS.  ALSO OPENS THE OUTPUT FILE FOR THE RECONSTRUCTED IMAGE.


          .TITL               BIMR
          .ENT                BIMR
          .TXTM               1
          .EXTD               .CPYL,.FRET
          .ENT                NPEL,NPMID,AIBUF,AALLG,ABUF,AALLE
ARG1=-167
ARG2=ARG1+1
ARG3=ARG2+1
ARG4=ARG3+1
ARG5=ARG4+1
          .ZREL

NPEL:     0
NPMID:    0
AIBUF:    0
ABUF:     0
AALLG:    0
AALLE:    0
OUTF:     .+1*2
          .TXT "IMROD"

          .NREL
          4
BIMR:     JSR                 @.CPYL
          LDA                 3,FSP
          LDA                 1,@ARG1,3
          STA                 1,NPEL
          MOVZR               1,1
          INC                 1,1
          STA                 1,NPMID
          LDA                 1,ARG2,3
          STA                 1,AIBUF
          LDA                 1,ARG3,3
          STA                 1,AALLG
          LDA                 1,ARG4,3
          STA                 1,ABUF
          LDA                 1,ARG5,3
          STA                 1,AALLE
          LDA                 0,OUTF
          SUB                 1,1
          .SYSTM
          .OPEN 4
          JMP                 ER
```

```
        JSR             @.FRET

ER:     .SYSTM
        .ERTN
.END
```

```
; **********************SUBROUTINE : INBUF; CALLED BY IMRECON*****************

; INITIALIZES THE CUMULATIVE BUFFER.


          .TITL               INBUF
          .ENT                INBUF
          .EXTD               .CPYL,.FRET
          .EXTD               NPEL,ABUF,C0,C1

          .NREL
          0
INBUF:    JSR                 @.CPYL
          LDA                 2,NPEL
          MOVZL               2,2
          STA                 2,NPL2
          LDA                 2,ABUF
          LDA                 3,C1
          SUB                 3,2
          STA                 2,22
          LDA                 0,C0
CIBUF:    STA                 0,@22
          DSZ                 NPL2
          JMP                 CIBUF
          JMP                 BACK
NPL2:     0

BACK:     JSR                 @.FRET
.END
```

```
; **********************SUBROUTINE : GINS; CALLED BY IMRECON*****************


; INSERTS A G WITH THE SPECIFIED POSITION, WIDTH, AND AMPLITUDE IN THE
; RECONSTRUCTED IMAGE LINE.



          .TITL               GINS
          .ENT                GINS
          .ENT                TNPEL,C1,C0,AHALF,AF1,C2
          .EXTD               .CPYL, .FRET
          .EXTD               NPEL,NPMID,AIBUF,AALLG,ABUF
          .EXTN               GLFX1,GXFL1
ARG1=-167
ARG2=ARG1+1
ARG3=ARG2+1
ARG4=ARG3+1
          .ZREL
TNPEL:    0
C2:       2
C6:       6
AF1:      .+1
F1:       040420
          0                   ; FLOATING POINT 1.0.
C1:       1
C0:       0
AHALF:    .+1
HALF:     040200
          0                   ; 0.5.


          .NREL
          4
GINS:     JSR                 @.CPYL
          LDA                 3,FSP
          LDA                 1,@ARG1,3
          STA                 1,IPG
          LDA                 1,@ARG2,3
          STA                 1,IRI
          LDA                 1,ARG3,3
          .FLDS               1
          LDA                 0,AA
          .FSRS               0
          .FMFT
          LDA                 0,AF1
          .FLDS               0
          .FSTS
          LDA                 0,AB
          .FSRS               0
```

```
        LDA     1,ARG4,3
        .FLDS   1
        LDA     0,AAMF
        .FSRS   0

        LDA     1,ABUF
        STA     1,RABUF

        LDA     2,NPEL
        LDA     3,C1
        LDA     0,AALLG
        SUB     3,0
        LDA     1,IRI
        MUL
        STA     1,23
        SUB     2,1
        STA     1,21
        LDA     1,AIBUF
        SUB     3,1
        STA     1,22

        LDA     2,NPEL
        STA     2,TNPEL
        JMP     D030

IPG:    0
IRI:    0
AA:     .+1
        .BLK 2
AB:     .+1
        .BLK 2
AAMF:   .+1
        .BLK 2

D030:   LDA     2,@21
        STA     2,ITEMP
        GXFL1                   ; THIS SUBROUTINE AND GLFX1 DESTROY THE CONTENTS OF ALL
        ITEMP                   ; OF THE ACCUMULATORS.
        LDA     2,AA
        .FMS    2
        .FMFT
        LDA     2,@23
        STA     2,ITEMP
        GXFL1
        ITEMP
        LDA     2,AB
        .FMS    2
        .FATS
        LDA     2,AHALF
        .FAS    2
        GLFX1
```

```
          ITEMP
          LDA       2,ITEMP
          STA       2,@22
          DSZ       TNPEL
          JMP       D030
          JMP       B70
ITEMP:    1750
IC:       0
RABUF:    0

B70:      LDA       2,NPEL
          STA       2,TNPEL
          LDA       2,IPG
          LDA       3,NPMID
          SUB       2,3
          STA       3,IC
D070:     LDA       2,IC
          INC       2,2
          STA       2,IC
          LDA       1,C1
          SUBZL#    1,2,SZC
          JMP       E70
          LDA       3,NPEL
          INC       3,3
          SUBZL#    3,2,SNC
          JMP       E70
          SUB       1,2
          LDA       3,AIBUF
          ADD       2,3
          LDA       2,0,3
          STA       2,ITEMP
          GXFL1
          ITEMP
          LDA       2,AAMF
          .FMS      2
          LDA       2,RABUF
          .FAS      2
          .FSRS     2
E70:      LDA       2,RABUF
          INC       2,2
          INC       2,2
          STA       2,RABUF
          DSZ       TNPEL
          JMP       D070


BACK:     JSR       @.FRET
ER:       .SYSTM
          .ERTN
.END
```

```
; ********************SUBROUTINE : EINS; CALLED BY IMRECON*****************

; INSERTS AN E WITH THE SPECIFIED POSITION, WIDTH, AND AMPLITUDE IN THE
; RECONSTRUCTED IMAGE LINE.


          .TITL               EINS
          .ENT                EINS
          .EXTD               TNPEL,C1,C0,AHALF,AF1,C2
          .EXTD               .CPYL, .FRET
          .EXTD               NPEL,NPMID,AIBUF,AALLE,ABUF
          .EXTN               GLFX1,GXFL1
ARG1=-167
ARG2=ARG1+1
ARG3=ARG2+1
ARG4=ARG3+1


          .NREL
          4
EINS:     JSR                 @.CPYL
          LDA                 3,FSP
          LDA                 1,ARG1,3
          .FLDS               1
          .FMFT
          LDA                 1,AHALF
          .FLDS               1
          .FATS
          GLFX1
          ITEMP
          LDA                 1,ITEMP
          STA                 1,IPE
          LDA                 3,FSP
          LDA                 1,@ARG2,3
          STA                 1,IEI
          LDA                 1,ARG3,3
          .FLDS               1
          LDA                 0,AA
          .FSRS               0
          .FMFT
          LDA                 0,AF1
          .FLDS               0
          .FSTS
          LDA                 0,AB
          .FSRS               0

          LDA                 1,ARG4,3
          .FLDS               1
          LDA                 0,AAMF
```

```
          .FSRS           0

          LDA             1,ABUF
          STA             1,RABUF

          LDA             2,NPEL
          MOVZL           2,2
          LDA             0,AALLE
          LDA             1,IEI
          MUL
          STA             1,RAE2
          SUB             1,2
          NEG             2,0
          STA             0,RAE1
          LDA             2,AIBUF
          LDA             3,C1
          SUB             3,2
          STA             2,22

          LDA             2,NPEL
          STA             2,TNPEL
          JMP             D030

IPE:      0
IEI:      0
RAE1:     0
RAE2:     0
AA:       .+1
          .BLK 2
AB:       .+1
          .BLK 2
AAMF:     .+1
          .BLK 2

D030:     .FLDS           0
          LDA             2,AA
          .FMS            2
          .FMFT
          .FLDS           1
          LDA             2,AB
          .FMS            2
          .FATS
          LDA             2,AHALF
          .FAS            2
          GLFX1
          ITEMP
          LDA             2,ITEMP
          STA             2,@22
          LDA             2,C2
          LDA             0,RAE1
          ADD             2,0
```

```
        STA     0,RAE1
        LDA     1,RAE2
        ADD     2,1
        STA     1,RAE2
        DSZ     TNPEL
        JMP     D030
        JMP     B70
ITEMP:  0
IC:     0
RABUF:  0

B70:    LDA     2,NPEL
        STA     2,TNPEL
        LDA     2,IPE
        LDA     3,NPMID
        SUB     2,3
        STA     3,IC
D070:   LDA     2,IC
        INC     2,2
        STA     2,IC
        LDA     1,C1
        SUBZL#  1,2,SZC
        JMP     E70
        LDA     3,NPEL
        INC     3,3
        SUBZL#  3,2,SNC
        JMP     E70
        SUB     1,2
        LDA     3,AIBUF
        ADD     2,3
        LDA     2,0,3
        STA     2,ITEMP
        GXFL1
        ITEMP
        LDA     2,AAMF
        .FMS    2
        LDA     2,RABUF
        .FAS    2
        .FSRS   2
E70:    LDA     2,RABUF
        INC     2,2
        INC     2,2
        STA     2,RABUF
        DSZ     TNPEL
        JMP     D070


BACK:   JSR     @.FRET
ER:     .SYSTM
        .ERTN
```

```
.END
```

```
; ********************SUBROUTINE : BFXW; CALLED BY IMRECON*****************

; CHECKS THE RECONSTRUCTED IMAGE LINE FOR OVER OR UNDERFLOW ( ) 255 OF ( 0),
; AND WRITES IT TO THE OUTPUT FILE OF THE RECONSTRUCTED IMAGE.


          .TITL       BFXW
          .TXTM       1
          .ENT        BFXW
          .EXTD       .CPYL, .FRET
          .EXTN       GLFX1,GXFL1
          .EXTD       NPEL,TNPEL,AIBUF,ABUF,C1,C0,AHALF

          .NREL
          0
BFXW:     JSR         @.CPYL
          LDA         3,C1
          LDA         1,AIBUF
          SUB         3,1
          STA         1,22
          LDA         1,ABUF
          STA         1,RABUF

          LDA         2,NPEL
          STA         2,TNPEL

D080:     LDA         2,RABUF
          .FLDS       2
          INC         2,2
          INC         2,2
          STA         2,RABUF
          LDA         2,AHALF
          .FAS        2
          GLFX1
          ITEMP
          LDA         2,ITEMP
          STA         2,@22
          DSZ         TNPEL
          JMP         D080
          JMP         B40

ITEMP:    0
RABUF:    0
C255:     377

B40:      LDA         3,C1
          LDA         1,AIBUF
          SUB         3,1
          STA         1,21
```

```
        STA     1,22
        LDA     2,NPEL
        STA     2,TNPEL
        LDA     0,C0
        LDA     3,C255
D040:   LDA     2,@21
        MOVZL#  2,2,SNC         ; IF(IBUF(I).LT.0)
        JMP     .+3
        STA     0,@22           ;*IBUF(I)=0
        JMP     E40

        SUBZL   3,2,SZC         ; IF(IBUF(I).GT.255)
        JMP     .+3
        STA     3,@22           ;*IBUF(I)=255
        JMP     E40
        LDA     1,@22           ; THIS DUMMY INSTRUCTION IS JUST TO INCREMENT 22.
E40:    DSZ     TNPEL
        JMP     D040
        JMP     WRITE


WRITE:  LDA     1,NPEL
        MOVZL   1,1
        LDA     0,AIBUF
        MOVZL   0,0
        .SYSTM
        .WRS 4
        JMP     ER


BACK:   JSR     @.FRET
ER:     .SYSTM
        .ERTN
.END
```

APPENDIX 3

COMPUTER PROGRAMS ASSOCIATED WITH CHAPTER 4

```
C
C
C                       **** EDGDIS3.FR ****
C
C       GENERATES EDGE STROKES USING AN EVENT DISTANCE MEASURE
C       THIS VERSION DOES NOT PRINT OUT THE STROKE INFORMATION
C
C       THE LOOKUP TABLE OF EDGE WIDTHS IS IN "WIDTHS"
C       THE LIST OF EDGES FROM "ESOD" IS IN "EDGES"
C       THE STROKED EDGE EVENTS ARE PUT IN "NEWEDGES"
C       THE CONSTANTS ARE READ FROM "CONSTANTS"
C
        REAL WIDTH(20),X(800),Z(800),A(800),W(800)
        REAL XS(800),AS(800),WS(800)
        REAL E(800),KT
        INTEGER PTR(800),Y(200),N(800),BY(800)
        INTEGER SED(500),SEU(500)
        INTEGER LS(800),HL(10),YS(800)
C       EDGE WIDTH FILE USED TO BE CALLED WIDTHS 6/5/83
        CALL FOPEN(20,"WIDTHS2")
        CALL FOPEN(21,"EDGES")
        CALL FOPEN(22,"NEWEDGES")
        CALL FOPEN(23,"CONSTANTS2")
C
C       HERE ARE THE IMPORTANT CONSTANTS
C
C       EP1 IS THE THRESHOLD FOR THE FIRST STROKING PASS
C       EP2 IS THE THRESHOLD FOR THE SECOND STROKING PASS
C       C1 IS THE CONSTANT FOR WIDTH IN THE DISTANCE FORMULA
C       C2 IS THE CONSTANT FOR AMPLITUDE IN THE DISTANCE FORMULA
C       ELIM IS THE FRACTION OF EDGE EVENTS USED BEFORE GOING TO A SECOND PASS
C       Q DETERMINES THE SHAPE OF THE CASSINI DISTANCE FUNCTION
C
        READ(23)EP1,EP2,C1,C2,ELIM,Q,BRK,L2
        TYPE "HAVE READ IN THE CONSTANTS"
        QS=Q*Q
        QF=QS*QS
C
C       GET THE DATA AND INTERPRET WIDTHS
C
        DO 1 K=1,7
1       READ(20) WIDTH(K)
        DO 5 J=1,1000
5       READ(21,END=15) X(J),N(J),Z(J),A(J)
15      L=J-1
        DO 20 I=1,L
        W(I)=0.
        IF (N(I).EQ.0) GO TO 20
        W(I)=WIDTH(N(I))*Z(I)+WIDTH(N(I)+1)*(1-Z(I))
20      CONTINUE
C
```

```
C       CALCULATE THE ENERGY AND SORT LS
C
        IRATE=1
        RMIN=1.0E10
        RMAX=0
        DO 21 I=1,L
        E(I)=A(I)*A(I)
        IF(E(I).LT.1.0E-6) GO TO 21
        IF(E(I).GT.RMAX) RMAX=E(I)
        IF(E(I).LT.RMIN) RMIN=E(I)
21      PTR(I)=0
C       COARSE LINEAR SORTING ALGORITHM
        RSTEP=(RMAX-RMIN)/9.99
        DO 25 J=1,10
        RLEVEL=RMAX-J*RSTEP
        DO 25 K=1,L
        IF(PTR(K).NE.0) GO TO 25
        IF(E(K).LT.RLEVEL) GO TO 25
        LS(IRATE)=K
        IRATE=IRATE+1
        PTR(K)=1
25      CONTINUE
C
C       SETUP Y, BY AND CLEAR PTR, SED, SEU
C
        K=1
        Y(1)=1
        DO 34 I=1,L
        BY(I)=K
        IX=IFIX(X(I)+.5)
        IF(IX.NE.0) GO TO 34
        K=K+1
        Y(K)=I+1
34      PTR(I)=0
        DO 35 I=1,500
        SEU(I)=0
35      SED(I)=0
        MAXY=K-1
        IFLAG=0
        EP=EP1
C
C       SETUP HL, THE SEARCHING HEMISPHERE
C
        HL(1)=9
        HL(2)=9
        HL(3)=9
        HL(4)=9
        HL(5)=8
        HL(6)=8
        HL(7)=7
        HL(8)=6
```

```
        HL(9)=4
        HL(10)=0
C       L1,L2 SET THE SIZE OF THE SEARCHING HEMISPHERE
        L1=1
C
C       SELECT NEXT ORIGIN OF STROKE
C       IPEN: HOW FAR DOWN IN LS WE ARE LOOKING
C       ISTR: THE STROKE NUMBER
C       ELIM: PERCENT LIMIT OF EVENTS BEFORE QUITTING
C       IMARK: NEXT THRESHOLD BEFORE EVENT CHECK
C
        IBOT=IFIX(ELIM*FLOAT(L-128))
        IPEN=1
        ISTR=0
        IMARK=IFIX(0.1*FLOAT(L-128))
        IINC=IMARK
36      IF(IPEN.LT.IMARK) GO TO 38
        ITOTAL=0
        DO 37 J=1,L
        IND=PTR(J)
        IF(IND.NE.0) ITOTAL=ITOTAL+1
 37      CONTINU
C       ALL DONE WITH EP1 WHEN ITOTAL>=IBOT
        IF(ITOTAL.GE.IBOT) GO TO 200
        IMARK=IMARK+IINC
38      IF(PTR(LS(IPEN)).EQ.0) GO TO 40
        IPEN=IPEN+1
         GO TO 3
40      ISTR=ISTR+1
        LSIP=LS(IPEN)
        TYPE "EP1 STROKE NUMBER ",ISTR
        SED(ISTR)=LS(IPEN)
        SEU(ISTR)=LS(IPEN)
        PTR(LS(IPEN))=-1
C
C       STROKE DOWNWARD
C       PTR(I) POINTS TO THE PREVIOUS EVENT OF STROKE
C       SED(I) POINTS TO THE DOWNWARD END OF STROKE
C
44      IY=BY(SED(ISTR))+1
50      IF (IY.GT.MAXY) GO TO 90
        ISRCH=Y(IY)
        DMIN=10.E6
C       NE IS THE EVENT NUMBER AT THE END OF STROKE
        NE=SED(ISTR)
C       IX IS THE X POSITION OF THE END OF STROKE
        IX=IFIX(X(NE)+0.5)
C       J IS THE NUMBER OF LINES DOWN FROM THE END OF THE STROKE
        DO 80 J=L1,L2
        LU=IX+HL(J)
        LL=IX-HL(J)
```

```
60      IF (PTR(ISRCH).NE.0) GO TO 70
        IF (X(ISRCH).GT.LU) GO TO 70
        IF (X(ISRCH).LT.LL) GO TO 70
C
C       GET THE DISTANCE
C
        XF=X(ISRCH)-IX
         YF=
        XSD=XF*XF
        YSD=YF*YF
C       CALCULATION OF THE CASSINI OVAL GEOMETRIC DISTANCE
        DIS=QS*(XSD-YSD)+SQRT(XSD*XSD+2*XSD*YSD*(1-QF)+YSD*YSD)
        DIS=DIS/((1-QF)*E(ISRCH))
C       CALCULATION OF WIDTH AND AMPLITUDE DISTANCES
        WD=C1*(W(ISRCH)-W(NE))**2/(W(ISRCH)+W(NE))**2
        AD=C2*(A(ISRCH)-A(NE))**2/(A(ISRCH)+A(NE))**2
        IF(A(ISRCH).GT.0.AND.A(NE).LT.0) AD=1E8
        IF(A(ISRCH).LT.0.AND.A(NE).GT.0) AD=1E8
C       TOTAL DISTANCE MEASURE
        DIS=DIS+WD+AD
        IF (DIS.GT.DMIN) GO TO 70
C       DMIN IS THE SMALLEST DISTANCE FOUND IN THE SEARCHING HEMISPHERE
        DMIN=DIS
C       NVNT IS THE NEXT EVENT TO BE CALLED THE END OF STROKE
        NVNT=ISRCH
        JMIN=J
70      ISRCH=ISRCH+1
        IF(IFIX(X(ISRCH)+0.5).EQ.0) GO TO 79
        GO TO 60
79      ISRCH=ISRCH+1
        IF (IY+J.GT.MAXY) GO TO 81
80      CONTINUE
81      IF(DMIN.GT.EP) GO TO 90
        IY=IY+JMIN
        PTR(NVNT)=SED(ISTR)
        SED(ISTR)=NVNT
        GO TO 50
C
C       STROKE UPWARD
C
90      IY=BY(SEU(ISTR))-1
95      IF (IY.LT.1) GO TO 210
        ISRCH=Y(IY)
        DMIN=10.E6
        NE=SEU(ISTR)
        IX=IFIX(X(NE)+0.5)
        DO 120 J=L1,L2
        LU=IX+HL(J)
        LL=IX-HL(J)
100     IF (PTR(ISRCH).NE.0) GO TO 110
        IF(X(ISRCH).GT.LU) GO TO 110
```

```
        IF(X(ISRCH).LT.LL) GO TO 110
C
C       GET THE DISTANCE
C
        XF=X(ISRCH)-FLOAT(IX)
        YF=J
        XSD=XF*XF
        YSD=YF*YF
C       CALCULATION OF THE CASSINI OVAL GEOMETRIC DISTANCE
        DIS=QS*(XSD-YSD)+SQRT(XSD*XSD+2*XSD*YSD*(1-QF)+YSD*YSD)
        DIS=DIS/((1-QF)*E(ISRCH))
C       CALCULATION OF THE WIDTH AND AMPLITUDE DISTANCES
        WD=C1*(W(ISRCH)-W(NE))**2/(W(ISRCH)+W(NE))**2
        AD=C2*(A(ISRCH)-A(NE))**2/(A(ISRCH)+A(NE))**2
        IF(A(ISRCH).GT.0.AND.A(NE).LT.0) AD=1E8
        IF(A(ISRCH).LT.0.AND.A(NE).GT.0) AD=1E8
C       TOTAL DISTANCE MEASURE
        DIS=DIS+WD+AD
        IF(DIS.GT.DMIN) GO TO 110
        JMIN=J
        DMIN=DIS
        NVNT=ISRCH
110     ISRCH=ISRCH+1
        IF(IFIX(X(ISRCH)+0.5).EQ.0) GO TO 119
        GO TO 100
119     IF (IY-J.LT.1) GO TO 121
        ISRCH=Y(IY-J)
120     CONTINUE
C       GET THE NEXT STROKE
121     IF(DMIN.GT.EP) GO TO 210
        IY=IY-JMIN
        PTR(NVNT)=SEU(ISTR)
        SEU(ISTR)=NVNT
        GO TO 95
C
C       REDUCE TO EP2 AND CONTINUE
C
200     EP=EP2
        TYPE "OK DOWN TO EP2"
C       IFLAG IS SET TO INDICATE THE SECOND PASS IS IN PROGRESS
        IFLAG=1
        ITOP=ISTR
        ISTR=0
210     IF(IFLAG.EQ.0) GO TO 36
        ISTR=ISTR+1
        IF (ISTR.GT.ITOP) GO TO 400
        TYPE "EP2 STROKE NUMBER ",ISTR
        GO TO 44

C       INTERPOLATES AND GENERATES A CORRECTION LIST
C
```

```
400     ISTR=ISTR-1
        TYPE "INTERPOLATE"
        K=0
        JKJ=0
        DO 700 I=1,ISTR
C       I IS THE STROKE NUMBER
C       IZ IS A FLAG FOR UP OR DOWN STROKING
        TYPE "I=",I
        IZ=1
        IFPTR=SEU(I)
        NPTR=PTR(IFPTR)
430     IFY=BY(IFPTR)
        K=K+1
        YS(K)=IFY
        XS(K)=X(IFPTR)
        AS(K)=A(IFPTR)
        WS(K)=W(IFPTR)
        IF (NPTR.EQ.-1) GO TO 460
        NY=BY(NPTR)
        IDY=IABS(NY-IFY)
        IF(IDY.EQ.1) GO TO 440
C       INTERPOLATES WHEN THERE ARE GAPS
        DX=(X(NPTR)-X(IFPTR))/FLOAT(IDY)
        DA=(A(NPTR)-A(IFPTR))/FLOAT(IDY)
        DW=(W(NPTR)-W(IFPTR))/FLOAT(IDY)
        IDYM=IDY-1
        DO 420 J=1,IDYM
        K=K+1
        XS(K)=X(IFPTR)+FLOAT(J)*DX
        AS(K)=A(IFPTR)+FLOAT(J)*DA
        WS(K)=W(IFPTR)+FLOAT(J)*DW
420     YS(K)=-IFY-FLOAT(IZ*J)
440     IFPTR=NPTR
        NPTR=PTR(NPTR)
        GO TO 430
460     IF(IZ.EQ.-1) GO TO 470
        IZ=-1
        IFPTR=SED(I)
        NPTR=PTR(IFPTR)
        KM=K
        GO TO 430
470     KH=(K-KM+1)/2
        DO 475 J=1,KH
        KT=YS(KM+J-1)
        YS(KM+J-1)=YS(K-J+1)
        YS(K-J+1)=KT
        KT=XS(KM+J-1)
        XS(KM+J-1)=XS(K-J+1)
        XS(K-J+1)=KT
        KT=AS(KM+J-1)
        AS(KM+J-1)=AS(K-J+1)
```

```
        AS(K-J+1)=KT
        KT=WS(KM+J-1)
        WS(KM+J-1)=WS(K-J+1)
        WS(K-J+1)=KT
475     CONTINUE
        YS(K)=0.
        XS(K)=0.
        AS(K)=0.
        WS(K)=0.
C
C       SPLITTING ROUTINE
C
C       COMPARES NUMBER OF ORIGINAL POINTS TO TOTAL IN EACH STROKE
C       IF THE RATIO EXCEEDS "ACCEPT" THEN THE STROKE IS SPLIT
C       INTO TWO SEPARATE STROKES.
C
        LMAX=0
        LGAP=0
        LTOT=0
        L=0
        I2=0
500     DO 550 J=1,K
        IF(YS(J).EQ.0) GO TO 520
        IF(YS(J).GT.0) GO TO 510
        L=L+1
        LGAP=LGAP+1
        IF(L.LE.LMAX) GO TO 505
        LMAX=L
        KMAX=J-L
505     LTOT=LTOT+1
        GO TO 550
510     L=0
        LTOT=LTOT+1
        GO TO 550
520     JJ=J-1
        IF(YS(JJ).EQ.0) GO TO 550
        RATIO=FLOAT(LGAP)/FLOAT(LTOT)
        IF(RATIO.LE.BRK) GO TO 540
        DO 530 M=1,LMAX
        MM=M+KMAX
        TYPE "Y,X,A,W=",YS(MM),XS(MM),AS(MM),WS(MM)
        YS(MM)=0
        XS(MM)=0
        AS(MM)=0
530     WS(MM)=0
        ISPLIT=1
        TYPE "SPLIT!  K=",K,"   LMAX=",LMAX,"   KMAX=",KMAX
540     LMAX=0
        LGAP=0
        LTOT=0
550     CONTINUE
```

```
      IF(ISPLIT.EQ.0) GO TO 560
      ISPLIT=0
      GO TO 500
C
C     OUTPUT THE CORRECT ARRAYS
C
560   DO 600 II=1,K
      IF(YS(II).NE.0) GO TO 580
      IF(IZ.NE.0) GO TO 600
      IZ=1
      GO TO 590
580   IZ=0
590   YS(II)=IABS(YS(II))
      WRITE(22) YS(II),",",XS(II),",",AS(II),",",WS(II)
600   CONTINUE
      JKJ=JKJ+K
700   K=0
      TYPE "THE FILE LENGTH IS:",JKJ
      STOP
      END
```

```
C       EDGCOD2A.FR
C
C
C       ****CODE EDGE STROKES****
C
C       THE GROUPED EDGE EVENTS FOUND IN THE INPUT FILE, "NEWEDGES",
C       ARE FITTED WITH SECOND ORDER CURVES USING THE SUBROUTINE
C       "CRVFIT".  THIS IS DONE FOR POSITION, WIDTH AND AMPLITUDE.
C       THE STROKES ARE THEN CODED AND PUT INTO THE OUTPUT FILE,
C       "EDGCODES2".  THE ERROR BOUNDS USED IN THE FITTING ROUTINE
C       ARE FOUND IN THE FILE EDGBOUNDS2.  ANY STROKE CONTAINING
C       LESS THAN FOUR EVENTS IS THROWN AWAY.
C
        REAL X(800),W(800),A(800),SLOPE(100),ACCEL(100),CODE(500)
        INTEGER LENGTH(100),Y(800)
        INTEGER ST,NST,ISTR
        REAL IT(200),IH(200),IR(200),ISR(200)
        REAL JO(200),JT(200),JH(200),JR(200),JSR(200)
        CALL FOPEN(20,"NEWEDGES")
        CALL FOPEN(30,"EDGBOUNDS2")
        CALL FOPEN(40,"EDGCODES2")
        READ(30)EX,EA,EW
        DO 1 I=1,1000
1       READ(20,END=2)Y(I),X(I),A(I),W(I)
2       L=I-1
C
C       CALCULATE SUMMATIONS FOR THE MEAN SQUARED ERROR FIT.
C
        IT(1)=1.
        IH(1)=1.
        IR(1)=1.
        ISR(1)=1.
        JO(1)=0
        JT(1)=0
        JH(1)=0
        JR(1)=0
        JSR(1)=0
        DO 5 I=2,128
        JO(I)=JO(I-1)+FLOAT(I-1)
        JT(I)=JT(I-1)+FLOAT(I-1)**2
        JH(I)=JH(I-1)+FLOAT(I-1)**3
        JR(I)=JR(I-1)+FLOAT(I-1)**4
        IT(I)=IT(I-1)+FLOAT(I)**2
        IH(I)=IH(I-1)+FLOAT(I)**3
        IR(I)=IR(I-1)+FLOAT(I)**4
        XO=JO(I)
        XT=JT(I)
        XH=JH(I)
        XR=JR(I)
        JSR(I)=I*XT*XR+XO*XT*XH+XO*XT*XH-I*XH*XH-XO*XO*XR-XT*XT*XT
5       ISR(I)=IT(I)*IR(I)-IH(I)**2
```

```
C
C       BEGIN CODING
C
        ST=1
        NST=1
        M=1
10      CALL CRVFIT(ST,NST,X,EX,N,SLOPE,ACCEL,LENGTH,ISR,IT,IH,IR,LLL)
        IF (NST-4.GT.ST) GO TO 15
        GO TO 45
15      CODE(M)=Y(ST)
        CODE(M+1)=X(ST)
        MA=M+2
        CODE(M+3)=W(ST)
        M=M+1
C
C       CODE X
C
        DO 20 J=1,N
        M=M+3
        CODE(M)=LENGTH(J)
        CODE(M+1)=SLOPE(J)
20      CODE(M+2)=ACCEL(J)
        CODE(M+3)=0
        M=M+1
C
C       CODE A
C
        CALL AMPFIT(ST,NST,A,A0,B0,C0,JSR,J0,JT,JH,JR,LLL)
        CODE(MA)=C0
        M=M+3
        CODE(M)=LLL-1
        CODE(M+1)=B0
        CODE(M+2)=A0
        CODE(M+3)=0
        M=M+1
C
C       CODE W
C
        CALL CRVFIT(ST,NST,W,EW,N,SLOPE,ACCEL,LENGTH,ISR,IT,IH,IR,LLL)
        DO 40 J=1,N
        M=M+3
        CODE(M)=LENGTH(J)
        CODE(M+1)=SLOPE(J)
40      CODE(M+2)=ACCEL(J)
        CODE(M+3)=0
        M=M+4
45      ST=NST
C
C       DO NEXT STROKE
C
        IF (ST.LE.L) GO TO 10
```

```
      M=M-1
      DO 50 I=1,M
50    WRITE(40)CODE(I)
      M=M-1
      STOP
      END
```

```
C       CRVFIT.FR
C
C
C       ****SECOND ORDER CURVE FITTING SUBROUTINE****
C
C
C       CALLED BY CRVFIT(ST,NST,VAR,E,N,SLOPE,ACCEL,LENGTH,ISR,IT,IH,IR,L)
C       WHERE "ST" IS THE STARTING POINTER TO THE "VAR" ARRAY, "NST" IS
C       A RETURNED VALUE WHICH POINTS TO THE NEXT STARTING LOCATION IN
C       THE ARRAY "VAR" IS THE ARRAY CONTAINING THE DATA, "E" IS THE
C       ERROR BOUND, "N" IS A RETURNED VALUE EQUAL TO THE NUMBER OF
C       CURVE SEGMENTS, "SLOPE" IS A RETURNED ARRAY CONTAINING THE INITIAL
C       SLOPE OF EACH SEGMENT, "ACCEL" IS A RETURNED ARRAY CONTAINING
C       THE CHANGE IN SLOPE OF EACH SEGMENT, "LENGTH" IS A RETURNED ARRAY
C       CONTAINING THE LENGTH OF EACH SEGMENT, "ISR", "IT", "IH", AND
C       "IR" ARE NEEDED FOR THE MEAN SQUARED ERROR FIT, L IS THE TOTAL
C       NUMBER OF POINTS IN THE STROKE.
C
C
C
      SUBROUTINE CRVFIT(ST,NST,VAR,E,N,SLOPE,ACCEL,LENGTH,ISR,IT,IH,IR,L)
      REAL VAR(2800),Y(200),SLOPE(128),ACCEL(128),E,C,CO
      INTEGER LENGTH(128),N,ST,IX,OX,K,M,PT
      REAL IT(128),IH(128),IR(128),ISR(128)
C
C       GENERATE ARRAY Y
C
      PT=ST
      DO 1000 I=1,128
       Y(I)=VAR(PT
      PT=PT+1
      IF(ST.EQ.NST) GO TO 999
      IF(I.GE.L) GO TO 500
      GO TO 1000
999   IF(IFIX(VAR(PT)*1E5+.01).EQ.0)GO TO 500
1000  CONTINUE
      I=128
500   L=I
      NST=PT+1
C
C       CONTINUE ONLY IF THE LENGTH IS MORE THAN TWO.

      IF (NST-2.LE.ST) GO TO 111
      IF (NST-3.GT.ST) GO TO 222
      N=1
      LENGTH(1)=1
      SLOPE(1)=Y(2)-Y(1)
      GO TO 111
C
C       CURVE FIT
C
```

```
C       N IS THE NUMBER OF SEGMENTS USED, K IS THE STARTING POINT OF
C       THE CURRENT SEGMENT, M IS THE DISTANCE INTO EACH SEGMENT,
C       C IS THE CALCULATED SLOPE, AND A IS THE CALCULATED SECOND
C       DERIVATIVE.
C
222     K=1
        N=1
        OX=1
        OY=Y(1)
40      M=2
 50      C=0
        A=0.
        DO 10 I=1,M
        C=C+(Y(I+OX)-OY)*(FLOAT(I)*IR(M)-IH(M)*FLOAT(I)**2)
10      A=A+(Y(I+OX)-OY)*(FLOAT(I)**2*IT(M)-IH(M)*FLOAT(I))
        C=C/ISR(M)
        A=A/ISR(M)
C
C       MEASURE THE DISTANCE FROM THE FITTED CURVE TO EACH POINT
C       IF ANY POINT IS FARTHER AWAY THAN THE ERROR BOUND, E, THEN
C       THE SEGMENT IS BROKEN AT THE PREVIOUS VALUE.
C
        DO 20 I=1,M
        D=ABS(C*FLOAT(I)+A*FLOAT(I)**2+OY-Y(I+OX))
        IF(D.GT.E)GO TO 100
20      CONTINUE
        IX=M
        CO=C
        AO=A
        M=M+1
        IF(K+M.GT.L)GO TO 200
        GO TO 50
C
C       STORE SLOPE AND LENGTH OF SEGMENT N
C
100     IF (M.GT.2) GO TO 125
        CO=C
        AO=A
        IX=M
125     SLOPE(N)=CO
        ACCEL(N)=AO
        LENGTH(N)=IX
        K=K+IX
        OX=K
        OY=OY+CO*FLOAT(IX)+AO*FLOAT(IX**2)
        IF(K.GE.L) GO TO 111
        N=N+1
        IF(K+1.EQ.L) GO TO 150
        GO TO 40
150     CO=Y(L)-Y(K)
        AO=0
```

```
      IX=1
C
C     STORE SLOPE AND LENGTH OF LAST SEGMENT AND RETURN
C
200   SLOPE(N)=CO
      ACCEL(N)=AO
      LENGTH(N)=IX
111   RETURN
      END
```

```
C       AMPFIT.FR
C
C
C       ****SECOND ORDER CURVE FITTING SUBROUTINE****
C
C
C       CALLED BY AMPFIT(ST,NST,VAR,A0,B0,C0,ISR,I0,IT,IH,IR,L)
C       WHERE "ST" IS THE STARTING POINTER TO THE "VAR" ARRAY, "NST" IS
C       A RETURNED VALUE WHICH POINTS TO THE NEXT STARTING LOCATION IN
C       THE ARRAY "VAR" IS THE ARRAY CONTAINING THE DATA, A0, B0, AND
C       C0 ARE THE RETURNED POLYNOMIAL COEFFICIENTS, "ISR","I0", "IT",
C       "IH", AND "IR" ARE NEEDED FOR THE MEAN SQUARED ERROR FIT, L IS
C       THE TOTAL NUMBER OF POINTS IN THE STROKE.
C
C
C
      SUBROUTINE AMPFIT(ST,NST,VAR,A0,B0,C0,ISR,I0,IT,IH,IR,L)
      REAL VAR(2800),Y(200)
      INTEGER ST,PT
      REAL I0(128),IT(128),IH(128),IR(128),ISR(128)
      REAL J0,JT,JH,JR
C
C     GENERATE ARRAY Y
C
      PT=ST
      DO 1000 I=1,128
       Y(I)=VAR(PT
      PT=PT+1
      IF(ST.EQ.NST) GO TO 999
      IF(I.GE.L) GO TO 500
      GO TO 1000
999   IF(IFIX(VAR(PT)*1E5+.01).EQ.0)GO TO 500
1000  CONTINUE
      I=128
500   L=I
      NST=PT+1
C
C     CONTINUE ONLY IF THE LENGTH IS MORE THAN TWO.

      IF (NST-2.LE.ST) GO TO 111
      IF (NST-3.GT.ST) GO TO 222
      A0=0
      B0=Y(2)-Y(1)
      C0=Y(1)
      GO TO 111
C
C     CURVE FIT
C
222   YI=0.
      YX=0.
      YXS=0.
```

```
      DO 10 I=1,L
      YI=YI+Y(I)
      YX=YX+Y(I)*FLOAT(I-1)
10    YXS=YXS+Y(I)*FLOAT((I-1)*(I-1))
      JO=IO(L)
      JT=IT(L)
      JH=IH(L)
      JR=IR(L)
      LL=FLOAT(L)
      CO=(YI*(JT*JR-JH*JH)+YX*(JH*JT-JO*JR)+YXS*(JO*JH-JT*JT))/ISR(L)
      BO=(YI*(JT*JH-JO*JR)+YX*(LL*JR-JT*JT)+YXS*(JO*JT-LL*JH))/ISR(L)
      AO=(YI*(JO*JH-JT*JT)+YX*(JO*JT-LL*JH)+YXS*(LL*JT-JO*JO))/ISR(L)
111   RETURN
      END
```

```
C
C
C       ***** REALREC2.FR *****
C
C
C
      REAL X(1000),W(1000),A(1000)
      INTEGER CH(1000),TOP(1000),Y(1000)
      REAL XP(1000),ZP(1000),AP(1000),WIDTH(20)
      INTEGER NP(1000)
14    FORMAT(2X,F6.1,',  ',I3,',  ',F7.3',  ',F7.3)
      CALL FOPEN(20,"WIDTHS2")
      CALL FOPEN(21,"EDGCODES2")
      CALL FOPEN(22,"BH10D")
      J=1
      DO 1 I=1,1000
      IF(I.GT.128) GO TO 1
      TOP(I)=0
1     CH(I)=0
2     READ(21,END=50) YR,X(J),A(J),W(J)
      Y(J)=YR+0.01
      CH(J)=TOP(Y(J))
      TOP(Y(J))=J
      JSAVE=J
3     READ(21)XL
      JINT=J
      IXL=IFIX(XL+0.001)
      IF(IXL.EQ.0) GO TO 5
      READ(21)XS
      READ(21)XA
      DO 4 I=1,IXL
      J=J+1
4     X(J)=X(JINT)+XS*I+XA*I*I
      GO TO 3
5     J=JSAVE
6     READ(21)AL
      JINT=J
      IAL=IFIX(AL+0.001)
      IF(IAL.EQ.0) GO TO 8
      READ(21)AS
      READ(21)AA
      DO 7 I=1,IAL
      J=J+1
7     A(J)=A(JINT)+AS*I+AA*I*I
      GO TO 6
8     J=JSAVE
9     READ(21)WL
      JINT=J
      IWL=IFIX(WL+0.001)
      IF(IWL.EQ.0) GO TO 11
      READ(21)WS
```

```
        READ(21)WA
        DO 10 I=1,IWL
        J=J+1
10      W(J)=W(JINT)+WS*I+WA*I*I
        GO TO 9
11      IS=JSAVE+1
        INIT=Y(JSAVE)
        DO 12 I=IS,J
        Y(I)=INIT+I-IS+1
        CH(I)=TOP(Y(I))
12      TOP(Y(I))=I
        J=J+1
        GO TO 2
50      J=J-1
C
C       PUT THE DATA INTO "ESOD"'S FORM
C
        DO 51 I=1,7
51      READ(20)WIDTH(I)
        K=0
        L=0
52      K=K+1
        IPTR=TOP(K)
53      L=L+1
        TYPE K,L
        IF(IPTR.EQ.0) GO TO 60
        XP(L)=X(IPTR)
        AP(L)=A(IPTR)
        IF(W(IPTR).LT.WIDTH(1))W(IPTR)=WIDTH(1)
        IF(W(IPTR).GT.WIDTH(7))W(IPTR)=WIDTH(7)
        DO 54 I=1,5
        IF(W(IPTR).LT.WIDTH(I+1)) GO TO 55
54      CONTINUE
        I=6
55      NP(L)=I
        ZP(L)=(W(IPTR)-WIDTH(I+1))/(WIDTH(I)-WIDTH(I+1))
        IPTR=CH(IPTR)
        GO TO 53
60      XP(L)=0.
        NP(L)=0.
        ZP(L)=0.
        AP(L)=0.
        IF(K.LT.128) GO TO 52
C
C       NOW SPIT IT OUT
C
        DO 70 I=1,L
        XP(I)=0.5*IFIX(2*XP(I)+0.5)
        IF(AP(I).GT.1.0)AP(I)=1.0
        IF(AP(I).LT.-1.0)AP(I)=-1.0
        IF(XP(I).LT.0.5)XP(I)=0.5
```

```
      IF(XP(I).GT.128)XP(I)=128
      IF(NP(I).EQ.0)XP(I)=0.
      WRITE(22,14) XP(I),NP(I),ZP(I),AP(I)
70    CONTINUE
      STOP
      END
```

```
C       HYBRID.FR
C
C       **** CREATE HYBRID EDGE LIST ****
C
C
C       THE OLD LIST GENERATED BY "ES" IS PUT INTO "OLD" FROM
C       THE FILE "EDGES".  THE NEW LIST FROM CODING IS PUT INTO
C       "NEW" FROM THE FILE "BH10D".  AS THE HYBRID LIST IS
C       GENERATED, IT IS PUT INTO "MIX", AND THEN WRITTEN
C       OUT TO THE FILE "MIXED"
C
C
        REAL MIXX(800),MIXZ(800),MIXA(800)
        REAL NEWX(800),NEWZ(800),NEWA(800)
        REAL OLDX(800),OLDZ(800),OLDA(800)
        INTEGER NEWN(800),OLDN(800),MIXN(800)
        CALL FOPEN(20,"EDGES")
        CALL FOPEN(21,"BH10D")
        CALL FOPEN(22,"MIXED")
        DO 1 I=1,800
1       READ(20,END=11) OLDX(I),OLDN(I),OLDZ(I),OLDA(I)
11      DO 12 J=1,800
12      READ(21,END=13) NEWX(J),NEWN(J),NEWZ(J),NEWA(J)
13      JNF=1
        JOF=1
        JSAVE=2
        MAXN=J-1
        J=1
        VAL=10.
2       X=NEWX(JNF)
        IM=IFIX(X)
        IF(IM.EQ.0) GO TO 8
3       IT=IFIX(OLDX(J))
        IF(IT.EQ.0) GO TO 5
        DIF=ABS(X-OLDX(J))
        IF(DIF.GT.VAL) GO TO 4
        IBEST=J
        VAL=DIF
4       J=J+1
        GO TO 3
5       IF(VAL.GT.2.1) GO TO 6
C
C       OLD EVENT:  USE THE OLD VALUE FOR POSITION AND WIDTH AND THE
C       NEW VALUE FOR AMPLITUDE.
C
        VAL=10.
        MIXX(JNF)=OLDX(IBEST)
        MIXZ(JNF)=OLDZ(IBEST)
        MIXN(JNF)=OLDN(IBEST)
        MIXA(JNF)=NEWA(JNF)
        GO TO 7
```

```
C
C       NEW EVENT:  USE ALL THE NEW VALUES.
C
6       MIXX(JNF)=NEWX(JNF)
        MIXZ(JNF)=NEWZ(JNF)
        MIXN(JNF)=NEWN(JNF)
        MIXA(JNF)=NEWA(JNF)
        VAL=10.
7       JNF=JNF+1
        JSAVE=J+1
        J=JOF
        GO TO 2
8       MIXX(JNF)=0.
        MIXZ(JNF)=0.
        MIXN(JNF)=0
        MIXA(JNF)=0.
        JNF=JNF+1
40      IF (IFIX(OLDX(JOF)).EQ.0) GO TO 50
        JOF=JOF+1
        GO TO 40
50      JOF=JOF+1
        J=JOF
        IF(JNF.GE.MAXN) GO TO 9
        GO TO 2
C
C       PUT THE OUTPUT INTO THE MIXED LIST.
C
9       DO 10 I=1,MAXN
10      WRITE(22,14) MIXX(I),MIXN(I),MIXZ(I),MIXA(I)
14      FORMAT(2X,F6.1,',  ',I3,',  ',F7.3,',  ',F7.3)
        STOP
        END
```

```
C
C
C                       **** GAUDIS.FR ****
C
C       GENERATES GAUSS STROKES USING AN EVENT DISTANCE MEASURE
C
        REAL X(1800),A(1800),W(1800)
        REAL AS(150),WS(150)
        REAL E(1800),KT
        INTEGER PTR(1800),Y(150),BY(1800)
        INTEGER SED(1000),SEU(1000)
        INTEGER LS(1800),HL(20)
        CALL FOPEN(20,"GWD")
        CALL FOPEN(21,"A10D")
        CALL FOPEN(22,"NEWGAUSS")
        CALL FOPEN(23,"GCONTS")
        CALL FOPEN(24,"DOGHEAD")
C
C       HERE ARE THE IMPORTANT CONSTANTS
C
C       EP1: FIRST PASS ERROR THRESHOLD
C       EP2: SECOND PASS ERROR THRESHOLD
C       C1: WIDTH CONSTANT IN THE DISTANCE MEASURE
C       C2: AMPLITUDE CONSTANT IN THE DISTANCE MEASURE
C       ELIM: FRACTION OF FIRST PASS EVENTS THAT MUST BE USED
C       Q: "PINCHING" FACTOR IN THE OVAL OF CASSINI
C            0 <= Q < 1
C            0 GIVES A PURE ELLIPSE
C            1 GIVES A PURE LEMNISCATE OF BERNOULLI
C       BRK: FRACTION OF INTERPOLATED EVENTS IN A STROKE PERMITTED WITHOUT
C            SPLITTING THE STROKE INTO TWO STROKES
C       L2:  THE HEIGHT OF THE TRUNCATED SEARCHING HEMISPHERE
C       WXVAR: HORIZONTAL PREFERENCE CONSTANT FOR WIDE GAUSSIANS
C            0 <= WXVAR <= 1
C            0 GIVES NO PREFERENCE
C            1 GIVES FULLY LINEAR PREFERENCE
C       WTVAR: TOTAL GEOMETRIC DISTANCE WEIGHTING
C       ICHECK: IF 1, THEN GEOMETRIC DISTANCE IS DIVIDED BY AMPLITUDE SQUARED
C            OTHERWISE, IT IS NOT.
        READ(23)EP1,EP2,C1,C2,ELIM,Q,BRK,L2,WXVAR,WTVAR,ICHECK
        QS=Q*Q
        QF=QS*QS
C
C       GET THE DATA AND INTERPRET WIDTHS
C
        DO 1 K=1,21
1       READ(20) WS(K)
        DO 5 J=1,10000
        READ(21,END=15) X(J),LS(J),E(J),A(J)
        IF(LS(J).NE.0)GO TO 5
C       THE WIDE GAUSSIANS ARE NOT NEEDED IN THIS PROGRAM
```

```
C       THEY ARE PUT INTO THE FILE "DOGHEAD"
        DO 2 I=1,7
        READ(21)DUMMY
2       WRITE(24)DUMMY
5       CONTINUE
15      L=J-1
        TYPE "L=",L
        DO 20 I=1,L
        W(I)=0.
        IF (LS(I).EQ.0) GO TO 20
C       THIS GIVES AN ABSOLUTE WIDTH FROM A RANGE AND OFFSET PASSED BY "LEST"
        W(I)=WS(LS(I))*E(I)+WS(LS(I)+1)*(1-E(I))
20      CONTINUE
C
C       CALCULATE THE ENERGY AND SORT LS
C
        TYPE "ENERGY"
        IRATE=1
        RMAX=0
        DO 21 I=1,L
        E(I)=A(I)*A(I)*W(I)
        IF(E(I).GT.RMAX) RMAX=E(I)
21      PTR(I)=0
C       THIS IS A COARSE EXPONENTIAL SORTING ALGORITHM
        DO 25 J=1,50
        RLEVEL=RMAX/(2.**(FLOAT(J)/3.))
        TYPE "IRATE = ",IRATE
        DO 25 K=1,L
        IF(PTR(K).NE.0) GO TO 25
        IF(E(K).LT.RLEVEL) GO TO 25
        LS(IRATE)=K
        IRATE=IRATE+1
        PTR(K)=1
25      CONTINUE
        TYPE"IRATE=",IRATE,"   RMAX=",RMAX,"   RMIN=",RMIN
        TYPE"RLEVEL=",RLEVEL
C
C       SETUP Y, BY AND CLEAR PTR, SED, SEU
C
        TYPE "SETUP Y, ETC."
        K=1
        Y(1)=1
        DO 34 I=1,L
        BY(I)=K
        IX=IFIX(X(I)+.5)
        IF(IX.NE.0) GO TO 34
        K=K+1
        Y(K)=I+1
34      PTR(I)=0
        DO 35 I=1,600
        SEU(I)=0
```

```
35      SED(I)=0
        MAXY=K-1
        IFLAG=0
        EP=EP1
C
C       SETUP HL, THE SEARCHING HEMISPHERE
C
        HL(1)=9
        HL(2)=9
        HL(3)=9
        HL(4)=9
        HL(5)=8
        HL(6)=8
        HL(7)=7
        HL(8)=6
        HL(9)=4
        HL(10)=0
C       L1 IS THE BOTTOM OF THE SEARCHING HEMISPHERE
        L1=1
C
C       SELECT NEXT ORIGIN OF STROKE
C       IPEN: HOW FAR DOWN IN LS WE ARE LOOKING
C       ISTR: THE STROKE NUMBER
C       ELIM: PERCENT LIMIT OF EVENTS BEFORE QUITTING
C       IMARK: NEXT THRESHOLD BEFORE EVENT CHECK
C
        TYPE "STROKE"
        IBOT=IFIX(ELIM*FLOAT(L-128))
        TYPE "L=",L,"   IBOT=",IBOT
        IPEN=1
        ISTR=0
        IMARK=IFIX(0.02*FLOAT(L-128))
        IINC=IMARK
36      IF(IPEN.LT.IMARK) GO TO 38
        ITOTAL=0
        DO 37 J=1,L
        IND=PTR(J)
        IF(IND.NE.0) ITOTAL=ITOTAL+1
 37      CONTINU
C       ALL DONE WITH EP1 WHEN ITOTAL>=IBOT
        IF(ITOTAL.GE.IBOT) GO TO 200
        IMARK=IMARK+IINC
38      IXX=IFIX(X(LS(IPEN)))
        IF(IXX.EQ.0) GO TO 39
        IF(PTR(LS(IPEN)).EQ.0) GO TO 40
39      IPEN=IPEN+1
        GO TO 3
40      ISTR=ISTR+1
        LSIP=LS(IPEN)
        TYPE "EP1 STROKE NUMBER ",ISTR
        SED(ISTR)=LS(IPEN)
```

```
      SEU(ISTR)=LS(IPEN)
      PTR(LS(IPEN))=-1
C
C     STROKE DOWNWARD
C     PTR(I) POINTS TO THE PREVIOUS EVENT OF STROKE
C     SED(I) POINTS TO THE DOWNWARD END OF STROKE
C
44    IY=BY(SED(ISTR))+1
50    IF (IY.GT.MAXY) GO TO 90
      ISRCH=Y(IY)
      IF(IFIX(X(ISRCH)+0.5).EQ.0) GO TO 90
      DMIN=10.E6
C     NE IS THE EVENT NUMBER AT THE END OF STROKE
      NE=SED(ISTR)
C     IX IS THE X POSITION OF THE END OF STROKE
      IX=IFIX(X(NE)+0.5)
C     J IS THE NUMBER OF LINES DOWN FROM THE END OF THE STROKE
      DO 80 J=L1,L2
      LU=IX+HL(J)
      LL=IX-HL(J)
60    IF (PTR(ISRCH).NE.0) GO TO 70
      IF (X(ISRCH).GT.LU) GO TO 70
      IF (X(ISRCH).LT.LL) GO TO 70
C
C     GET THE DISTANCE
C
      XF=2*(X(ISRCH)-FLOAT(IX))/(W(ISRCH)**WXVAR)
      YF=J
      XSD=XF*XF
       YSD=YF*Y
      DIS=QS*(XSD-YSD)+SQRT(XSD*XSD+2*XSD*YSD*(1-QF)+YSD*YSD)
      DIS=DIS*WTVAR/(1-QF)
      IF (ICHECK.EQ.1) DIS=DIS/(A(ISRCH)*A(ISRCH))
      WD=C1*(W(ISRCH)-W(NE))**2/(W(ISRCH)+W(NE))**2
      AD=C2*(A(ISRCH)-A(NE))**2/(A(ISRCH)+A(NE))**2
      IF(A(ISRCH).GT.0.AND.A(NE).LT.0) AD=1E6
      IF(A(ISRCH).LT.0.AND.A(NE).GT.0) AD=1E6
      DIS=DIS+WD+AD
      IF (DIS.GT.DMIN) GO TO 70
C     DMIN IS THE SMALLEST DISTANCE FOUND IN THE SEARCHING HEMISPHERE
      DMIN=DIS
C     NVNT IS THE NEXT EVENT TO BE CALLED THE END OF STROKE
      NVNT=ISRCH
      JMIN=J
70    ISRCH=ISRCH+1
      IF(IFIX(X(ISRCH)+0.5).EQ.0) GO TO 79
      GO TO 60
79    ISRCH=ISRCH+1
      IF (IY+J.GT.MAXY) GO TO 81
80    CONTINUE
81    IF(DMIN.GT.EP) GO TO 90
```

```
      IY=IY+JMIN
      PTR(NVNT)=SED(ISTR)
      SED(ISTR)=NVNT
      GO TO 50
C
C     STROKE UPWARD
C
90    IY=BY(SEU(ISTR))-1
95    IF (IY.LT.1) GO TO 210
      ISRCH=Y(IY)
      IF(IFIX(X(ISRCH)+0.5).EQ.0) GO TO 210
      DMIN=10.E6
      NE=SEU(ISTR)
      IX=IFIX(X(NE)+0.5)
      DO 120 J=L1,L2
      LU=IX+HL(J)
      LL=IX-HL(J)
100   IF (PTR(ISRCH).NE.0) GO TO 110
      IF(X(ISRCH).GT.LU) GO TO 110
      IF(X(ISRCH).LT.LL) GO TO 110
C
C     GET THE DISTANCE
C
      XF=2*(X(ISRCH)-FLOAT(IX))/(W(ISRCH)**WXVAR)
      YF=J
      XSD=XF*XF
      YSD=YF*YF
      DIS=QS*(XSD-YSD)+SQRT(XSD*XSD+2*XSD*YSD*(1-QF)+YSD*YSD)
      DIS=WTVAR*DIS/(1-QF)
      IF (ICHECK.EQ.1) DIS=DIS/(A(ISRCH)*A(ISRCH))
      WD=C1*(W(ISRCH)-W(NE))**2/(W(ISRCH)+W(NE))**2
      AD=C2*(A(ISRCH)-A(NE))**2/(A(ISRCH)+A(NE))**2
      IF(A(ISRCH).GT.0.AND.A(NE).LT.0) AD=1E6
      IF(A(ISRCH).LT.0.AND.A(NE).GT.0) AD=1E6
      DIS=DIS+WD+AD
      IF(DIS.GT.DMIN) GO TO 110
      JMIN=J
      DMIN=DIS
      NVNT=ISRCH
110   ISRCH=ISRCH+1
      IF(IFIX(X(ISRCH)+0.5).EQ.0) GO TO 119
      GO TO 100
119   IF (IY-J.LT.1) GO TO 121
      ISRCH=Y(IY-J)
120   CONTINUE
C     GET THE NEXT STROKE
121   IF(DMIN.GT.EP) GO TO 210
      IY=IY-JMIN
      PTR(NVNT)=SEU(ISTR)
      SEU(ISTR)=NVNT
      GO TO 95
```

```
C
C       REDUCE TO EP2 AND CONTINUE
C
200     EP=EP2
        TYPE "OK DOWN TO EP2"
C       IFLAG IS SET TO INDICATE THAT THIS IS THE SECOND PASS
        IFLAG=1
        ITOP=ISTR
        ISTR=0
210     IF(IFLAG.EQ.0) GO TO 36
        ISTR=ISTR+1
        IF (ISTR.GT.ITOP) GO TO 400
        TYPE "EP2 STROKE NUMBER ",ISTR
        GO TO 44

C       INTERPOLATES AND GENERATES A CORRECTION LIST
C
400     ISTR=ISTR-1
        TYPE "INTERPOLATE"
        K=0
        JKJ=0
        DO 700 I=1,ISTR
C       I IS THE STROKE NUMBER
C       IZ IS A FLAG FOR UP OR DOWN STROKING
        TYPE "I=",I
        IZ=1
        IFPTR=SEU(I)
        NPTR=PTR(IFPTR)
430     IFY=BY(IFPTR)
        K=K+1
        LS(K)=IFY
        E(K)=X(IFPTR)
        AS(K)=A(IFPTR)
        WS(K)=W(IFPTR)
        IF (NPTR.EQ.-1) GO TO 460
        NY=BY(NPTR)
        IDY=IABS(NY-IFY)
        IF(IDY.EQ.1) GO TO 440
C       INTERPOLATES WHEN THERE ARE GAPS
        DX=(X(NPTR)-X(IFPTR))/FLOAT(IDY)
        DA=(A(NPTR)-A(IFPTR))/FLOAT(IDY)
        DW=(W(NPTR)-W(IFPTR))/FLOAT(IDY)
        IDYM=IDY-1
        DO 420 J=1,IDYM
        K=K+1
        E(K)=X(IFPTR)+FLOAT(J)*DX
        AS(K)=A(IFPTR)+FLOAT(J)*DA
        WS(K)=W(IFPTR)+FLOAT(J)*DW
420     LS(K)=-IFY-FLOAT(IZ*J)
440     IFPTR=NPTR
        NPTR=PTR(NPTR)
```

```
        GO TO 430
460     IF(IZ.EQ.-1) GO TO 470
        IZ=-1
        IFPTR=SED(I)
        NPTR=PTR(IFPTR)
        KM=K
        GO TO 430
C       FLIPS THE DOWNWARD PORTION OF STROKE OVER
C       THIS ACCOUNTS FOR THE FACT THAT ALL OF THE POINTERS
C       INITIALLY POINTED TO THE ORIGIN OF THE STROKE
470     KH=(K-KM+1)/2
        DO 475 J=1,KH
        KT=LS(KM+J-1)
        LS(KM+J-1)=LS(K-J+1)
        LS(K-J+1)=KT
        KT=E(KM+J-1)
        E(KM+J-1)=E(K-J+1)
        E(K-J+1)=KT
        KT=AS(KM+J-1)
        AS(KM+J-1)=AS(K-J+1)
        AS(K-J+1)=KT
        KT=WS(KM+J-1)
        WS(KM+J-1)=WS(K-J+1)
        WS(K-J+1)=KT
475     CONTINUE
        LS(K)=0.
        E(K)=0.
        AS(K)=0.
        WS(K)=0.
C
C       SPLITTING ROUTINE
C
C       COMPARES NUMBER OF ORIGINAL POINTS TO TOTAL IN EACH STROKE
C       IF THE RATIO EXCEEDS "BRK" THEN THE STROKE IS SPLIT INTO
C       TWO SEPARATE STROKES.
C
        LMAX=0
        LGAP=0
        LTOT=0
        L=0
        IZ=0
500     DO 550 J=1,K
        IF(LS(J).EQ.0) GO TO 520
        IF(LS(J).GT.0) GO TO 510
        L=L+1
        LGAP=LGAP+1
        IF(L.LE.LMAX) GO TO 505
        LMAX=L
        KMAX=J-L
505     LTOT=LTOT+1
        GO TO 550
```

```
510     L=0
        LTOT=LTOT+1
        GO TO 550
520     JJ=J-1
        IF(LS(JJ).EQ.0) GO TO 550
        RATIO=FLOAT(LGAP)/FLOAT(LTOT)
        IF(RATIO.LE.BRK) GO TO 540
        DO 530 M=1,LMAX
        MM=M+KMAX
        TYPE "Y,X,A,W=",LS(MM),E(MM),AS(MM),WS(MM)
        LS(MM)=0
        E(MM)=0
        AS(MM)=0
530     WS(MM)=0
        ISPLIT=1
C       A STROKE HAS BEEN SPLIT
        TYPE "SPLIT!  K=",K,"   LMAX=",LMAX,"   KMAX=",KMAX
540     LMAX=0
        LGAP=0
        LTOT=0
550     CONTINUE
        IF(ISPLIT.EQ.0) GO TO 560
        ISPLIT=0
        GO TO 500
C
C       OUTPUT THE CORRECT ARRAYS
C
560     DO 600 II=1,K
        IF(LS(II).NE.0) GO TO 580
        IF(IZ.NE.0) GO TO 600
        IZ=1
        GO TO 590
580     IZ=0
590     LS(II)=IABS(LS(II))
        WRITE(22) LS(II),",",E(II),",",AS(II),",",WS(II)
600     CONTINUE
        JKJ=JKJ+K
700     K=0
        TYPE "THE FILE LENGTH IS:",JKJ
        STOP
        END
C
C       THE END ... GAUDIS.FR
C
```

```
C       GSSCOD2.FR
C
C
C       ****CODE GAUSSIAN STROKES****
C
C       THE GROUPED GAUSSIAN EVENTS FOUND IN THE INPUT FILE, "NEWGAUSS",
C       ARE FITTED WITH SECOND ORDER CURVES USING THE SUBROUTINE
C       "CRVFIT".  THIS IS DONE FOR POSITION, WIDTH, AND AMPLITUDE.
C       THE STROKES ARE THEN CODED AND PUT INTO THE OUTPUT FILE
C       "GSSCODES2".  THE ERROR BOUNDS USED IN THE FITTING ROUTINE
C       ARE FOUND IN THE FILE "GSSBOUNDS2".  IN THIS VERSION OF THE
C       PROGRAM, NO STROKES ARE THROWN AWAY.
C
        REAL X(2800),W(2800),A(2800),SLOPE(128),ACCEL(128),CODE(400)
        INTEGER LENGTH(128),Y(2800)
        INTEGER ST,NST,ISTR,ZZ
        REAL IT(128),IH(128),IR(128),ISR(128)
        CALL FOPEN(20,"NEWGAUSS")
        CALL FOPEN(30,"GSSBOUNDS2")
        CALL FOPEN(40,"GSSCODES2")
        READ(30)EX,EA,EW
        DO 1 I=1,2000
1       READ(20,END=2)Y(I),X(I),A(I),W(I)
2       L=I-1
C
C       CALCULATE SUMMATIONS FOR MEAN SQUARED ERROR FIT.
C
        IT(1)=1.
        IH(1)=1.
        IR(1)=1.
        ISR(1)=1.
        DO 5 I=2,128
        IT(I)=IT(I-1)+FLOAT(I)**2
        IH(I)=IH(I-1)+FLOAT(I)**3
        IR(I)=IR(I-1)+FLOAT(I)**4
5       ISR(I)=IT(I)*IR(I)-IH(I)**2
C
C       BEGIN CODING
C
        ST=1
        NST=1
        M=1
        ZZ=0
10      CODE(M)=Y(ST)
        CODE(M+1)=X(ST)
        CODE(M+2)=A(ST)
        CODE(M+3)=W(ST)
        M=M+1
C
C       CODE X
C
```

```
      CALL CRVFIT(ST,NST,X,EX,N,SLOPE,ACCEL,LENGTH,ISR,IT,IH,IR,LLL)
      IF (NST-2.LE.ST) GO TO 45
      IF (NST-3.GT.ST) GO TO 222
      CODE(M+3)=1
      CODE(M+4)=SLOPE(1)
      CODE(M+5)=0
      CODE(M+6)=0
      ZZ=1
      GO TO 21
222   DO 20 J=1,N
      M=M+3
      CODE(M)=LENGTH(J)
      CODE(M+1)=SLOPE(J)
20    CODE(M+2)=ACCEL(J)
      CODE(M+3)=0
      M=M+1
C
C     CODE A
C
21    CALL CRVFIT(ST,NST,A,EA,N,SLOPE,ACCEL,LENGTH,ISR,IT,IH,IR,LLL)
      IF (ZZ.EQ.0) GO TO 22
      CODE(M+7)=1
      CODE(M+8)=SLOPE(1)
      CODE(M+9)=0
      CODE(M+10)=0
      GO TO 31
22    DO 30 J=1,N
      M=M+3
      CODE(M)=LENGTH(J)
      CODE(M+1)=SLOPE(J)
30    CODE(M+2)=ACCEL(J)
      CODE(M+3)=0
      M=M+1
C
C     CODE W
C
31    CALL CRVFIT(ST,NST,W,EW,N,SLOPE,ACCEL,LENGTH,ISR,IT,IH,IR,LLL)
      IF (ZZ.EQ.0) GO TO 32
      ZZ=0
      CODE(M+11)=1
      CODE(M+12)=SLOPE(1)
      CODE(M+13)=0
      CODE(M+14)=0
      M=M+14
      GO TO 46
32    DO 40 J=1,N
      M=M+3
      CODE(M)=LENGTH(J)
      CODE(M+1)=SLOPE(J)
40    CODE(M+2)=ACCEL(J)
45    CODE(M+3)=0
```

```
      M=M+3
46    ST=NST
      DO 50 II=1,M
50    WRITE(40)CODE(II)
      M=1
C
C     DO NEXT STROKE
C
      IF (ST.LE.L) GO TO 10
      STOP
      END
```

```
C
C
C       ***** RGSSREC2.FR *****
C
C
C
        REAL X(2800),W(2800),A(2800),IZZZ(10)
        INTEGER CH(2800),TOP(130),Y(2800)
        REAL XP(100),ZP(100),AP(100),WIDTH(30)
        INTEGER NP(100)
14      FORMAT(2X,I3,',  ',I3,',  ',F7.3,'  ',F7.3)
        CALL FOPEN(20,"GWD")
        CALL FOPEN(21,"GSSCODES2")
        CALL FOPEN(22,"BH2OD")
        CALL FOPEN(23,"DOGHEAD")
        CALL FOPEN(44,"SMEGMOID")
        IZZZ(1)=114.449
        IZZZ(2)=12.6572
        IZZZ(3)=106.456
        IZZZ(4)=16.9922
        IZZZ(5)=104.535
        IZZZ(6)=18.1250
        IZZZ(7)=112.948
        J=1
        DO 1 I=1,2800
        IF(I.GT.128) GO TO 1
        TOP(I)=0
1       CH(I)=0
2       READ(21,END=50) YR,X(J),A(J),W(J)
        IF (A(J).EQ.0.) TYPE "A=0 AT J=",J
        Y(J)=YR+0.01
        TYPE "J=",J,"   Y(J)=",Y(J),"   X(J)=",X(J)
        CH(J)=TOP(Y(J))
        TOP(Y(J))=J
        JSAVE=J
        READ(21)XL
        JINT=J
        IXL=IFIX(XL+0.001)
        IF(IXL.NE.0) GO TO 111
        TYPE "SINGLE EVENT!"
        J=J+1
        GO TO 2
3       READ(21)XL
        JINT=J
        IXL=IFIX(XL+0.001)
        IF(IXL.EQ.0) GO TO 5
111     READ(21)XS
        READ(21)XA
        DO 4 I=1,IXL
        J=J+1
4       X(J)=X(JINT)+XS*I+XA*I*I
```

```
        GO TO 3
5       J=JSAVE
6       READ(21)AL
        JINT=J
        IAL=IFIX(AL+0.001)
        IF(IAL.EQ.0) GO TO 8
        READ(21)AS
        READ(21)AA
        DO 7 I=1,IAL
        J=J+1
7       A(J)=A(JINT)+AS*I+AA*I*I
        IF(IFIX(10000*A(J)+.01).EQ.0) TYPE "INERPOLATED A=0 AT J=",J
        GO TO 6
8       J=JSAVE
9       READ(21)WL
        JINT=J
        IWL=IFIX(WL+0.001)
        IF(IWL.EQ.0) GO TO 11
        READ(21)WS
        READ(21)WA
        DO 10 I=1,IWL
        J=J+1
10      W(J)=W(JINT)+WS*I+WA*I*I
        GO TO 9
11      IS=JSAVE+1
        INIT=Y(JSAVE)
        DO 12 I=IS,J
        Y(I)=INIT+I-IS+1
        CH(I)=TOP(Y(I))
12      TOP(Y(I))=I
        J=J+1
        GO TO 2
50      J=J-1
C
C       PUT THE DATA INTO "LEOD"'S FORM
C
        DO 2222 JJ=1,J
        IF(IFIX(10000*A(JJ)+.01).EQ.0)TYPE "A=0 BEFORE LEOD AT J=",JJ
        IF(IFIX(10000*(1-A(JJ))+.01).EQ.0)TYPE "A=1 BEFORE LEOD AT J=",JJ
2222    WRITE (44) Y(JJ),X(JJ),A(JJ),W(JJ)
        DO 51 I=1,21
51      READ(20)WIDTH(I)
        K=0
        L=0
52      K=K+1
        IPTR=TOP(K)
53      L=L+1
        TYPE K,L
        IF(IPTR.EQ.0) GO TO 60
        XP(L)=X(IPTR)
        AP(L)=A(IPTR)
```

```
      IAP=IFIX(10000*AP(L)+.01)
      IF (IAP.EQ.0) TYPE "AP=0 AT L=",L
      IF(W(IPTR).LT.WIDTH(1))W(IPTR)=WIDTH(1)
      IF(W(IPTR).GT.WIDTH(21))W(IPTR)=WIDTH(21)
      DO 54 I=1,19
      IF(W(IPTR).LT.WIDTH(I+1)) GO TO 55
54    CONTINUE
      I=20
55    NP(L)=I
      ZP(L)=(W(IPTR)-WIDTH(I+1))/(WIDTH(I)-WIDTH(I+1))
      IZP=IFIX(10000*ZP(L)+.01)
      IF (IZP.EQ.0) TYPE "ZP=0 AT L=",L
      IF (N.EQ.20) TYPE "N=20 AT L=",L
      IF (N.EQ.1)  TYPE "N=1 AT L=",L
      IPTR=CH(IPTR)
      GO TO 53
60    XP(L)=0.
      NP(L)=0.
      ZP(L)=0.
      AP(L)=0.
C
C     NOW SPIT IT OUT
C
      DO 70 I=1,L
      XP(I)=0.5*IFIX(2*XP(I)+0.5)
      IF(AP(I).GT.1.0)AP(I)=1.0
      IF(AP(I).LT.-1.0)AP(I)=-1.0
      IF(XP(I).LT.1)XP(I)=1.01
      IF(XP(I).GT.128)XP(I)=128
      IF(NP(I).EQ.0)XP(I)=0.
      IXP=XP(I)
      WRITE(22,14) IXP,NP(I),ZP(I),AP(I)
70    CONTINUE
      DO 100 II=1,7
      READ(23)DUMMY
100   WRITE(22)IZZZ(II)
      L=0
      IF(K.LT.128) GO TO 52
      STOP
      END
```