(NASA-CR-174341) INFORMATION EXTRACTION AND TRANSMISSION TECHNIQUES FOR SPACEBORNE SYNTHETIC APERTURE RADAR IMAGES Final Technical Report (Kansas Univ. Center for Research, Inc.) 274 p HC A12/MF A01 CSCL

N85-17256

Unclas
G3/32 13531

THE UNIVERSITY OF KANSAS CENTER FOR RESEARCH, INC.

2291 Irving Hill Drive-Campus West
Lawrence, Kansas 66045

INFORMATION EXTRACTION AND TRANSMISSION TECHNIQUES FOR SPACEBORNE SYNTHETIC APERTURE RADAR IMAGES

Final Technical Report 596-5

by

Victor S. Frost, Principal Investigator,
with L. Yurovsky, E. Watson, K. Townsend,
S. Gardner, D. Boberg, J. Watson, G.J. Minden,
and K.S. Shanmugan

Supported by NASA Headquarters
NASA Grant NAGW-381

December 1984

TISL

TELECOMMUNICATIONS AND INFORMATION SCIENCES LABORATORY
THE UNIVERSITY OF KANSAS CENTER FOR RESEARCH, INC.
2291 Irving Hill Drive - Campus West Lawrence, Kansas 66045

ABSTRACT

The purpose of this research was to investigate information extraction and transmission techniques for synthetic aperture radar (SAR) imagery. Four interrelated problems were addressed. An optimal tonal SAR image classification algorithm was developed and evaluated. A data compression technique was developed for SAR imagery which is simple and provides a 5:1 compression with acceptable image quality. An optimal textural edge detector was developed. Several SAR image anhancement algorithms have been proposed. A study was undertaken to quantatively compare the effectiveness of each algorithm.

1.0 INTRODUCTION

The value of spaceborne remote sensing systems have been clearly demonstrated by the success of the LANDSAT series of satellites. The technology for extracting useful information from the data returned from these satellites is well developed. Recently, a new class of spaceborne imaging systems have been tested, Synthetic Apertures Radar (SAR). Processing the data from spaceborne SAR systems to form images is a non-trivial task and a great deal of research is currently underway to develop high speed SAR processors. Unfortunately, a similar effort is not being devoted to the development of information extraction techniques for SAR. The research reported here developed information extraction and transmission techniques for SAR.

Four interrelated problems have been addressed: 1) classification; 2) data compression; 3) textural modeling and edge detection; and 4) image enhancement. An appendix is attached for each of these topics which describes the results of our investigations.

2.0 BACKGROUND AND OVERVIEW

Digital image processing techniques have been successfully applied to a wide variety of problem ranging from the analysis of X-ray images to identeification of handwriting to the extraction of information from satellite images [1, 2, 3, 4, 5]. Each new problem in image processing presents new technical challenges which must be addressed. This research addressed important problems involved in digital processing of SAR images.*

Spaceborne imaging radars are important because they provide a unique view of the earth's surface. Their imaging geometry, spectral characteristics, and all-weather capability give active microwave systems an advantage over conventional imaging techniques, e.g., photography. For many years, imaging radars have been successfully used for military reconnaissance and geologic mapping. Operational systems arecurrently in use for ice surveys and the detection of oil spills on the world's oceans.

Until very recently, all informatin was extracted from radar images by human interpreters. Other techniques were not considered because radar data was disseminated as photographic products, e.g., paper or film positives. Not only was it awkward for an interpreter to convert these data into a digital format for automatic or machine-aided information extraction, but this conversion process degraded the quality of the data. The advent of the digital signal correlator [e.g., 6] for synthetic aperture radars (SAR) has changed this, and now radar images are commonly distributed in a digital format. In addition, quantitative information is now desired from radar images, e.g., soil moisture and crop type estimates are being sought. The volume of data collected from proposed spaceborne imaging radar missions can be enormous, as shown by the Seasat-A SAR mission. If imaging radars are to meet their full potential, then automatic information extraction is needed, or at least machine-aided analysis to ease the work load on the interpreters is required.

An initial approach for automatic information extraction from radar imagery would be to employ the well developed technology associated with other sensors [1-5] (e.g., LANDSAT). This method was unsuccessful. The failure of

* We are concerned here with extracting information from SAR images, not processing the received inphase and quadrature voltages to form the SAR image.

this approach occurred because existing processing algorithms were designed assuming a specific system and statistical model. The most common model assumed was additive, white Gaussian noise (AWGN). This model does not apply to radar images. Therefore, processing algorithms based on an AWGN did not perform satisfactorily when applied to radar images.

It has been shown that [7, 8] radar images can be modeled by a multiplicative noise process which is non-Gaussian and has a poor signal-to-noise ratio. The standard technique for improving the interpretability of SAR images has been to use noncoherent integration [9-11]. Noncoherent averaging is used extensively for noise reduction in coherent systems [12-14] and can be implemented in many ways. The basic idea behind this method is that independent samples (or looks) of the terrain are gathered and averaged after detection. Independent samples can be obtained by polarization or frequency diversity. Most SAR systems use frequency diversity, i.e., nonoverlapping subareas of the spectrum of the complex received signals are used to form the independent images.

Continuous scanning of the spectrum is also used [11, 13, 14]. The net effect is to reduce the bandwidth (degrade the spatial resolution), while improving the signal-to-noise ratio, S/N. Several studies [10, 15, 16] have shown the advanrages of noncoherent processing for the interpretation of SAR imagery. However, there are several disadvanrages to this approach: 1) the technique is spatially invariant and thus does not account for the multiplicative nature of the noise and the nonstationarity of the signal; 2) the technique was developed for coherent optical processors and thus it is easily implemented with such a processor but is not necessarily optimum for digital processors; and 3) the technique is only aimed at improving the S/N and not for direct extraction of information.

Several digital image enhancement algorithms have recently been developed [17, 18, 19] to treat the multiplicative nature of SAR images. Also, several hueristic approaches have been applied [20]. The technique we developed in [17] is an adaptive minimum mean square error spatial filter which preserves edges while smoothing homogeneous areas, e.g., agricultural fields. Homomorphic filtering has also been used [18] for enhancement of speckled images. A linear approximation to the multiplicative noise process is used in [19] to develop an adaptive filter based on the local mean and variance. Whereas [17] and [19] demonstrate the performance of their algorithms on actual SAR imagery, only simulations are used in [18]. All of these algorithms on actual SAR imagery, only simulations are used in [18]. All of these algorithms have only been directed toward image enhancement assuming that the information extraction process would be performed manually. While these techniques are of great value for manual interpretation of SAR imagery, algorithms for automating the information extraction tasks are needed.

Appendix A describes a quantative evaluation of several of these enhancement algorithms. The comparison was based on both an edge quality measure and computer execution time.

What is the information to be extracted from spaceborne SAR imagery? Active microwave remote sensing is being used for several applications (e.g., geology, ice mapping, and monitoring wave conditions in the ocean); several more applications have been proposed, e.g., soil moisture measurement, crop classification, monitoring crop growth and harvest progress, and snow mapping [21].

In each application, the SAR is used to measure different properties of the earth's surface. The problem of information extraction can be viewed as a signal analysis problem where different properties of the signal (here the SAR image) are used to imply certain properties of one target. For example, in geology SAR images are analyzed for their large scale spatial structure. Different image structures imply different geologic structures [22]. The backscattered power (which is mapped into the SAR image intensity) is used to estimate the soil moisture of crop type [23]. For ocean applications, it is the Fourier spectrum of the SAR image which is used to estimate ocean conditions.

Thus, the successful extraction and handling of information for SAR images requires: 1) identifying unique image properties; 2) relating those image properties to the ratget characteristics of interest; 3) finding ways of efficiently measuring those image properties; and 4) finding ways of efficiently representing (for storage or transmission) these image properties

Appendix B contains a SAR image classification algorithm. In this algorithm image, tone was the property of the signal (SAR image) used to obtain information about the earth. The multiplicative model for SAR images was used to develop a maximum likelihood classification algorithm. It was first assumed that the target feature information was known a-priori. A probability of incorrect classification was then determined for this algorithm as a function of the SAR parameters (e.g., the degree of noncoherent averaging). A technique was also developed to extract the target feature information from the supplied image. This classification algorithm was tested on SEASAT-A SAR imagery.

Appendix C contains a description of a data compression developed for SAR images. This technique was based on the multiplicative noise SAR image model and is designed to preserve the local statistics of the image. The technique is an adaptive variable rate modification of the block truncation coding technique developed in [24]. A data rate of approximately 1.6 bits/pixel is achieved with the technique while maintaining the image quality and cultural (point like) targets. The algorithm requires no large data storage and is computationally simple.

In some applications, the image attribute, desired signal characteristic, which is needed is clearly defined as in the case of image intensity. And the classification algorithm described in Appendix C provides the maximum likelihood technique to separating image features based on intensity (or tone). However, for others, a distinct signal property has yet to be identified. One promising attribute is texture. Texture is known to be important for the manual interpretation of SAR images for geology and these large scale texture differences have been shown to be separable using digital techniques [22]. Appendix D contains a description of an optimum textural edge detection filter.* This filter will be of value in separating regions of different textures in SAR images. The filter described in Appendix D is optimal in the sense that for the given texture model, a maximum amount of output signal energy is concentrated within a given resolution interval about the textural transition for a set filter bandwidth. The filter developed here is also a

* The development of the textural edge detection filter was also partially supported by NASA Contract No. NASA 9-16664.

global operator, rather than a local operator. In other words, the entire image is transformed and modified by the filter, instead of breaking the processing down into many local operations. The optimum textural edge detector is an extension of the optimum tonal edge detector [25].

With the increased availability of digital SAR images, information extraction and transmission algorithms will increase in importance. This research effort addressed several critical problems in SAR image processing. It is hoped that these algorithms will be both useful and provide a basis for further developmenet of radar image processing techniques.

3.0 REFERENCES

[1] W. K. Pratt, Digital Image Processing, John Wiley and Sons, New York, 1978.

[2] H. C. Andrews and B. R. Hart, Digital Image Restoration, Prentice-Hall, Inc., 1977.

[3] K. R. Castleman, Digital Image Processing, Prentice-Hall, Inc., 1979.

[4] "Special Issue on Pattern Recognition and Image Processing," IEEE Proceeding, Vol. 67, No. 5, May 1979.

[5] "Special Issue on Image Processing," IEEE Proceeding, Vol. 69, No. 5, May 1981.

[6] J. R. Bennet and I. G. Cummings, 1979. "A Digital Processor for the Production of Seasat-Synthetic Aperture Radar Imagery," SURGE Workshop, ESRIN, Frascati, Italy, (ESA Reprint SP-154).

[7] V. S. Frost, et al., "A Model for Imaging Radars and Its Application to Adaptive Digital Filtering for Multiplicative Noise," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 1, PAMI-4, No. 2, March 1982.

[8] J. W. Goodman, "Noise in Coherent Optical Processing," Optical Information Processing, Plenum Press, New York 1976.

[9] W. A. Penn, "Signal Fidelity in Radar Processing," IRE Trans. on Military Electronics, Vol. 6, April 1962, pp. 204-218.

[10] L. J. Porcello, "Speckle Reduction in Synthetic-Aperture Radars," J. Opt. Soc. Am., Vol. 66, No. 11, 1976, pp. 1305-1311.

[11] J. S. Zelenka, "Comparison of Continuous and Discrete Mixed-Integrator Processors," J. Opt. Soc. Am., Vol. 66, No. 11, 1976, pp. 1295-1304.

[12] J. C. Dainty, Laser Speckle and Related Phenomenon, Springer-Verlag, New York, 1975.

[13] P. Hariharan and Z. S. Hegedus, "Reduction of Speckle in Coherent Imaging by Spatial Frequency Sampling, Part I," Optica Acta, Vol. 21, No. 9, 1974.

[14] P. Hariharan and Z. S. Hegedus, "Reduction of Speckle in Coherent Imaging by Spatial Frequency Sampling, Part II," Optica Acta, Vol. 21, No. 9, 1974.

[15] R. K. Moore, "Tradeoff Between Picture Element Dimensions and Noncoherent Averaging in Side-Looking Airborne Radar," IEEE Trans. on Aerospace and Electronics, Vol. AES-15, September 1979, pp. 697-708.

[16] G. R. DiCaprio and J. F. Wasielewski, "Radar Image Processing and Interpreter Performance," Photogrammetic Eng. and Remote Sensing, Vol. 52, No. 8, August 1976, pp. 1043-1048.

[17] V. S. Frost, et al., "Adaptive Smoothing for Noisy Radar Images," IEEE Proceedings, Vol. 69, No. 1, January 1981.

[18] J. S. Lim and H. Nawab, "Techniques for Speckle Noise Removal," Optical Engineering, Vol. 20, No. 3, May/June 1981.

[19] J. S. Lee, "Speckle Analysis and Smoothing of Synthetic Aperture Radar Images," Computer Graphics and Image Processing, Vol. 17, pp. 17-32, December 1981.

[20] P. S. Chavez, "Automatic Shading Correction and Speckle Noise Mapping/Removal Techniques for Radar Image Data," Radar Geology: An Assessment, JPL Publication 80-61, July 1979.

[21] F. T. Ulaby, et al., Microwave Remote Sensing - Active and Passive, Vol. I, Addison-Wesley Co., Reading, Mass., 1981.

[22] K. S. Shanmugam, et al., "Texture Features for Radar Image Analysis," IEEE Transaction Geoscience and Remote Sensing, Vol. GE-19, No. 3, July 1981.

[23] F. T. Ulaby et al., "A Simulation Study of Soil Moisture Estimation by a Space SAR," Photogrammetic Engineering and Remote Sensing, Vol. 48, No. 4, April 1982, pp. 645-660.

[24] E. J. Delp and O. R. Mitchell, "Image Compression Using Block Truncation Coding," IEEE Trans. on Communication, Vol. COM-27, No. 9, Sept. 1979, pp. 1335-1342.

[25] K. S. Shanmugan, F. M, Dickey, J. A. Green, "An Optimal Frequency Domain Filter for Edge Detection in Pictures," IEEE Trans. Patt. Analysis and Mach. Intell., Vol. PAMI-1, No. 1, pp. 37-49, January 1979.

APPENDIX A

Evaluation of Edge Preserving Properties of Radar Image Enhancement Algorithms

EVALUATION OF EDGE PRESERVING PROPERTIES OF RADAR IMAGE ENHANCEMENT ALGORITHMS

J. Scott Gardner
Victor S. Frost
Jeff Watson

University of Kansas
Telecommunications and Information Sciences Laboratory

Technical Report TISL-5960-2

Supported by
NASA Headquarters Grant #NAGW-381

ABSTRACT

Many algorithms exist for the enhancement of synthetic aperture radar (SAR) images. These algorithms improve the signal-to-noise ratio of an image in order to make the image more useful for an observer. The trade-off for this improvement is a decrease in the image resolution and is seen as a blurring of fine detail. For many applications, most notably for the analysis of agricultural scenes, this blurring effect is only critical in edge areas where a boundary exists.

In this study several different enhancement algorithms were implemented. These algorithms were then compared based on an edge quality testing procedure. This procedure established a method with which the edge-preserving abilities of the various algorithms could be compared. The amount of computer processing time required by each algorithm was also recorded. Using the results of these comparisons, recommendations are made as to which algorithms are best suited for various applications.

LIST OF FIGURES

1.0 INTRODUCTION

The usefulness of synthetic aperture radar (SAR) imagery is dependent on the ability of an observer to recognize detailed features in an image. This ability is often greatly decreased by the presence of noise. Various algorithms have been developed to suppress noise in SAR imagery. (1,2,3,4) The problem with noise reduction algorithms is that they tend to suppress the desired signal as well as the noise. This is particularly a problem for edge areas. The purpose of this paper is to evaluate several of the algorithms in order to determine how well each algorithm preserves edge information while suppressing noise.

The principle type of noise in SAR imagery is speckle noise, which is multiplicative in nature. Speckling effects are due to the fact that SAR generates images by the coherent processing of the reflected signals, resulting in more noise in those areas where the signal is greater. This can be modeled mathematically by the equation:

$$Z_{i,j} = X_{i,j} V_{i,j} \tag{1.1}$$

where $Z_{i,j}$ is the observed power at a particular range and azimuth, $X_{i,j}$ is the signal that would ideally be observed and $V_{i,j}$ is the noise. The subscripts are present in order to emphasize that the image is to be processed digitally with i and j corresponding to a row and column in the SAR image to be analyzed. Hereafter, for simplicity, the subscripts will be omitted.

In order to improve the interpretability of an image, it is necessary to suppress speckle noise while enhancing the desired signal. Many different enhancement algorithms have been developed for this purpose. (1,2,3,4) Most algorithms suppress noise by averaging the surrounding points. That is, a pixel is replaced by an average of its neighbors, producing a smoother, less noisy image. However, since neighborhood averaging is often applied to all points within an image, the desired signal points are also averaged, leading to a degradation of the image resolution. In an edge area (defined as a boundary between two areas of differing average power return), the edge will also be smoothed, causing the boundary to appear "blurry". In many cases this retards interpretability as effectively as speckle noise. Obviously, it is desireable to suppress speckle while also preserving the edge information.

A useful application of edge preservation is analysis of agricultural areas in which relatively large homogeneous areas of differing radar reflectivity have distinct boundaries. While smoothing in the uniform regions suppresses speckle, a loss of resolution in these areas is not as critical, since there should be little variation in signal strength for a level field containing only one type of crop. (Variables such as soil mosture or plant disease would cause the field to appear as more than one region of uniformity.) However, no smoothing is desired in the edge area in order to preserve the distinct boundaries. The type of filter chosen for processing of noisy images significantly affects the amount of edge degradation in the processed image.

The extent to which an enhancement algorithm preserves edges can be evaluated by an "edge figure of merit" (EFM) algorithm. (5) The EFM establishes a means of image comparison by detecting the amount by which an edge was smoothed. In addition to detecting smoothed edges, the EFM may detect "false" edges due to noise. These combined effects contribute to the calculation of a relative EFM value which is used in drawing conclusions about the effectiveness of each algorithm.

This report provides a quantitative comparison of several different noise suppression algorithms. Section 2 provides a brief description of the development of a digital noise filter. Section 3 explains the manner in which the filter performances were quantitatively compared, and section 4 outlines the development of each filter compared in this study. A discussion of the results is given in section 5. The appendices contain the listings for all computer programs used in this study.

2.0 DIGITAL SPATIAL FILTERING TECHNIQUES

The purpose of this section is to briefly outline the considerations involved in the development of a spatial digital filter. Although many different types of image processing algorithms exist (1), this study focuses only on spatial filtering techniques. A large number and variety of algorithms have been developed to perform this type of filtering. The SAR image to be digitally enhanced is contained in a two-dimensional array of values representing the reflected power at each discrete area of the target. Each element of the array is a pixel in the image. Usually the image is been scaled to accomodate the display system.

The term "image enhancement" is the process of suppressing the image noise while retaining the signal. To best illustrate the procedure used in developing a specific filter for a particular application, it is helpful to examine the development and implementation of a simple, yet effective, filter -- the "equal-weighted" or "box filter."

Consider an NxM image f(i,j) containing both the signal and multiplicative noise. The enhancement procedure is to generate a smoothed image g(i,j) in which the gray level at every point (i,j) is the average of the gray levels of f contained in a predefined neighborhood of (i,j). (For this study, the neighborhood of (i,j) includes the point (i,j).) That is,

$$g(i,j) = \frac{1}{T} \sum_{(n,m)\varepsilon S} f(n,m) \qquad (2.1)$$

where S is the set of coordinates of points in the neighborhood of the point (i,j), and T is the total number of points defined by the coordinates in S (7). Computationally this process involves calculating an average for each nxm region of the image. The nxm region is referred to as a "window" because of the way in which the entire image is viewed nxm pixels at a time.

To generalize the filter, the filter window is defined in terms of an nxm array of weighted values (fig. 2.1). Each cell in the window contains a value which determines the degree to which the image gray level at that coordinate influences the average. By changing the window weight, it is possible to change the filter characteristics. Subsequent sections of this report examine the determination of window weightings for several different filters.

As an example, consider a 5x5 window with a gaussian weighting to be used in filtering a 100x100 pixel image. By dividing each element in the window by the sum of the window elements, the window is normalized so that it sums

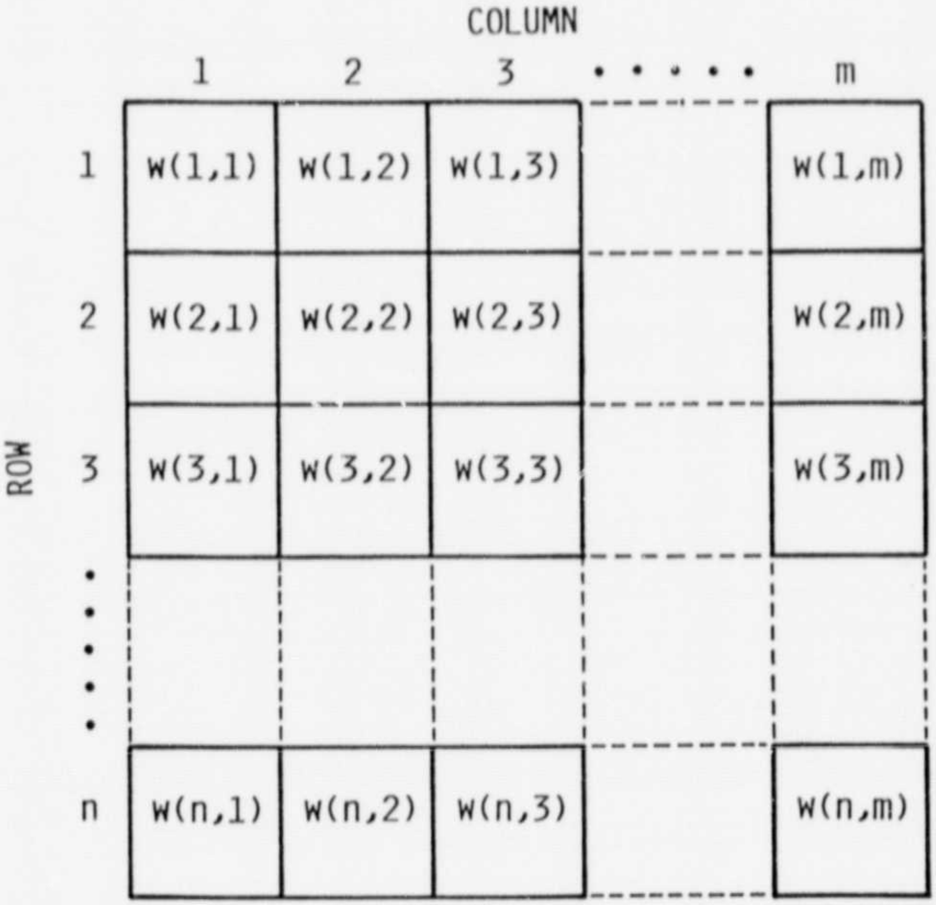


Figure 2.1 Weighted filter window

to one. The operation to be performed is a "moving average." Visualize placing the window on top of the image, starting with the upper left corner and moving across the image. At each position of the window, the sum of the products for the overlapping cells can be calculated. Since the window has been normalized, this sum will equal the weighted average of the values for the pixels "covered" by the window. The values in the window determine how much each covered image pixel is weighted in the average. For the normal weighting used in this example, the points farther from the center of the window have less influence on the average. Each sum is a point in the output image. Moving the window across generates a line in the output image. Once a line is output, the window is moved back to the left side of the image, down a row, and then across to generate the next line of output (fig. 2.2 illustrates the operation). At this point it is clear that the window will not be able to cover enough points at the end of a column or row for determining an average (fig. 2.3). These end points can either be copied directly into the filtered image or discarded, making the filtered image smaller by an amount equal to one less than the window size. (To prevent false edges from being created, the filters used in this study discard the end points).

With a basic idea of why windows are used in image

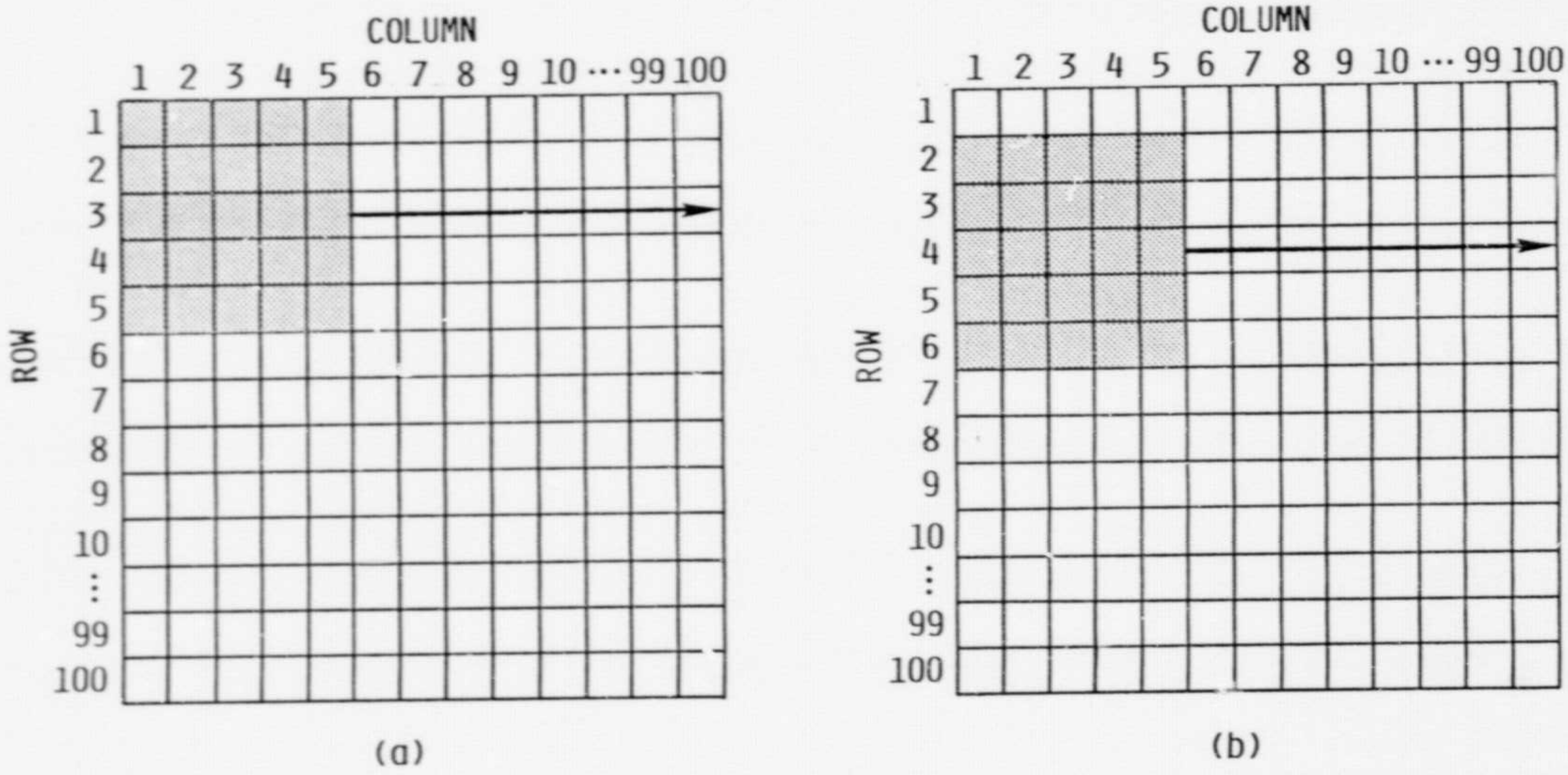


Figure 2.2 Filter window operation - One row of the output image is generated by each pass of the window across the image.

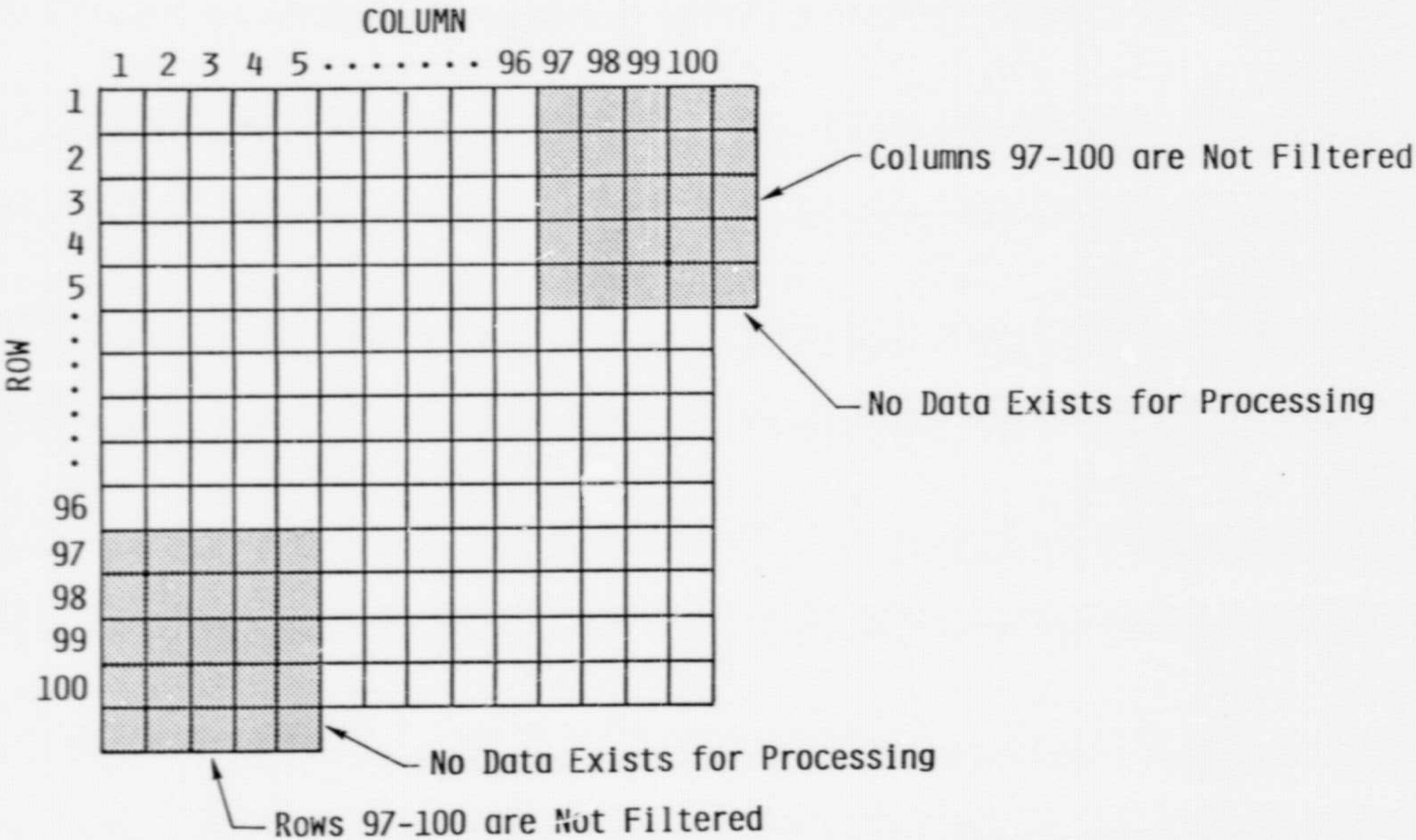


Figure 2.3 The size of the window affects the size of the output image

processing, the computational requirements for the implementation of a window operation can be examined. One of the most fundamental considerations for image processing is memory management. Typically a SAR image is quite large; usually there are over 500x500 pixels, while an image containing 2000x2000 pixels is not uncommon. To store an entire image in memory while it is being processed can take up many megabytes of storage space. Obviously some thought must be given to how much of the image is needed at any one time for processing. Most images are stored sequentially. That is, the top record or line of data is read first while following lines are read in the order that they appear in the image (fig. 2.4). Therefore the algorithm must also take into account the order in which the data is read.

One of the most efficient algorithms to deal with window operations is a two-dimensional circular queue. This data structure is a "first-in-first-out" construct. Since only the rows covered by the window are being processed at any one time, this is the only data which needs to be in memory.

All of the programs presented in this paper utilize a circular queue to implement various types of window operations. A straightforward illustration of the queueing technique as applied to a spatially invariant two-dimensional convolution is presented in Appendix A.

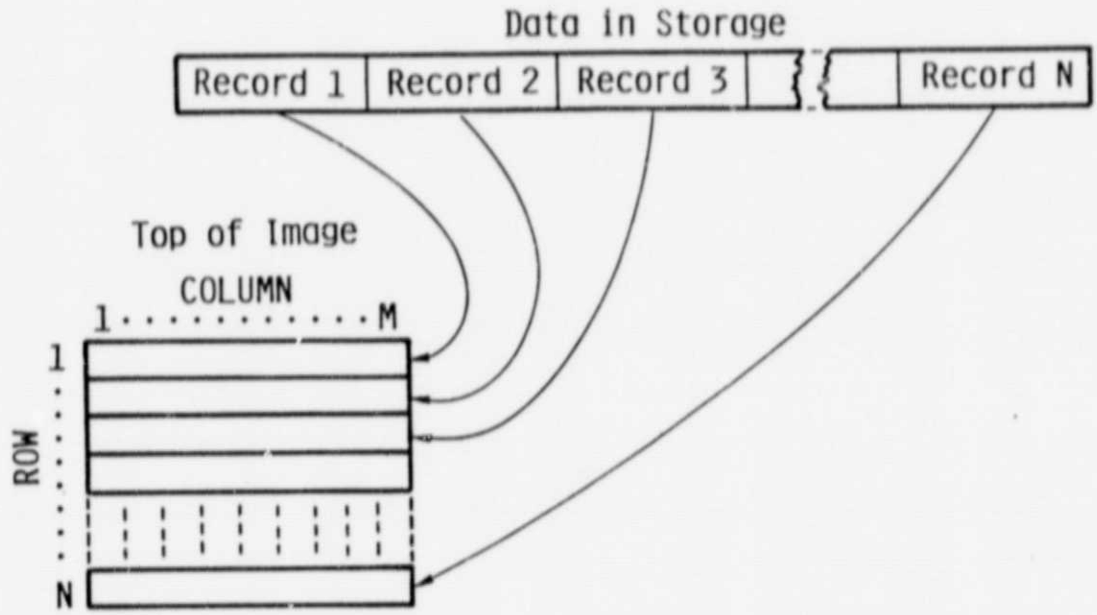


Figure 2.4 Sequential storage of image data

This filter is better shown in the section on the equal-weighted filter, but this simple example program makes the queuing operation more obvious. In the next section of this report, the performance criteria for the digital filters are presented, and the development of the edge figure of merit algorithm is outlined.

3.0 Performance Criteria

All of the filters compared in this study improve the signal-to-noise ratio of SAR imagery. However, the degree by which edges are degraded differs from one filter to the next. This section outlines the procedure used in developing a "fair" test of the edge preserving qualities of each filter.

The most difficult question asked when developing a "fair" test deals with the amount of filtering done in the homogeneous areas of the image. This is an important consideration, since a filter might appear to perform much better based solely on the results of the edge figure of merit (EFM) test. The EFM algorithm used in this study is based on the mean-square distance EFM developed by Pratt (5) defined as

$$F = \frac{1}{\max(I_I, I_A)} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha d^2(i)} \tag{3.1}$$

where I_I and I_A represent the number of ideal and actual edge map points, α is a scaling constant, and d is the separation distance of an actual edge point normal to a line of ideal edge points. The scaling constant was chosen to be $\alpha=1/9$ to provide a relative penalty between smeared edges and isolated, but offset, edges.

Since filter parameters differ widely from one algorithm to the next, a filter may not do as much smoothing in the homogeneous areas as another filter. If too little smoothing is performed, then the presence of noise will cause the detection of false edges, leading to a lower EFM. To validate the results of this study, each filter was applied so that the amount of filtering performed in homogeneous regions was approximately equal. Equal amounts of filtering were most important in the adaptive filters where a great deal of flexibility exists in the filter characteristics. For more rigidly defined filters, the window size is used as a common factor, while it still might be true that a filter operating on a 5 x 5 local area of one type does as much filtering as a 7 x 7 filter of another type.

The input images used in this study were 144 rows by 144 columns, containing only one vertical edge near the center of the image. The edge ratios used in this study were 3, 6, and 9 dB, with the number of independent looks equal to 1 and 4. A total of 6 images were processed by each filter for various window sizes.

Once the filtered images were produced, they were processed using a differential operator (see App. A). For this study the 2 x 2 Robert's gradient was utilized. The gradient images became the input images for the edge figure of merit program which generates a binary image by thresholding the gradients. That is, if a gradient point was found to be greater than or equal to an edge threshold, that point was set equal to a value representing the existence of an edge. Otherwise the point was set equal to zero. This operation resulted in the generation of an "edge map" which, when displayed as an image, shows edges as bright areas and homogeneous areas as dark. Since the original images contained only one edge, the optimum result would be a single vertical line at the edge's location in the image. In practical cases, however, the actual edge is not completely identified.

The threshold chosen in the generation of the edge map radically influences where edge points are identified, since too low a threshold highlights smaller gradients which might be false edges, while too high a threshold may miss some edges which actually exist. Therefore the edge figure of merit routine has two distinct, yet related tasks. The best threshold is sought in order to obtain the best edge map and the highest EFM, while the EFM is used as a way of determining the best threshold.

The EFM routine implemented for this study uses a quadratic search for finding the best threshold in order to minimize the number of passes through the images, since each EFM calculation for a particular threshold requires processing the entire image. Once the best threshold was calculated for the 3 dB image, the same threshold was applied to the 6 and 9 dB images. These calculations allowed comparison of the filter's ability to preserve low contrast edges. The optimum threshold was also found for the 6 and 9 dB images so that further comparisons could be made. As an additional factor of comparison, the amount of computer processing time was recorded, yielding a relative measure of cost. The results are presented and discussed in Section 5. In the next section, the development of each filter is examined separately.

4.0 FILTER IMPLEMENTATION

In this section, each filter compared in this study is discussed separately. Complete program listings are given at the end of the report in Appendix A. The filters presented in this section were written, tested and implemented using a HARRIS 230 minicomputer and FORTRAN 77. Some routines were converted from existing programs, while others were designed from information supplied in reports. (Specific source references are made where applicable.) In each case, an attempt was made to make the algorithms as general as possible with emphasis on readability. To achieve both generalizability and readability, the programs should be viewed as a package, since program layout and structure are common for most of the filters. Computational efficiency was an important consideration in the original design, and while specific algorithms could be optimized slightly, the common design approach does yield a usable product.

4.1 The Equal-Weighted Filter

The equal-weighted filter, often referred to as a box filter, is probably the most widely used spatial filter. The term "equal-weighted" describes the filter weighting function applied over each local area. In this filter, all points in the area of the window are each weighted equally, resulting in the average for the area being the output point. The second term, "box filter," views the filter as a discrete two-dimensional convolution with a box function. This is given by the relation

$$f(i,j) * g(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)\, g(x-m, y-n) \quad (4.1)$$

for i=0,1,2,...,M-1 and j=0,1,2,...,N-1 where g(i,j) represents the SAR image as a two-dimensional function and f(i,j) is the box function. The MxN array given by this equation is one period of the discrete, two-dimensional convolution (7). The equal-weighted filter performs a moving average over the entire image, smoothing edge points as well as homogeneous areas. This procedure makes the implementation of the filter straightforward, since only one pass through the filter window is required for each local neighborhood. It should be noted that this filter program is a straightforward application. The code can be optimized by moving some operations out of the innermost loop, resulting in a slight increase in efficiency. This optimization was not done here in order to maintain program readability.

4.2 The Median Filter

The median filter is another commonly used filter, but in many respects it is quite different from other filters. The operation performed by the median filter is to replace an image point by the median value of the points in the local area. That is, for the points contained within the filter window, the point is found for which there are an equal number of pixels with lower intensities as there are pixels with higher intensities.

In developing this filter, the question of window size must be addressed. Completely different programming approaches are required in order to optimize the filter for either small or large window sizes. This consideration is important, since the median filter, though simple in concept, requires a significant amount of computational overhead. For a small window size, it is easiest to sort the points and retreive the median. For larger window sizes, the increasing number of points makes this multi-pass approach very inefficient, and it becomes more desirable to generate a histogram of the points and sum counts until a median is found. The trade-off in central processing unit (CPU) time is summarized below for the relatively small 144 x 144 pixel images used in this study:

WINDOW SIZE	SORT METHOD	HISTOGRAM METHOD
3x3	23.16 sec	67.08 sec
5x5	90.64	79.14
7x7	271.37	96.55

Larger images increase the need for computational efficiency. Since a more general approach was desired for this study, the histogram method was implemented.

4.3 Lee's Edge Filter

Most filters developed for the enhancement of SAR images take a general approach to the suppression of speckle; no special consideration is given to the filtering of edge areas. The local statistics algorithm developed by Lee (2) attempts to identify edges so that less smoothing can be done in these areas.

To determine if an edge is present for a local area of the image defined by the filter window, the local statistics are first calculated. An edge is defined as a point of transition between two areas of differing properties. The property examined by the local statistics algorithm is the pixel intensity. If an edge exists within the local area, then a transition is present between relatively high and low pixel intensities. This results in the local area having a higher pixel variance. In this manner edges may be identified as being present within the local area by establishing a threshold for the variance.

Lee also presented a statistical model based on the signal dependency of speckles (3). Using the model given by (Eq. 1.1), an extension may be made to find the a priori mean and variance

$$x = \frac{\bar{z}}{\bar{v}} \tag{4.2}$$

and

$$Var(x) = \frac{Var(z) + \bar{z}^2}{Var(v) + \bar{v}^2} \tag{4.3}$$

where $\bar{z}$ and Var(z) are approximated by the local mean and variance of the speckle corrupted image. The parameters $\bar{x}$ and $\bar{v}$ are the means of the desired signal and the noise, while var(v) is the variance of the speckle noise. By linearizing the observed pixel z using the first-order Taylor series expansion about $(\bar{x},\bar{v})$:

$$z = \bar{v}x + \bar{x}(v - \bar{v}) \tag{4.4}$$

and

$$\begin{aligned} Var(z) &= E[(xv - \bar{x}\bar{v})^2] \\ &= E[x^2]\, E[v^2] - \bar{x}^2\bar{v}^2, \end{aligned} \tag{4.5}$$

which can be further simplified if the window is assumed to cover an area of constant average intensity. Therefore,

$$E[x^2] = \bar{x}^2 \tag{4.6}$$

and

$$\begin{aligned} Var(z) &= x^2(E[v^2] - \bar{v}^2) \\ &= \bar{x}^2 \; Var(v) \end{aligned} \tag{4.7}$$

$$Var(v) = \frac{Var(z)}{\bar{z}^2} \tag{4.8}$$

Equations (4.5) through (4.8) are used to justify the multiplicative noise model, while also providing a simplified expression for Var(v) which is needed to complete Eq. (4.4). From Eq. (4.7) it can be further determined that

$$Var(v) = \frac{1}{N} \tag{4.9}$$

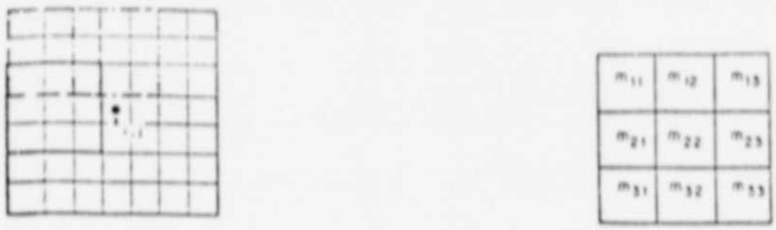
where N is the number of looks for the SAR image. This information, along with the assumption that v=1 and Eq. (4.2), leads to the final equation for the estimation of x,

$$\hat{x} = \bar{x} + k(z - \bar{x}) \tag{4.10}$$

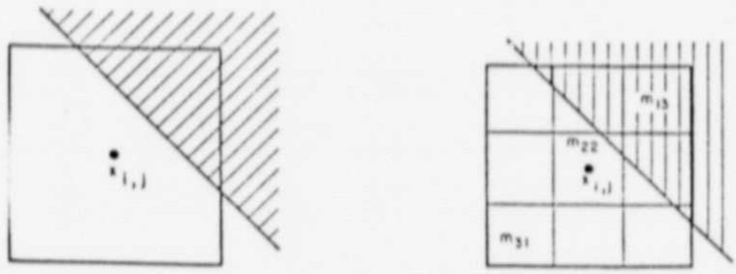
with

$$K = \frac{Var(x)}{\bar{x}^2\, Var(v) + Var(x)} \tag{4.11}$$

This allows the actual signal to be estimated for a pixel by knowing the local mean and local variance for the filter window and also knowing the average number of looks for the SAR image. The best results were achieved by setting the number of looks parameter equal to an exact value calculated from the statistics for the SAR image being processed. The edge preserving quality of Lee's filter is achieved through his use of the local statistics of the homogeneous areas of the image. If the local area is determined to be homogeneous based on the variance test, then the statistics for the entire window area are applied to the model in obtaining the estimate for the signal. If an edge is present, then its orientation is found by applying a 3x3 gradient mask to a 3x3 array of subareas calculated from the window. Fig. 4.1 shows the steps taken in obtaining the edge orientation. Once the edge position and orientation are known, statistics are calculated for the homogeneous portion of the local area. The new statistics are calculated by masking the edge points in the window when calculating the new mean. These masks are given in fig. 4.2. These statistics are applied to the noise model to obtain a new estimate for the signal. Since this new estimate is only dependent on the homogeneous area, there is less degradation of the edge information.



Formation of 3x3 subarea means from a 7x7 window.



After a 3x3 gradient mask has been applied to determine the edge orientation, the pixels perpendicular to the edge are compared to one another to determine on which side of the edge the center pixel lies.

Fig. 4.1 Steps used in determining edge orientation. (2)

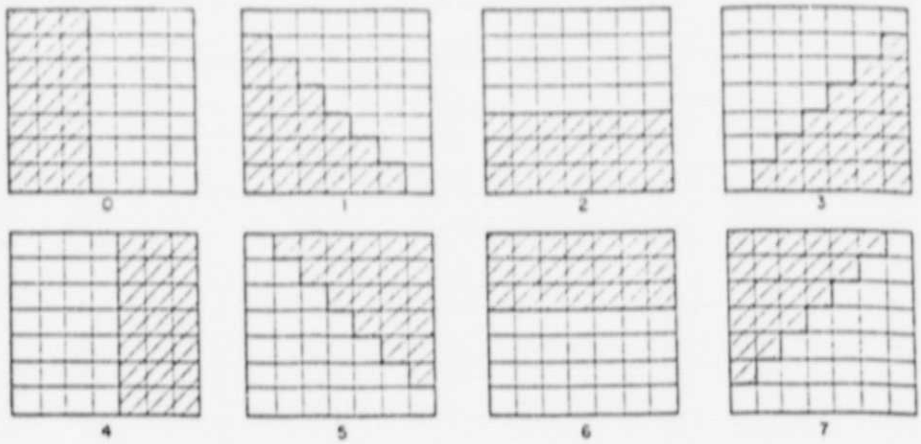


Fig. 4.2 Edge masks used in calculating new window statistics. (2)

The entire procedure is flowcharted in fig. 4.3.

For the purposes of this study, the implementation of Lee's filter was designed with an emphasis on flexibility. The window size was allowed to vary so that comparisons could be made based on this parameter. The edge threshold value, used as an input parameter for determining the presence of edges, is modified based on the results of previous executions of the program which report the percentage of the image assumed homogeneous. Since the images used in this study contain an exactly defined number of edge points, this allowed the results to be kept consistent for all applications of the filter.

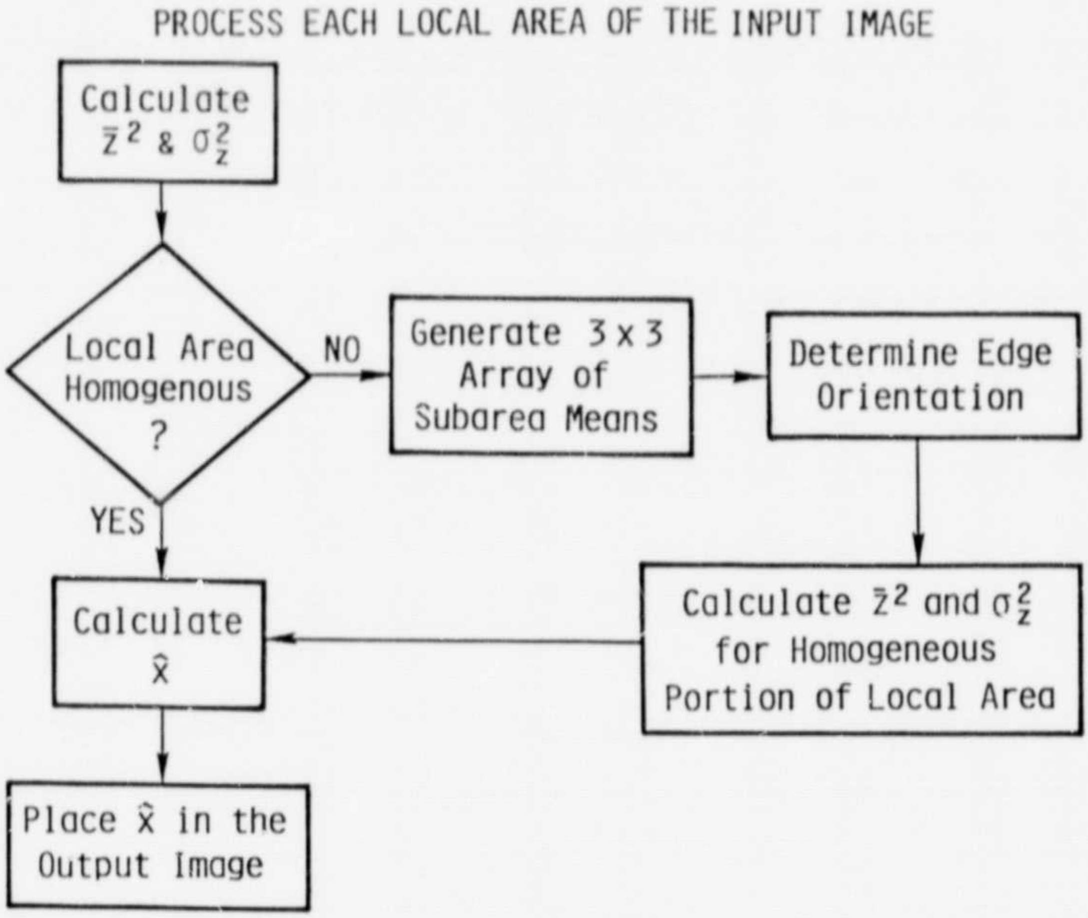


Figure 4.3 Flowchart of Lee's filter

4.4 The Adaptive Filter

The operation of the adaptive filter is as the name implies; the filter is adapted to the local area based on some criteria. The filter developed in this study uses the local statistics of the area defined by the filter window in order to determine the filter window weightings. (1) For each local area of an image, the local number of looks is calculated using the relationship

$$N_\ell = \frac{\bar{z}^2}{\mathrm{Var}(z)} \tag{4.12}$$

where N_ℓ is the local number of looks, $\bar{z}$ is the mean of the area defined by the window, and var(z) is the variance of the local area. Using the value N_ℓ, an index into an array of filters is chosen. The filter weightings range from an equal-weighted filter to a filter which does no averaging. The filters defined within this range are weighted using

$$F(x) = e^{-\alpha|x|} \tag{4.13}$$

The rate of decay of the exponential determines how heavily surrounding pixels are weighted in the local average and, hence, how much smoothing is done for the pixel being processd. Figure 4.4 illustrates this for the one-dimensional discrete case, but an extension can easily be made to two dimensions.

The adaptive filter preserves edge information by applying a filter window which affects less averaging for the local areas with a lower N_ℓ (higher Var(z)), while homogeneous areas receive more averaging, since a more uniformly weighted window is chosen for those areas.

Several different parameters are involved in the definition of the adaptive filter, allowing a great deal of flexibility in the characteristics of the filter. The window size is variable, while the number of filters to be used is also variable. A larger number of filters allows a more continuous smoothing effect. To generate the filters, the first and last filters are generated, with the first filter equal-weighted, and the last filter weighted so that no averaging is done. The weightings of the filter windows within this range of filters were specified by the value, alpha, which relates to the rate of decay of the exponential. Alpha is first calculated, based on the relationship,

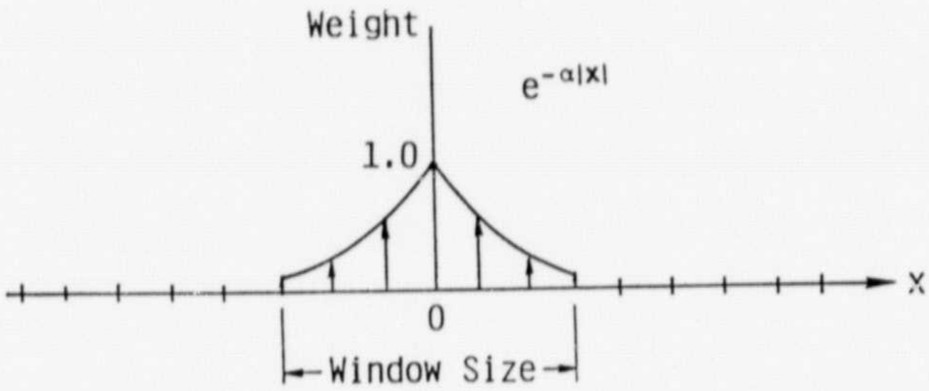


Figure 4.4 Exponential filter weighting

$$\alpha = \frac{2}{W} \tag{4.14}$$

where w is the equivalent resolution for a box filter. Alpha is then found by evaluating the integral

$$W = 2\int_0^{\infty} e^{-\alpha x}\, dx \tag{4.15}$$

Quantizing alpha gives

$$\alpha = K_0 \text{ Index} \tag{4.16}$$

where K_0 is a constant evaluated for the case where $\alpha = .5$ when $N_\ell = N$, the number of looks for the image. The motivation behind these choices of values is based on the constraint that when the local variance (described by N_ℓ) is equal to the average of all the local variances (described by N), then the filter applied to this area should be the filter in the middle of the range of filters. This yields

$$K_0 = \frac{2}{W}\ \frac{\Delta_N}{N} \tag{4.17}$$

with

$$\Delta_N = \frac{\text{Max}(N_\ell) - \text{Min}(N_\ell)}{\text{\# Filters}} \tag{4.18}$$

Collecting terms produces

$$\alpha = \frac{2}{w} \frac{\Delta_N}{N} \text{ Index} \tag{4.19}$$

which describes the filter shape for each filter. In order to generate a complete range of filter shapes, the program uses the user-supplied parameters, w and Δ_N. The other filter characteristic which may be modified is the rate of filter usage. By calculating the local statistics, the local number of looks may be found from

$$N_\ell = \frac{\bar{z}^2}{Var(z)} \tag{4.20}$$

which is used to select the filter to be applied to the area,

$$\text{Filter} = \#\text{ Filters} - \frac{N_\ell}{\Delta_N} \tag{4.21}$$

Here the only parameter to be varied is Δ_N, which also was used in the filter generation. The parameters, w and Δ_N, are both factors used in the generation of the filter weightings, while only Δ_N is needed for the determination of the filter usage. To accommodate this parameter dependency, the filter usage should first be determined. Once this characteristic is resolved, then the amount of averaging desired for the image may be regulated by varying the parameter w.

4.5 The Edge Adaptive Filter

This filter combines the edge-locating attributes of Lee's edge filter with the filter flexibility of the adaptive filter. Once Lee's edge filter determines the presence of an edge within the local area, the edge orientation can be found. With this information, it is possible to develop an adaptive filter algorithm which uses non-isotropic filter windows so that the edge pixels within the local area receive less averaging than those of the homogeneous portion within the window.

Assume that for a particular local area of an image, a vertical edge has been found to exist on the left side of the window area (Fig. 4.5). Depending on the local statistics for the window region, the adaptive filter of Section 4.4 would apply an isotropic filter window to the area. However, since it is known which pixels are part of the edge, a non-isotropic filter window may be defined which does less averaging for the edge points. An

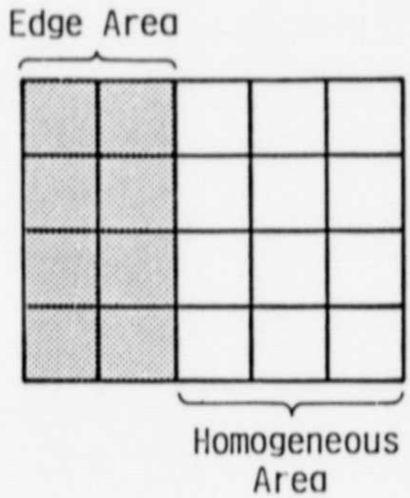


Figure 4.5 A vertical edge is present on the left side of the local area.

exponential weighting function similar to the one used for the adaptive filter is utilized. However, the edge adaptive filter applies an exponential weighting function with a steeper decay on the pixels containing the edge (Fig. 4.6). This weighting is accomplished by first calculating a new set of statistics for the homogeneous portion of the window, establishing the weighting factor for the right-hand side of the window. For the remaining pixels within the window, filter weightings are applied using weightings for the filter defined by

$$F_E = \#\ \text{Filters} - F_H + 1 \qquad (4.22)$$

where F_H is the filter index chosen for the homogeneous portion of the local area, and F_E is the filter to be applied to the edge area.

The parameters needed to define the edge adaptive filter are very similar to those used for Lee's edge filter and for the adaptive filter.

4.6 The Sigma Filter

The chief attraction of the sigma filter developed by Lee (6) is its simplicity and speed. The pixel to be processed is replaced by the average of those neighboring pixels having their gray level within two standard

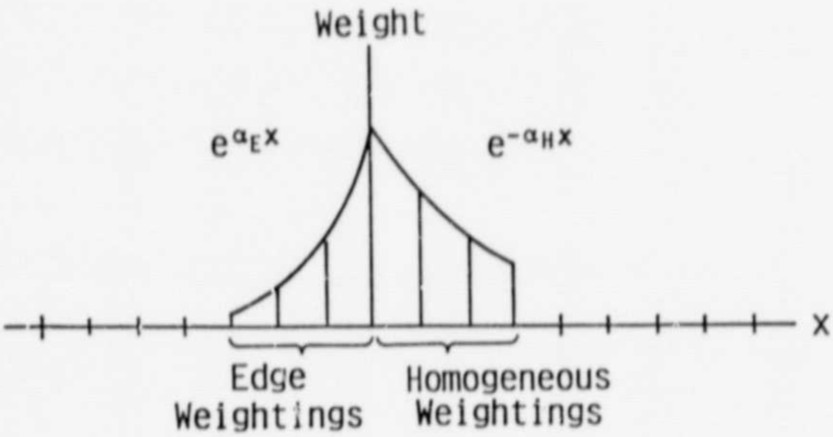


Figure 4.6 Non-Isotropic window weighting

deviations from that of the concerned pixel.

The sigma filter is based on the multiplicative noise model for speckle in SAR images as presented in Eq. (1.1) with the assumption that the multiplicative noise, $v_{i,j}$, has a mean of 1 and a variance $Var(v)$. From this it follows that

$$\bar{z} = \bar{x}\,\bar{v} = \bar{x} \tag{4.23}$$

and

$$\begin{aligned} Var(z) &= E[(xv - \bar{x}\bar{v})^2] \\ &= E[x^2]\,E[v^2] - \bar{x}^2\bar{v}^2 \end{aligned} \tag{4.24}$$

For a small local area, the signal may be assumed nearly constant, allowing $E(x^2)=\bar{x}^2$, which reduces $Var(z)$ to

$$Var(z) = \bar{x}^2\,Var(v) \tag{4.25}$$

or

$$\sigma_v = \frac{\sigma_z}{\bar{x}} = \frac{\sigma_z}{\bar{z}} \tag{4.26}$$

Equation 4.26 describes the standard deviation of multiplicative noise in SAR imagery as the ratio of the standard deviation of z and the mean of z.

It is assumed that $z_{i,j}$ is the a priori mean of $x_{i,j}$ and also that pixels in the window with gray level within two standard deviations from $z_{i,j}$ are from the same distribution. Since the speckle noise is multiplicative in nature, the two-sigma intensity range from Eq. (4.26) is $(z_{i,j} - 2\sigma_v\ z_{i,j},\ z_{i,j} + 2\sigma_v\ z_{i,j})$. The average of pixels in this intensity range replaces the center pixel as the smoothed value of $z_{i,j}$ (6).

The implementation of the sigma filter requires that the relative limits for the distribution of σ_v be established at the onset of program execution. In the filter implemented by Lee, a normal distribution was assumed. For comparison, a chi-square distribution was also modeled in this study. Additionally, a threshold K is established to deal with the presence of spot noise in the filtered image. The retention of spot noise is due to the fact that the gray level of the spot noise is significantly different from its neighborhood pixels. If the total number of pixels within the two-sigma range is less than or equal to K, the center pixel is replaced by the average of its four neighbors.

The sigma filter was applied to the test image using a wide range of configurations. Both chi-square and normal distributions were modeled, while the threshold K was set to 3 for the 7x7 window and 2 for the 5x5. Results are presented for a single pass and for a second pass of the filter.

5.0 RESULTS, CONCLUSIONS AND RECOMMENDATIONS

This final section presents the results of the filter performance comparisons. The results are shown in plots so that comparisons can be more easily made. The edge figure of merit value is expressed as a percentage shown on the vertical axis. The EFM is plotted versus the edge contrast ratio for varying filter window sizes and number of looks. The EFM is also shown with respect to the window size for varying edge ratios and number of looks. These results are summarized in fig. 5.25 which ranks the six filters using several different criteria. A value of one indicates the best performance by a filter.

There were two methods used in obtaining the EFM values. The first method calculated the best edge threshold for the 3 dB image and applied the same threshold to the 6 dB and 9 dB images to determine their EFM. The second method found the optimum threshold at each edge ratio and used this threshold in calculating the EFM. Results are plotted for both methods.

Figures 5.23 and 5.24 show the central processing unit (CPU) time required of each algorithm. This is first shown in terms of the actual time (fig. 5.23) reported for each algorithm using various window sizes. The values are then scaled and shown in fig. 5.24 as a relative time versus window size. The relative time factor is determined by dividing the time required for each algorithm by the time required for the equal-weighted filter of the same size. The scaling factor varied with window size.

The results should be analyzed carefully, since the wide diversity of filtering methods leads to a large number of factors which should be considered in the evaluation of each filter. Several general observations can be made about the results. As expected, the higher contrast edges yield a higher edge figure of merit. Also, the four-look images, since they are not as noisy, show a higher EFM. These variations are independent of the filter type. However, allowing for this inherent EFM improvement, it is still possible to draw some conclusions about how well a particular filter type performs relative to edge quality. Another general observation is that all filters yielded results which were, to varying degrees, better than those obtained from the unfiltered images. As a final overall observation, it should be noted that for most filters an increase in the filter window size led to an increase in the edge figure of merit. This improvement can be explained by the fact that, since a larger window usually allows more averaging, fewer false edges are detected. For reasonably sized windows, the suppression of false edges is usually enough to outweigh the contrast loss caused by a larger window performing more smoothing.

5.1 Evaluation of Equal-Weighted Filter Performance

The results for the equal-weighted filter are important, since it is the most widely used spatial filter and may be used as a standard for comparison. The popularity of the equal-weighted filter is justified by the results. The filter, even though it makes no assumptions concerning edges, averaging all pixels equally, still yields a fairly good EFM when compared to the other algorithms. This performance is clearly evident from the results. For the low contrast edges, the equal-weighted filter provides results which are only a few percent higher than those of the other filters. (A percentage value given for comparison is a value difference read from the graph--not a ratio of the EFM for each filter.) For 6 dB edges, however, the equal-weighted filter is superior by as much as 20 percent. As the edge sharpness is increased to 9 dB, the results of the other algorithms improve dramatically, though the equal-weighted filter still earns a comparable EFM.

A comparison based on filter window size also shows the effectiveness of the equal-weighted filter. A 5x5 pixel window gives an EFM very nearly equal to that for the more complex filters, while the 7x7 window gives results which are as much as 20% higher than those of the other filters (fig. 5.2). A 9x9 equal-weighted window was not applied in this study, so a definite comparison cannot be made for large window sizes.

Fig. 5.25 shows that the equal-weighted filter performed best for the N=1 images. This conclusion is not as evident from the plots, since fig. 5.25 was based on an average for all the N=1 data. The fact that the equal-weighted filter obtained top ranking in this catagory is not surprising when it is considered that in order to prevent false edged from being detected, more averaging is required. The equal-weighted filter performs more averaging than any other type of filter using the same window size.

As a further benefit, when processing cost is considered a factor, the equal-weighted filter algorithm turns in the best times of all the filters considered. In some cases, the algorithm is over three times faster. This filter has another advantage in that it is very easy to apply, since the only parameter is the size of the window. This can sometimes be an important consideration when compared to the more complex edge filters utilizing several different parameters.

5.0 Evaluation of Median Filter Performance

The median filter is also quite common and simple to use. However, the results of the EFM were nearly the lowest of those from all the filters. This is evident from the plots which show that the median filter gives better results for larger window sizes and greater edge ratios. This improvement with window size and edge ratio is probably related to the nature of the EFM as discussed earlier in this section, since the improvement is not nearly so dramatic as for the other filters. Fig. 5.25 shows that in every catagory compared, the median filter earns a low ranking. In addition, the large overhead in processor time tends to discount this filter as a practical alternative to the other filters.

5.3 Evaluation of Lee's Filter Performance

Lee's edge filter compares well to the other filters. This was expected, since special processing is performed for edge areas. However, for smaller windows the algorithm does not do as well as the box filter. In fact the lowest overall EFM values are for Lee's filter using a 5 x 5 window (fig. 5.1 and 5.25). This poor performance is probably due to the fact that with a smaller window, there is a greater chance that a false edge will be detected by Lee's filter. This speculation receives some justification when it is noted that for the less noisy N=4 images and for more distinct edges, the equal-weighted filter is not as significantly superior to Lee's filter.

As the window size, edge ratio or number of looks was increased, the performance of Lee's algorithm improves dramatically as shown in the plots. In some cases, Lee's filter had an EFM as much as 40 percent greater than that of the other filters (fig. 5.5). A conclusion that may be made about this filter is that it is prone to the detection of false edges, but by using less noisy images (larger N) with higher contrast edges and a larger window size, quite good results can be attained.

The time for processing Lee's edge filter might be considered too long--especially since a larger window size and more CPU time is required in order to achieve the best results of the filter. It should also be noted that the CPU time shown for Lee's filter is somewhat misleading, since the parameters for Lee's filter were adjusted for the mostly homogeneous test images. These images are not very realistic, since most SAR imagery contains more edge information. Between 3% and 7% of each test image was known to contain edges while roughly 20% to 60% of an actual image may have edge information. For actual imagery, the required CPU time is much closer to that for the more time consuming filters compared in this study.

5.4 Evaluation of Adaptive Filter Performance

The adaptive filter also yields a high EFM, but unlike Lee's filter, achieves good results for the smaller window sizes. The adaptive filter earned the best ranking for low contrast edges (fig. 5.25), and when the rankings for all the comparison criteria are averaged, the result is 2.75 (The result is 2.0 if the computer time is not included in the average.).

Again, noisy images and low contrast edges tend to lead to inferior performance, since the adaptive filter also uses the local statistics method in order to determine the amount of smoothing to apply to an area. Larger window sizes improve the performance of the adaptive filter for these images.

The amount of CPU time required for the adaptive filter is quite large. This is especially true when compared to that of the equal-weighted filter, but also true when compared to the time required by Lee's filter. However, since Lee's algorithm requires a larger window size to get similar results, this disadvantage is slightly offset. For the adaptive filter, the processing time required for larger windows becomes even harder to justify, but some of the best results were achieved using this filter configuration (fig. 5.9).

A disadvantage of the adaptive filter is that it is relatively difficult to establish the filter characteristics. This disadvantage can be minimized by gaining experience with the filter, but often it can still take at least one initial pass with the filter before optimum filter characteristics may be determined.

5.5 Evaluation of Edge Adaptive Filter Performance

The edge adaptive filter gave some of the best results of all filters compared. For the N=4 images with a 5x5 window, this filter earns the best overall EFM (fig. 5.10). The improvement is slight, and in general the unmodified adaptive filter produces better results. As is shown in figure 5.25, the edge adaptive filter earned the best ranking for small window sizes, but the filter also obtained the worst ranking for noisy images. CPU time is not significantly higher than that for the adaptive filter, while the smaller window size allows good results without using a great deal of processor time.

5.6 Evaluation of Sigma Filter Performance

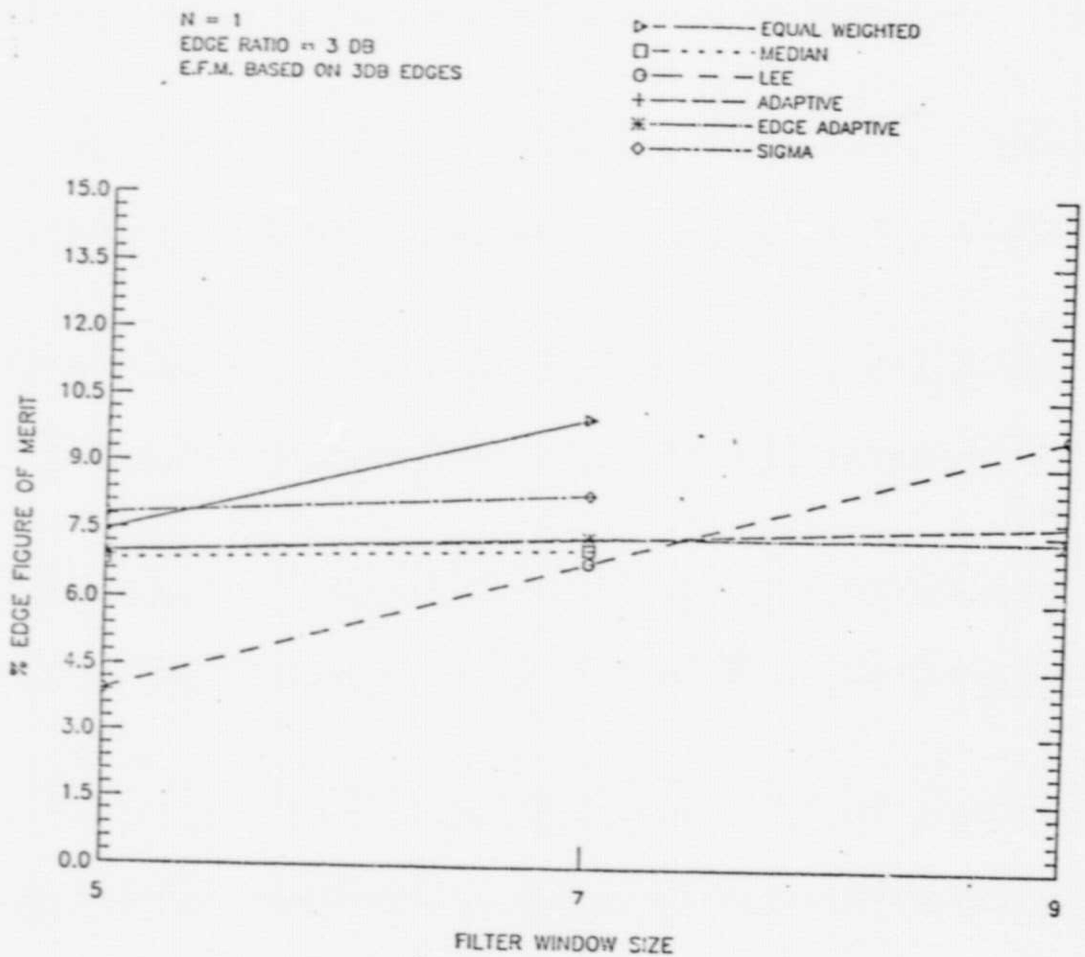
The results for the sigma filter are somewhat discouraging. This is shown in figure 5.25. Even though the filter is quite fast, two passes are required in order to receive results comparable to those of the other filters. Two passes more than doubles the amount of processing required, yet the results are still inferior. In fairness, it should be noted that the poor results are probably due to the presence of spot noise which the EFM routine detects as false edges. Lee gives several techniques for reducing spot noise, which if applied, might lead to better results for the EFM test.

5.7 Recommendations

Having reviewed each filter separately, several conclusions and recommendations can be made. Figure 5.25 cannot by completely relied on, since it was compiled from averages for all values given for each particular parameter. Optimum combinations of parameters are not shown by these rankings. In addition, rankings were not calculated for large window sizes.

If processing time is not a factor, then it can be assumed that a large filter window should be used, since a higher EFM results by using the larger windows. Lee's algorithm gives the best results for the noisier images, while the adaptive or the edge adaptive would be better for less noisy images. If a smaller window size is preferred, then the equal-weighted filter should be used for noisy images. If processing time is an important consideration, then the equal-weighted filter is clearly the best choice.

The main basis of comparison in this study was the edge figure of merit. Though the desirability of an algorithm should not be determined solely from this criterion, some indication is given as to how well each algorithm will filter edge areas.



N = 1
EDGE RATIO = 3 DB
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
15.0
13.5
12.0
10.5
9.0
7.5
6.0
4.5
3.0
1.5
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.1

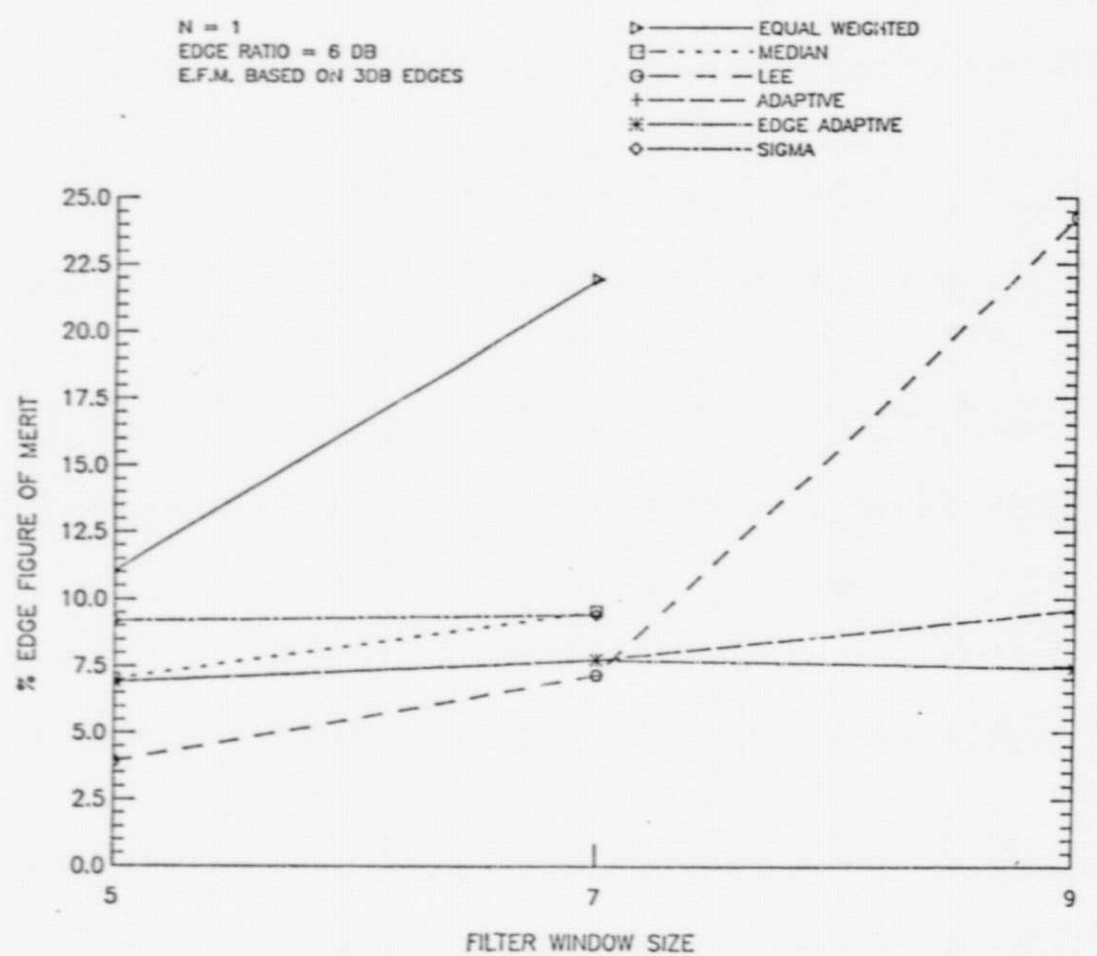

N = 1
EDGE RATIO = 6 DB
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.2

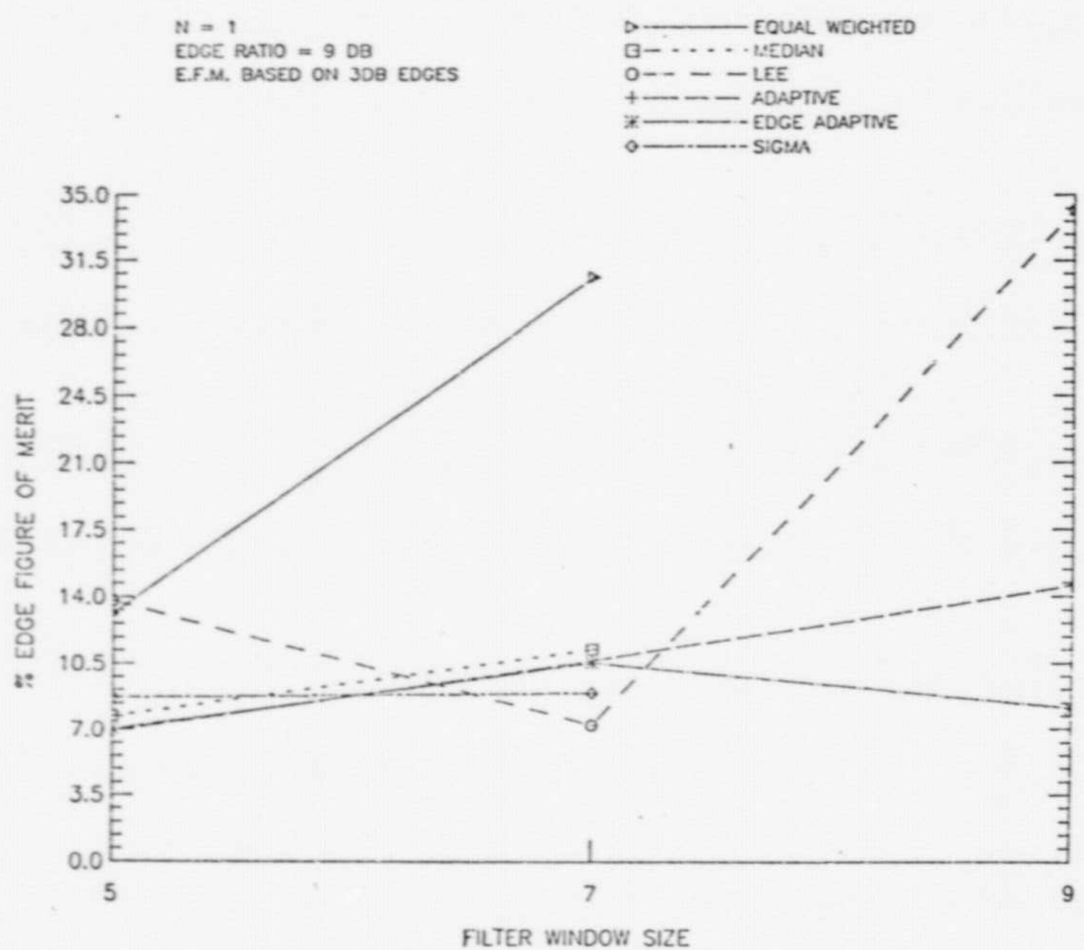
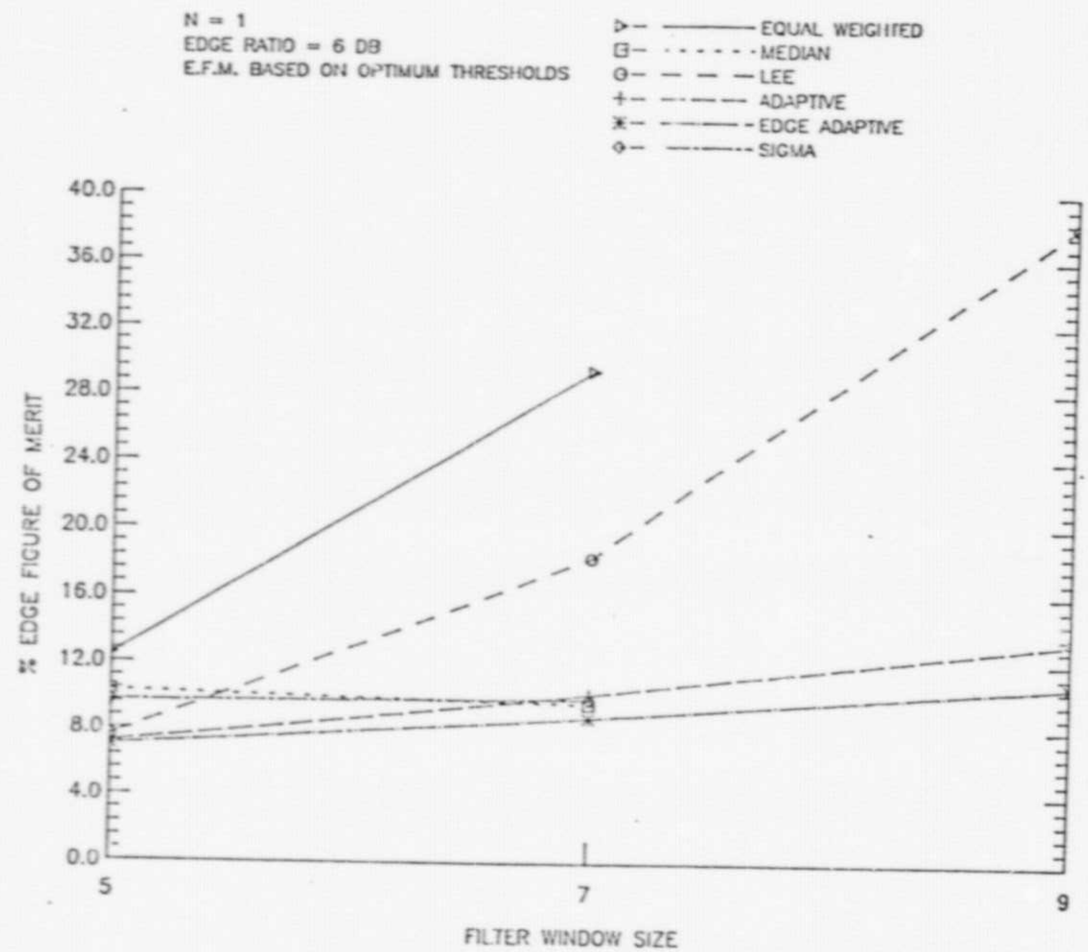
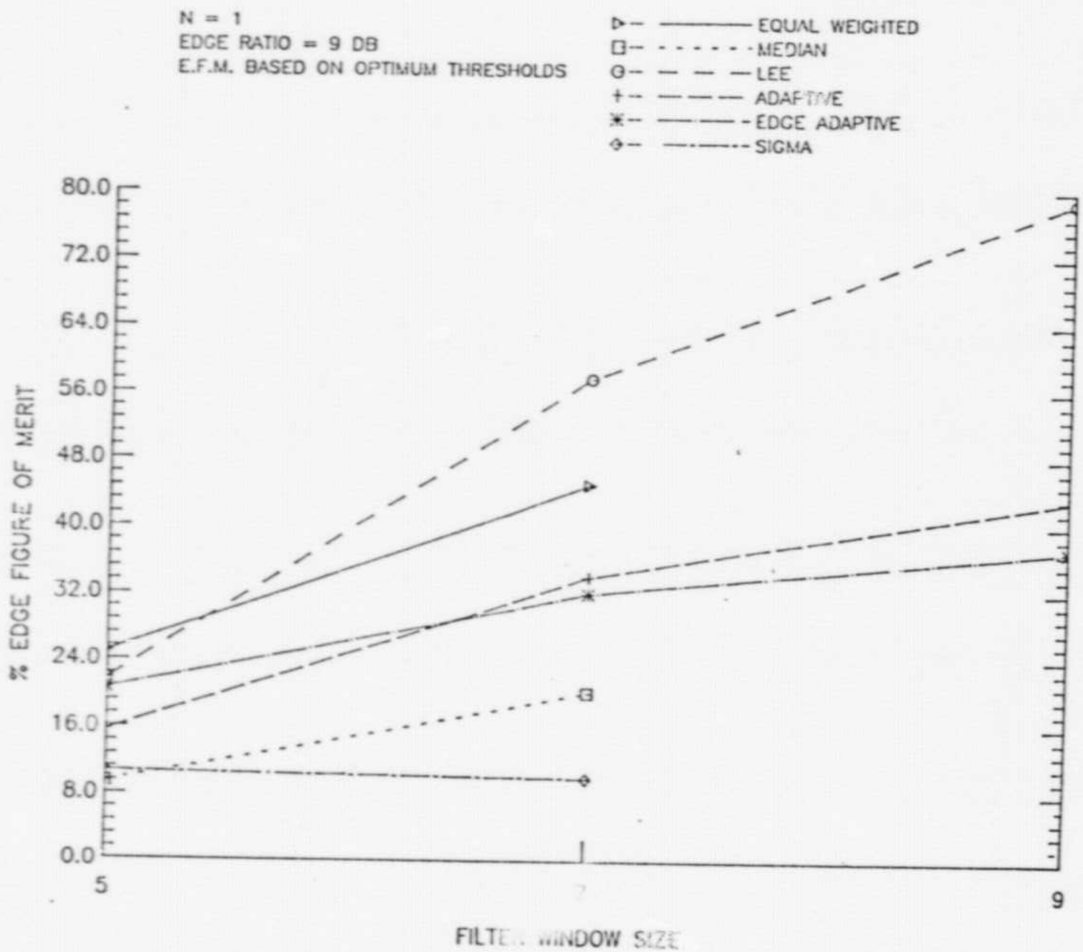

N = 1
EDGE RATIO = 9 DB
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
35.0
31.5
28.0
24.5
21.0
17.5
14.0
10.5
7.0
3.5
0.0
% EDGE FIGURE OF MERIT
5
7
9
FILTER WINDOW SIZE

Figure 5.3



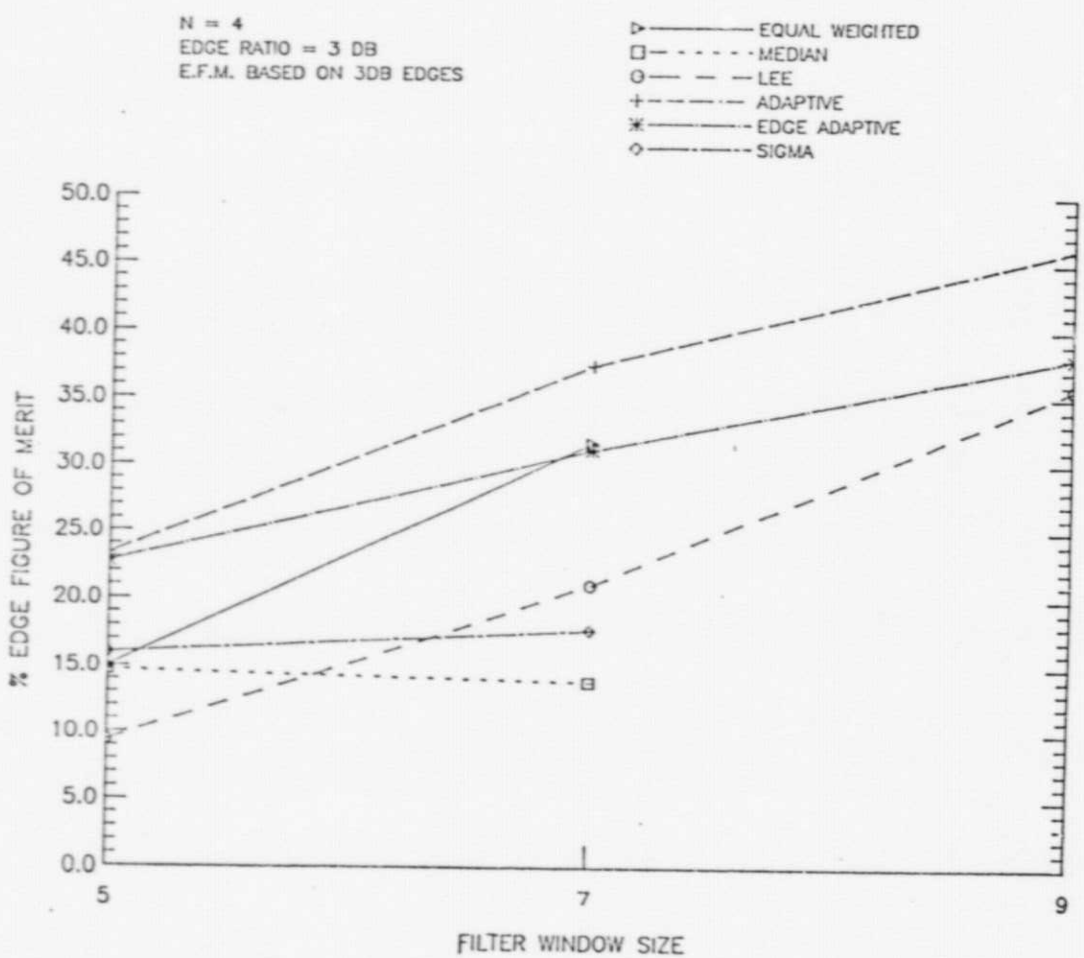
N = 1
EDGE RATIO = 6 DB
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
40.0
36.0
32.0
28.0
24.0
20.0
16.0
12.0
8.0
4.0
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.4



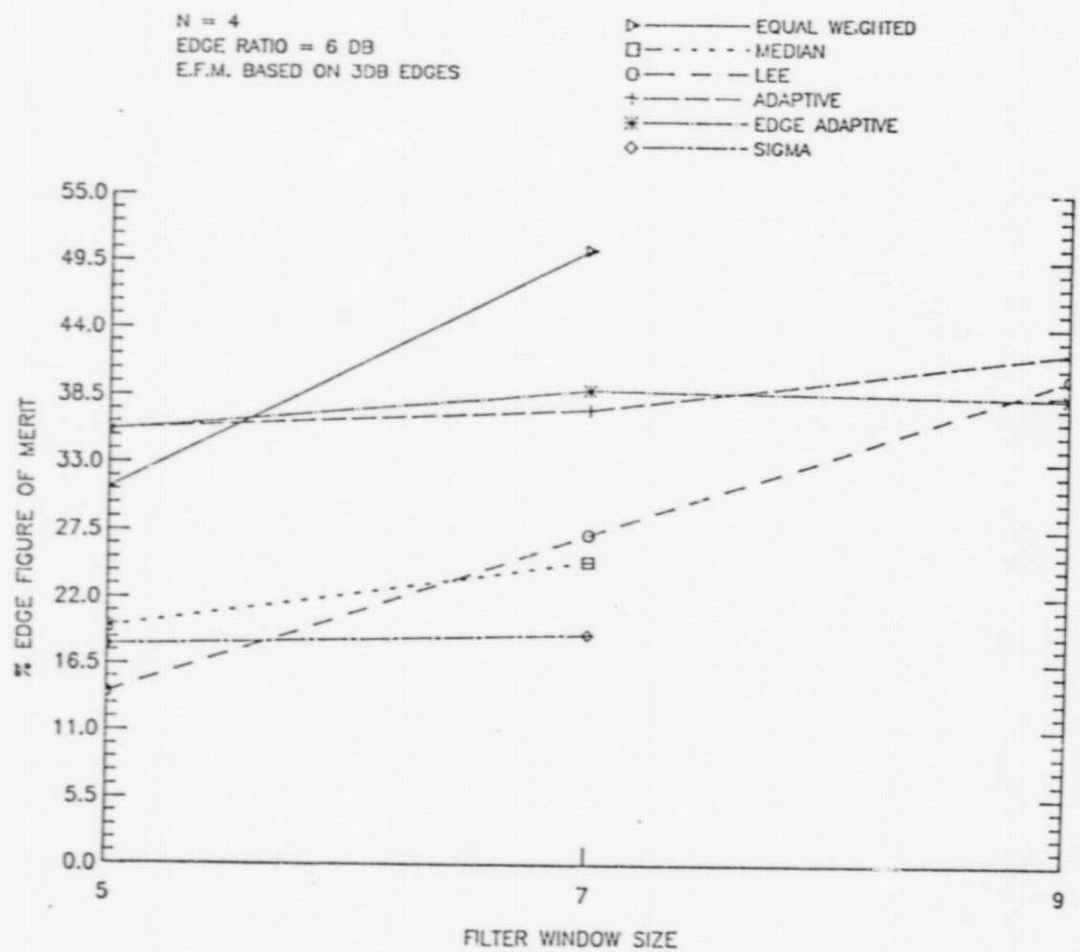
N = 1
EDGE RATIO = 9 DB
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
80.0
72.0
64.0
56.0
48.0
40.0
32.0
24.0
16.0
8.0
0.0
5
9
FILTER WINDOW SIZE

Figure 5.5



N = 4
EDGE RATIO = 3 DB
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
50.0
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.6



N = 4
EDGE RATIO = 6 DB
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
55.0
49.5
44.0
38.5
33.0
27.5
22.0
16.5
11.0
5.5
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.7

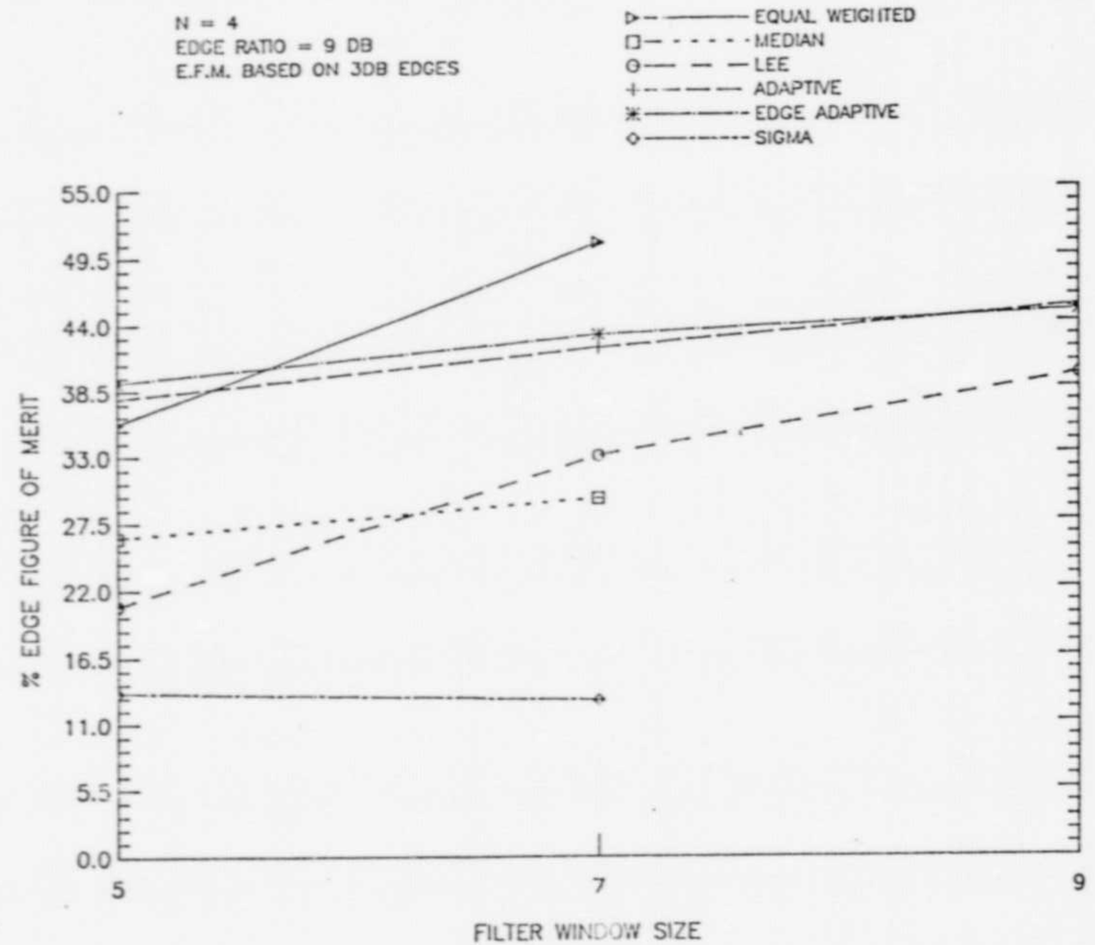

N = 4
EDGE RATIO = 9 DB
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
55.0
49.5
44.0
38.5
33.0
27.5
22.0
16.5
11.0
5.5
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.8

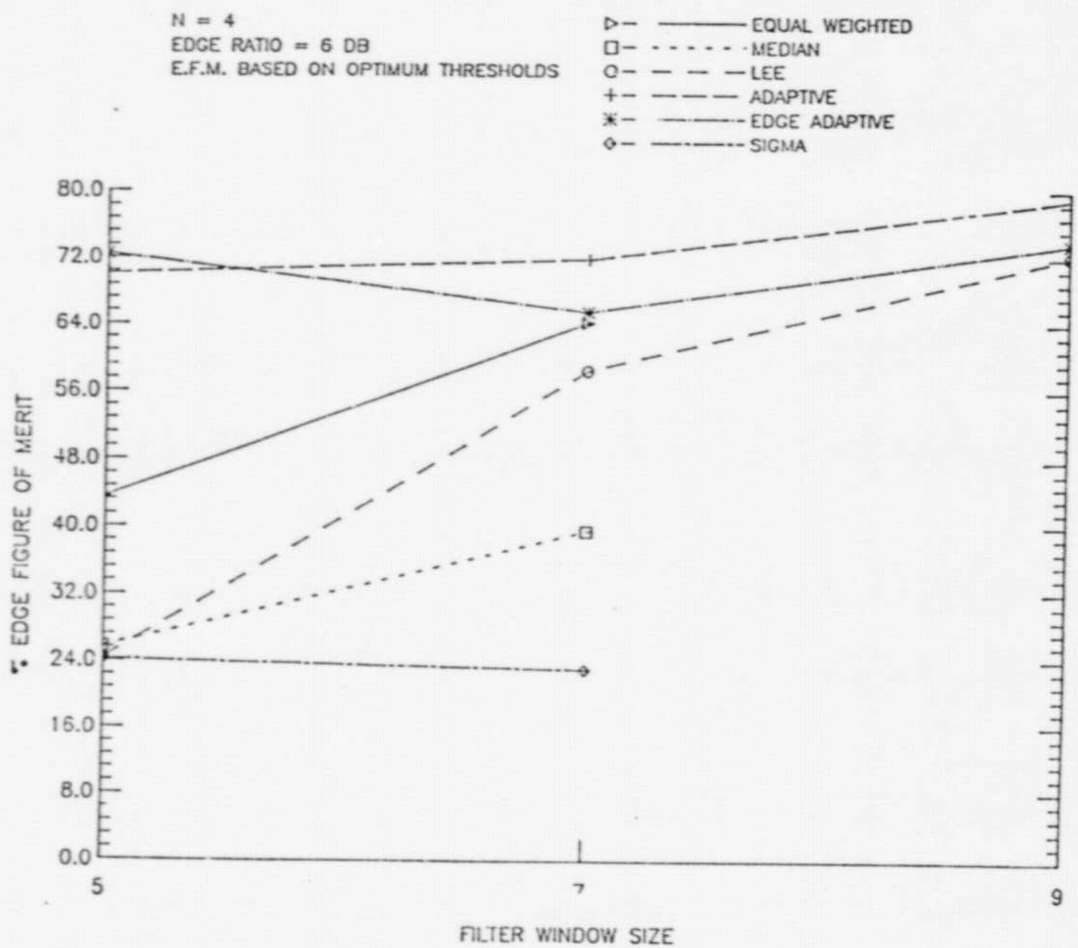

N = 4
EDGE RATIO = 6 DB
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
80.0
72.0
64.0
56.0
48.0
40.0
32.0
24.0
16.0
8.0
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.9

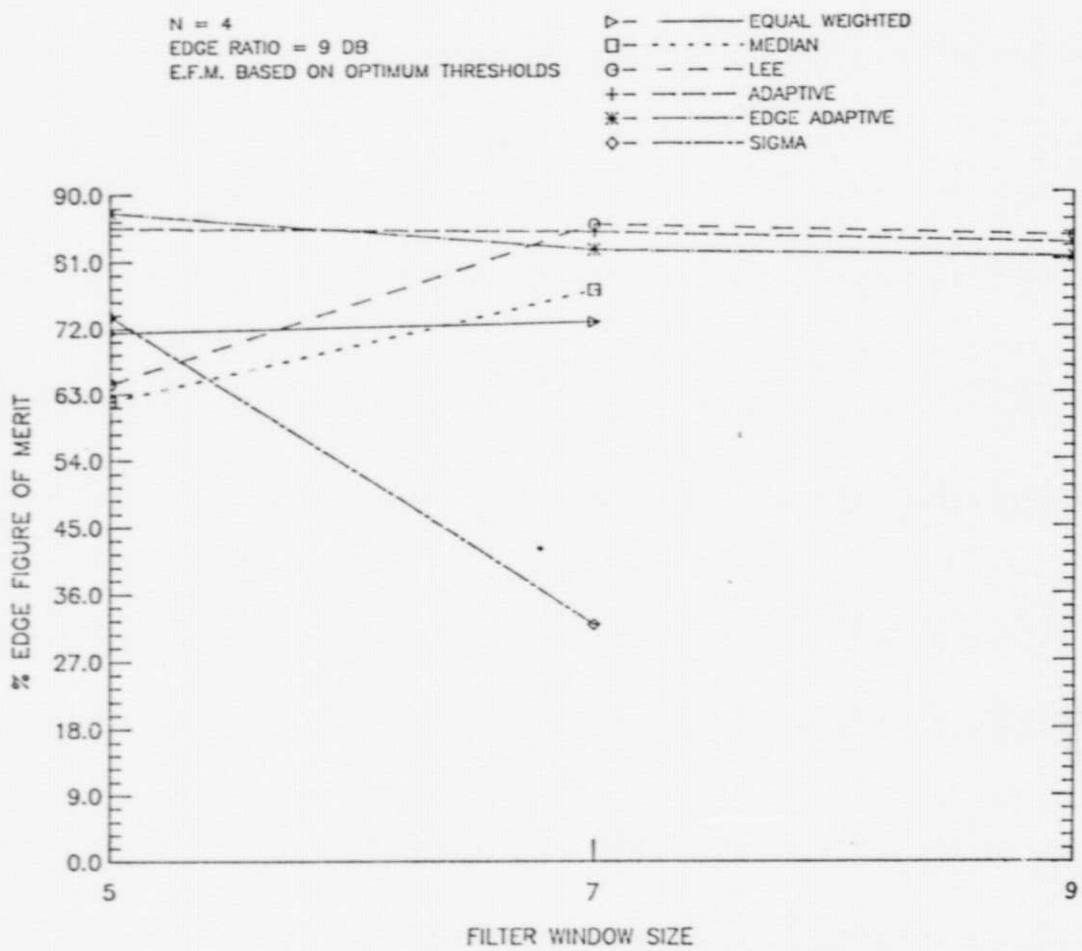

N = 4
EDGE RATIO = 9 DB
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
90.0
81.0
72.0
63.0
54.0
45.0
36.0
27.0
18.0
9.0
0.0
% EDGE FIGURE OF MERIT
5
7
9
FILTER WINDOW SIZE

Figure 5.10

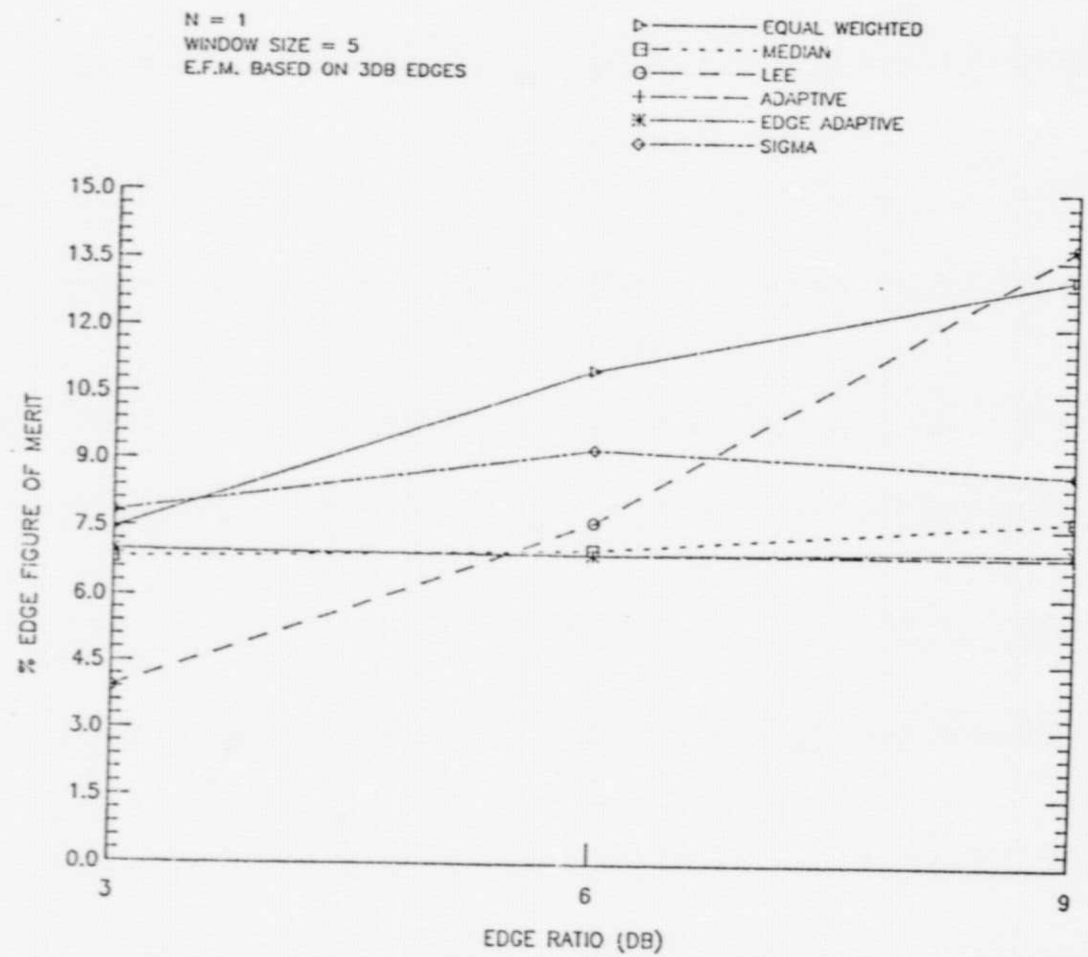

N = 1
WINDOW SIZE = 5
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
15.0
13.5
12.0
10.5
9.0
7.5
6.0
4.5
3.0
1.5
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.11

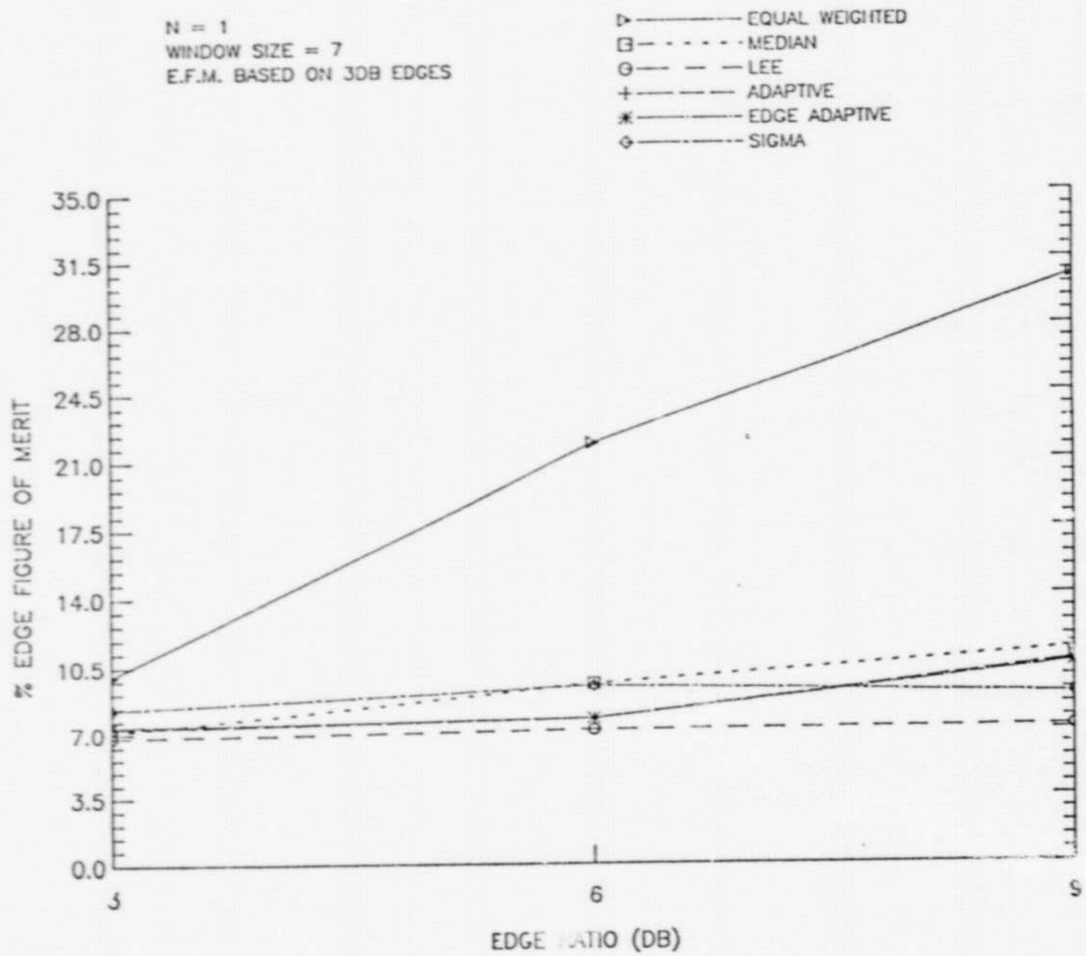

N = 1
WINDOW SIZE = 7
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
35.0
31.5
28.0
24.5
21.0
17.5
14.0
10.5
7.0
3.5
0.0
% EDGE FIGURE OF MERIT
6
EDGE RATIO (DB)

Figure 5.12

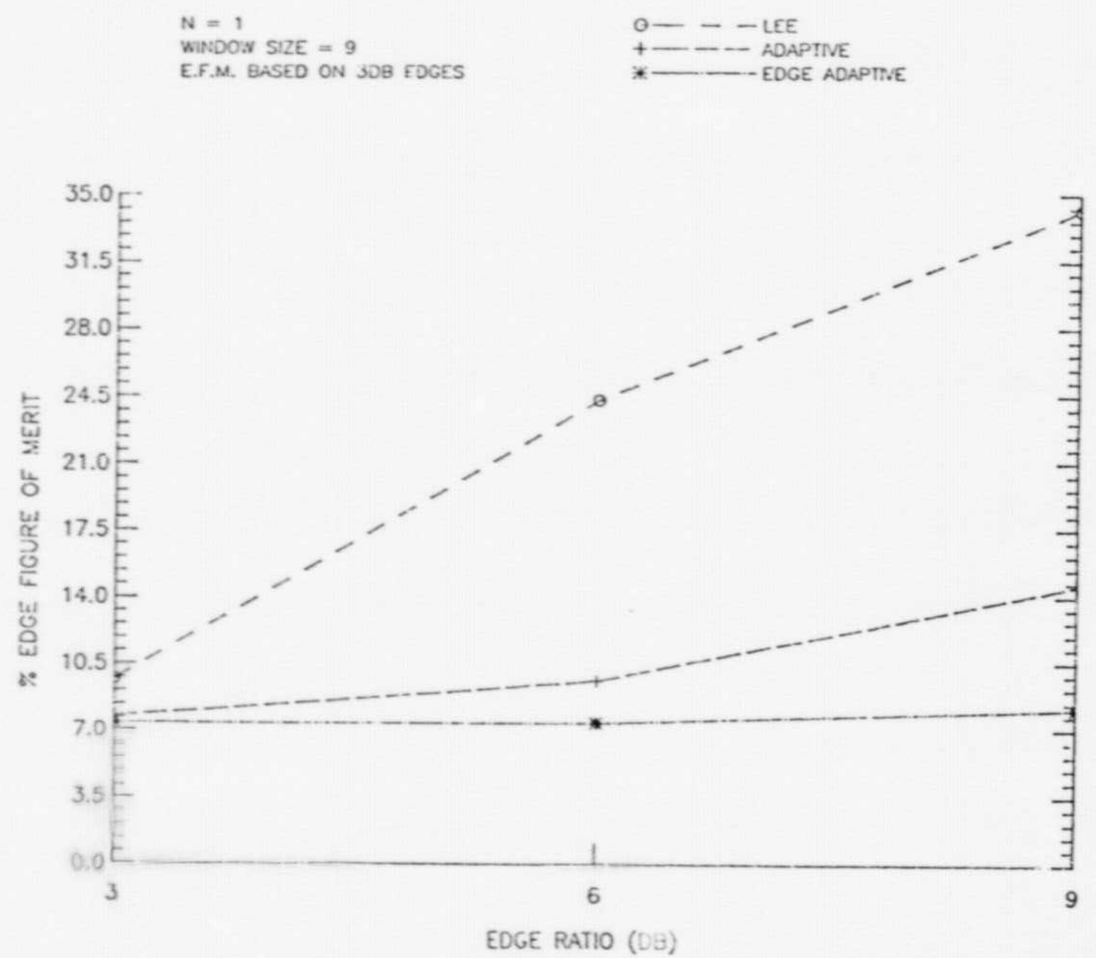

N = 1
WINDOW SIZE = 9
E.F.M. BASED ON 3DB EDGES
LEE
ADAPTIVE
EDGE ADAPTIVE
35.0
31.5
28.0
24.5
21.0
17.5
14.0
10.5
7.0
3.5
0.0
% EDGE FIGURE OF MERIT
3
6
9
EDGE RATIO (DB)

Figure 5.13

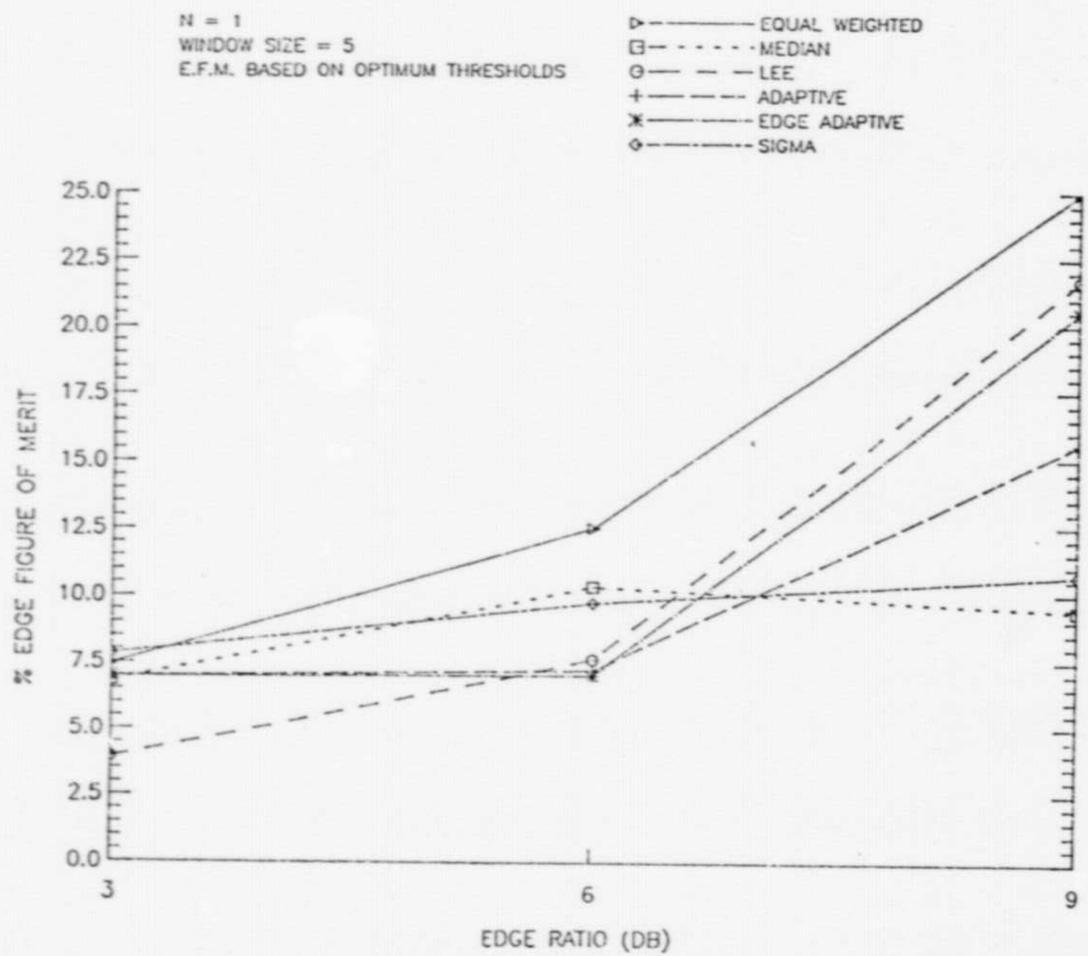

N = 1
WINDOW SIZE = 5
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.14

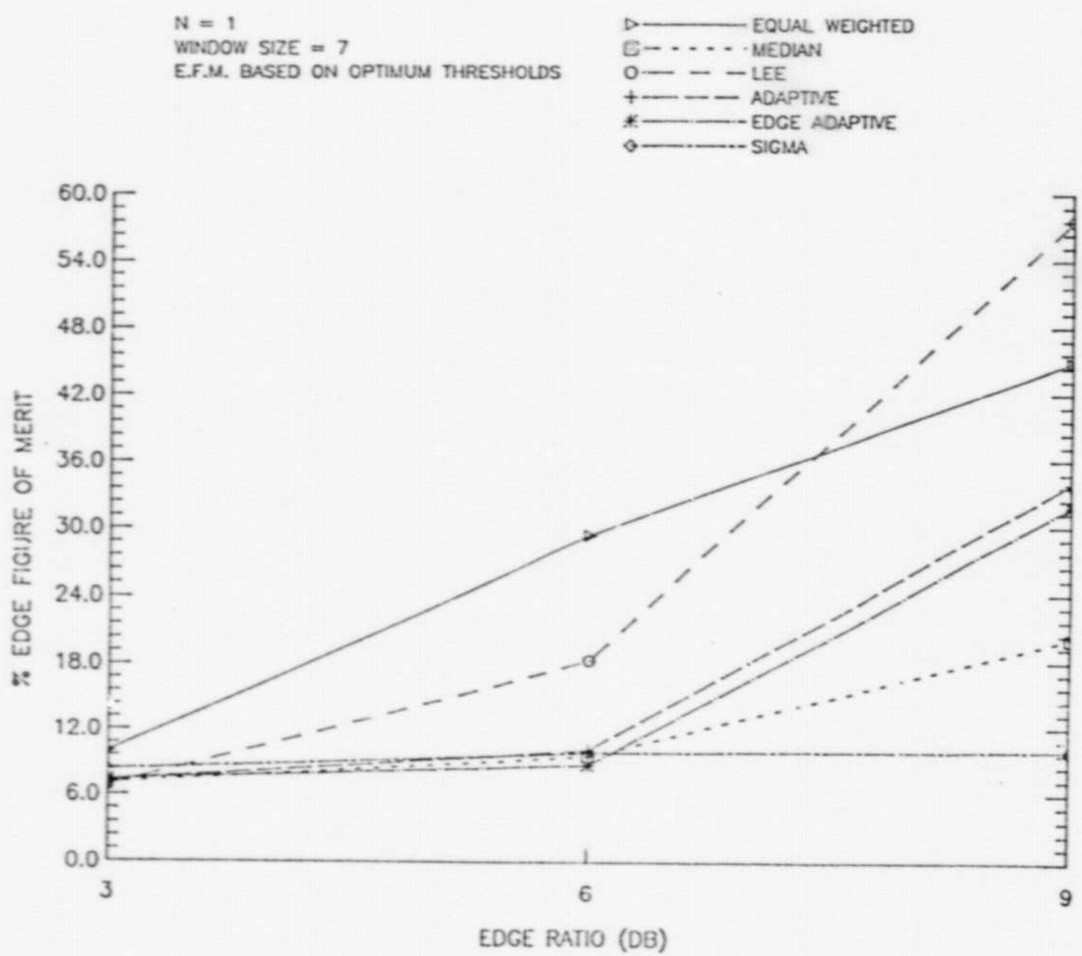

N = 1
WINDOW SIZE = 7
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
60.0
54.0
48.0
42.0
36.0
30.0
24.0
18.0
12.0
6.0
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.15

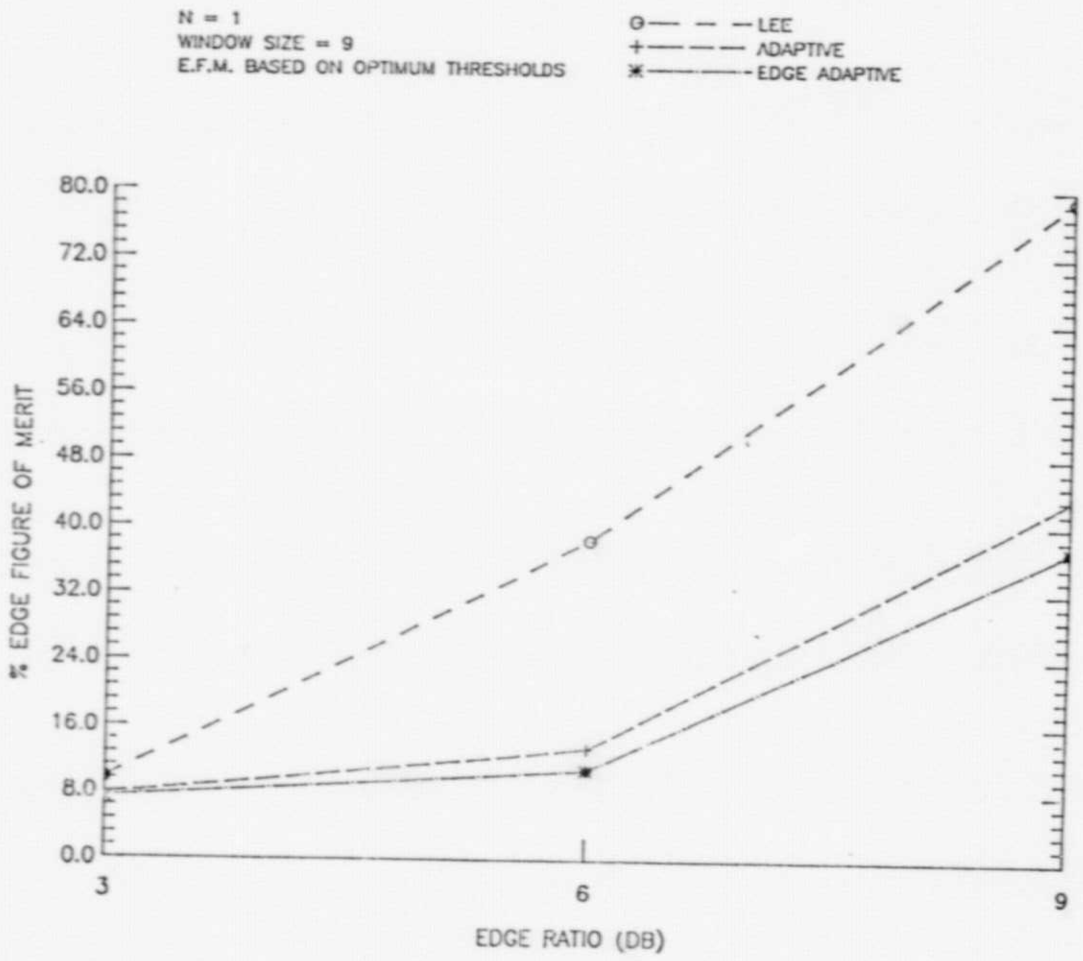

N = 1
WINDOW SIZE = 9
E.F.M. BASED ON OPTIMUM THRESHOLDS
LEE
ADAPTIVE
EDGE ADAPTIVE
% EDGE FIGURE OF MERIT
80.0
72.0
64.0
56.0
48.0
40.0
32.0
24.0
16.0
8.0
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.16

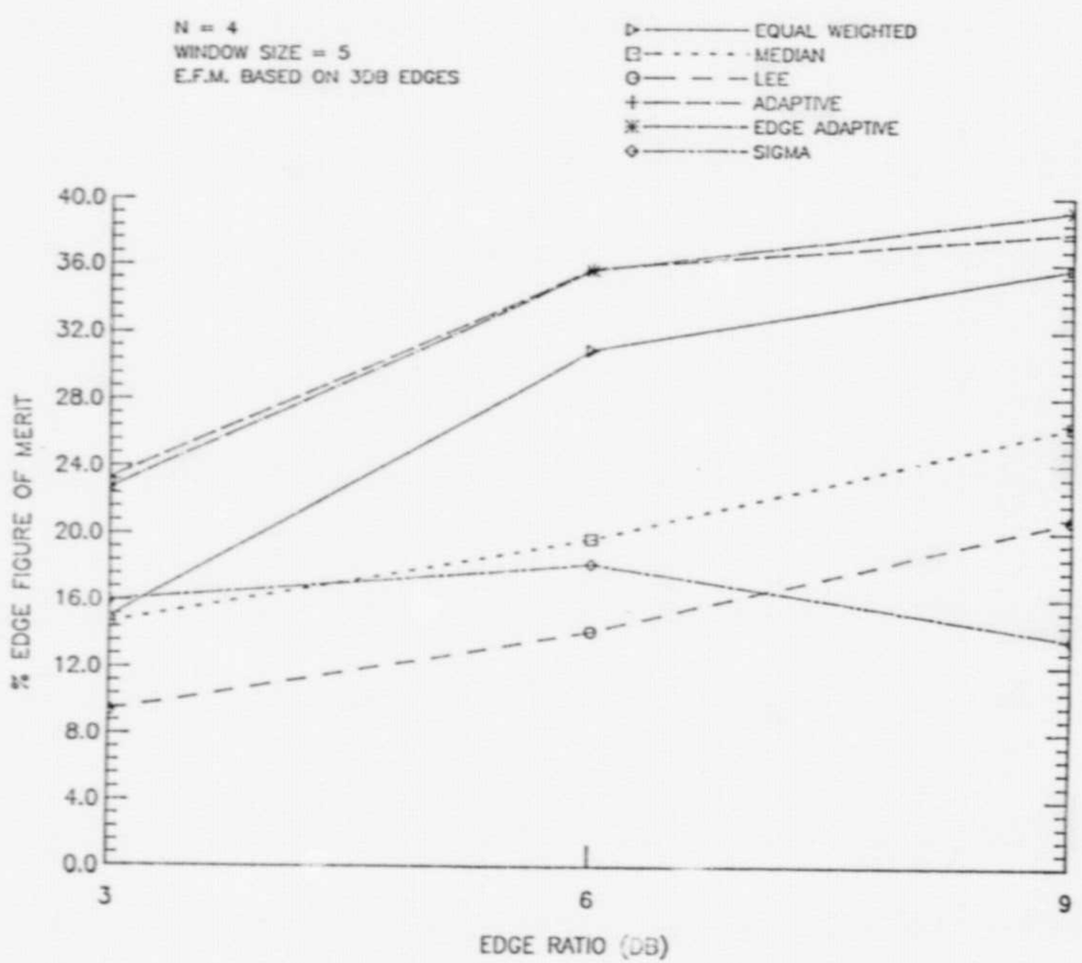

N = 4
WINDOW SIZE = 5
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
40.0
36.0
32.0
28.0
24.0
20.0
16.0
12.0
8.0
4.0
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.17

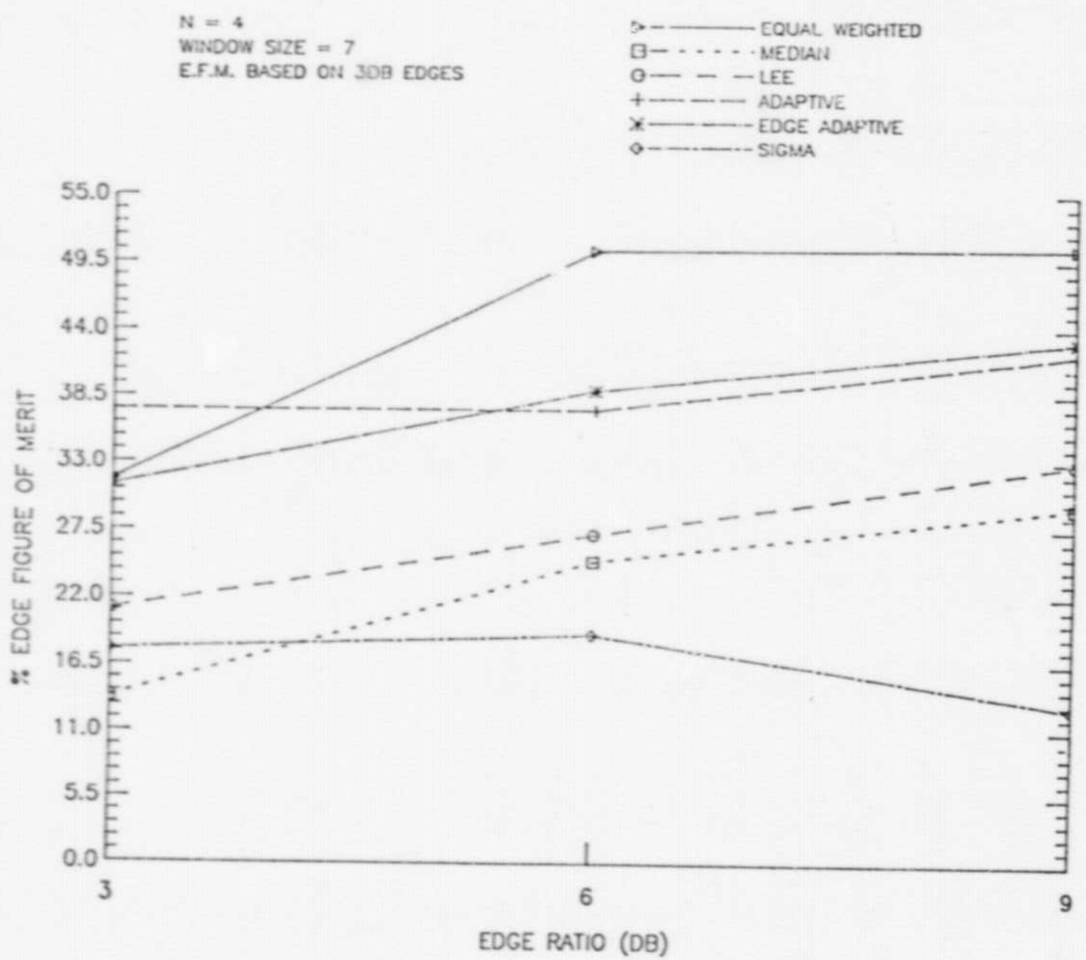

N = 4
WINDOW SIZE = 7
E.F.M. BASED ON 3DB EDGES
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
55.0
49.5
44.0
38.5
33.0
27.5
22.0
16.5
11.0
5.5
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.18

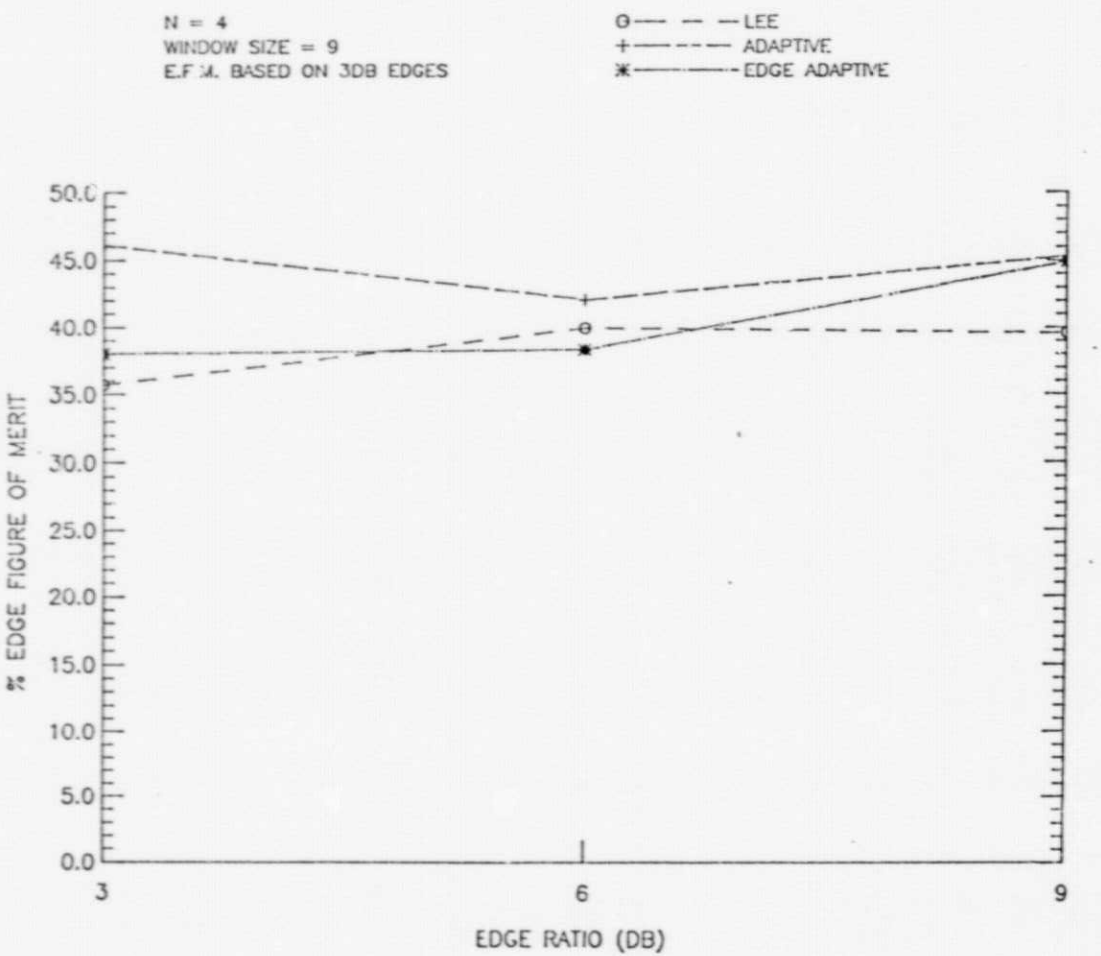

N = 4
WINDOW SIZE = 9
E.F.M. BASED ON 3DB EDGES
LEE
ADAPTIVE
EDGE ADAPTIVE
% EDGE FIGURE OF MERIT
50.0
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.19

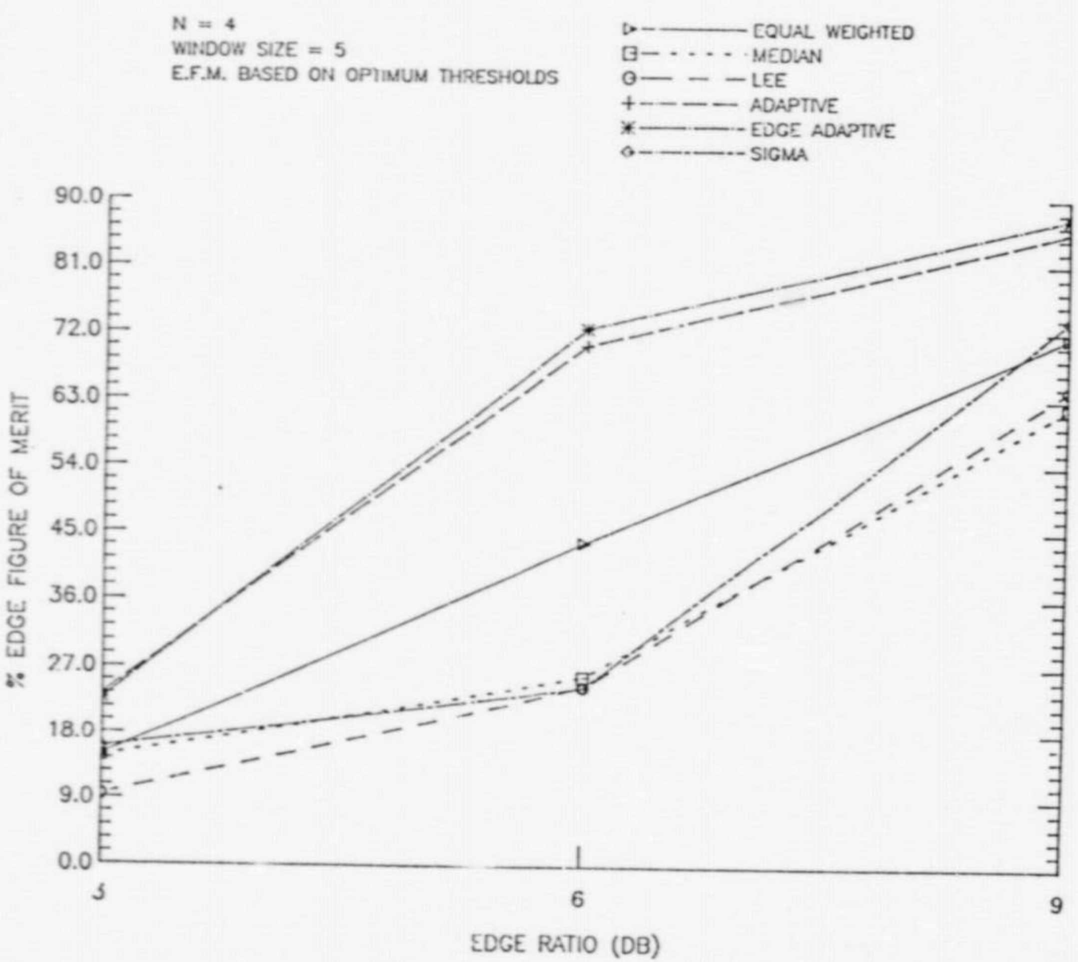

N = 4
WINDOW SIZE = 5
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
% EDGE FIGURE OF MERIT
90.0
81.0
72.0
63.0
54.0
45.0
36.0
27.0
18.0
9.0
0.0
3
6
9
EDGE RATIO (DB)

Figure 5.20

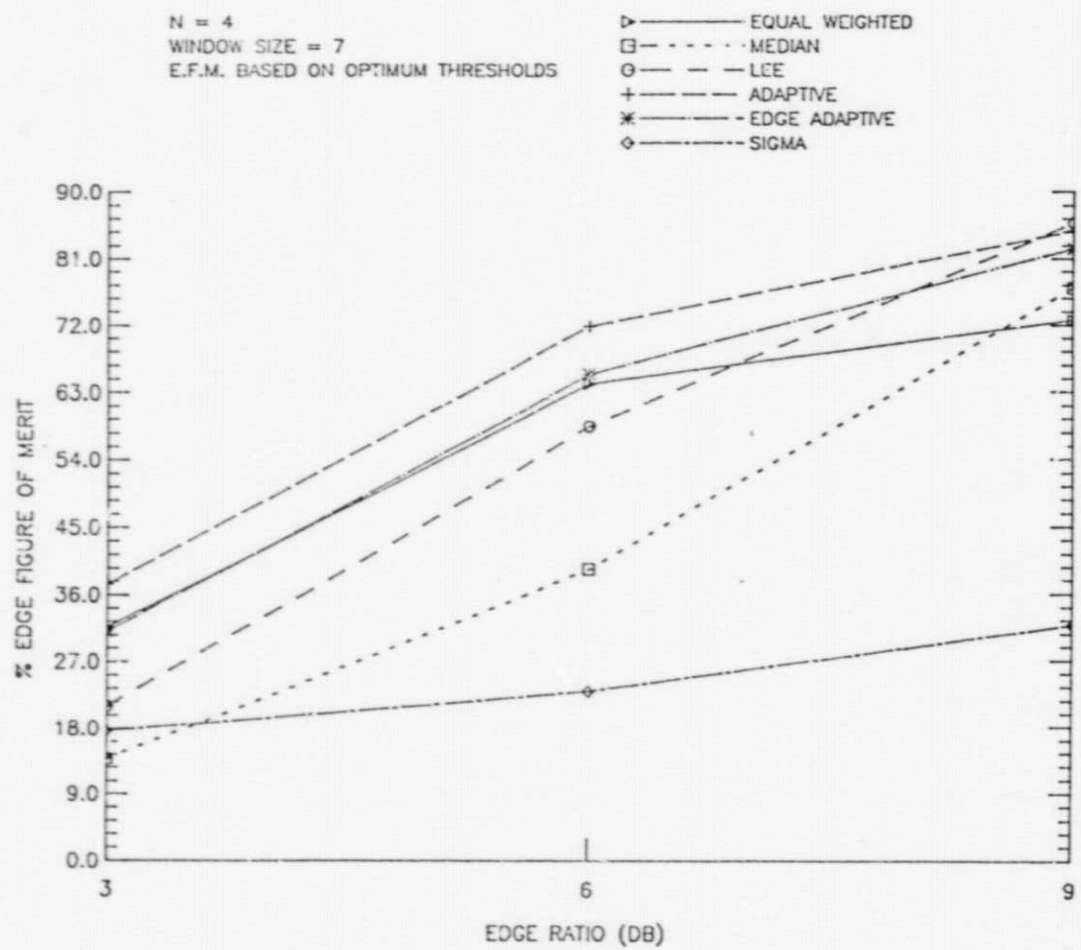

N = 4
WINDOW SIZE = 7
E.F.M. BASED ON OPTIMUM THRESHOLDS
EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
90.0
81.0
72.0
63.0
54.0
45.0
36.0
27.0
18.0
9.0
0.0
% EDGE FIGURE OF MERIT
3
6
9
EDGE RATIO (DB)

Figure 5.21

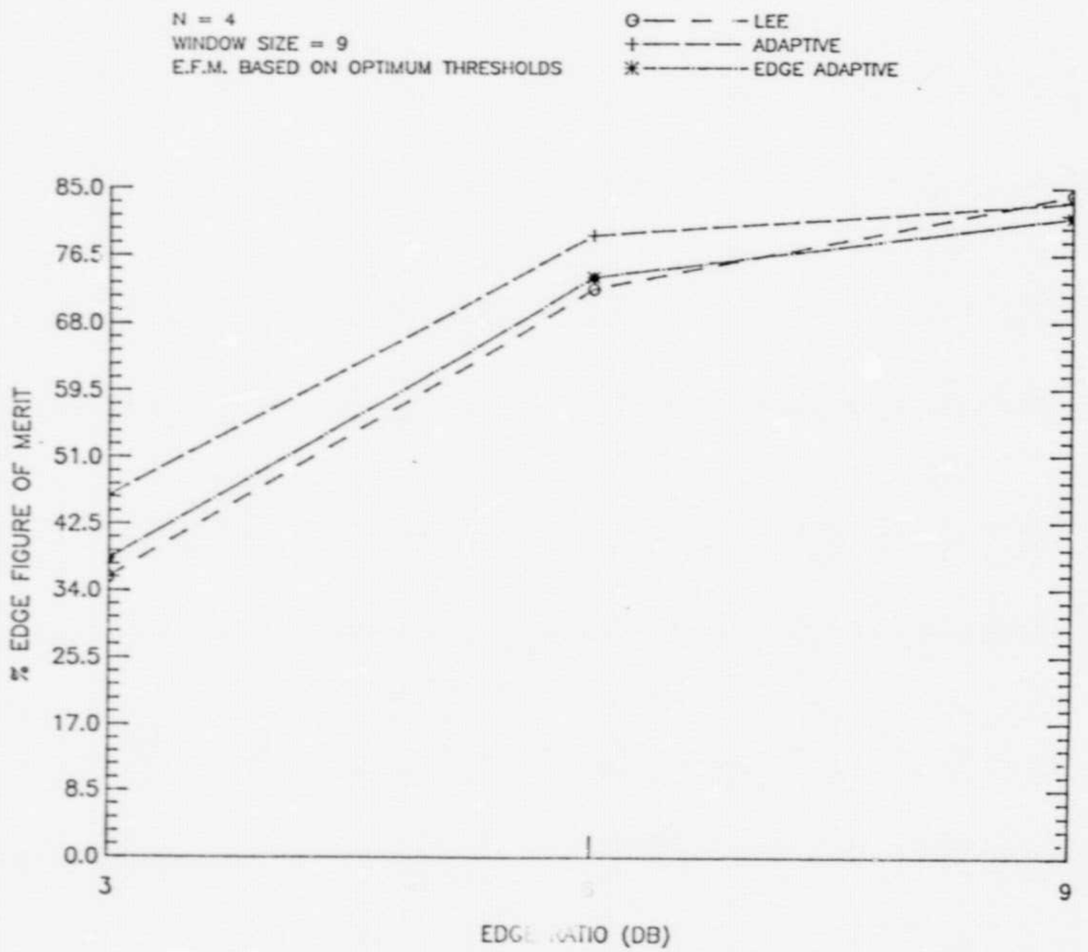
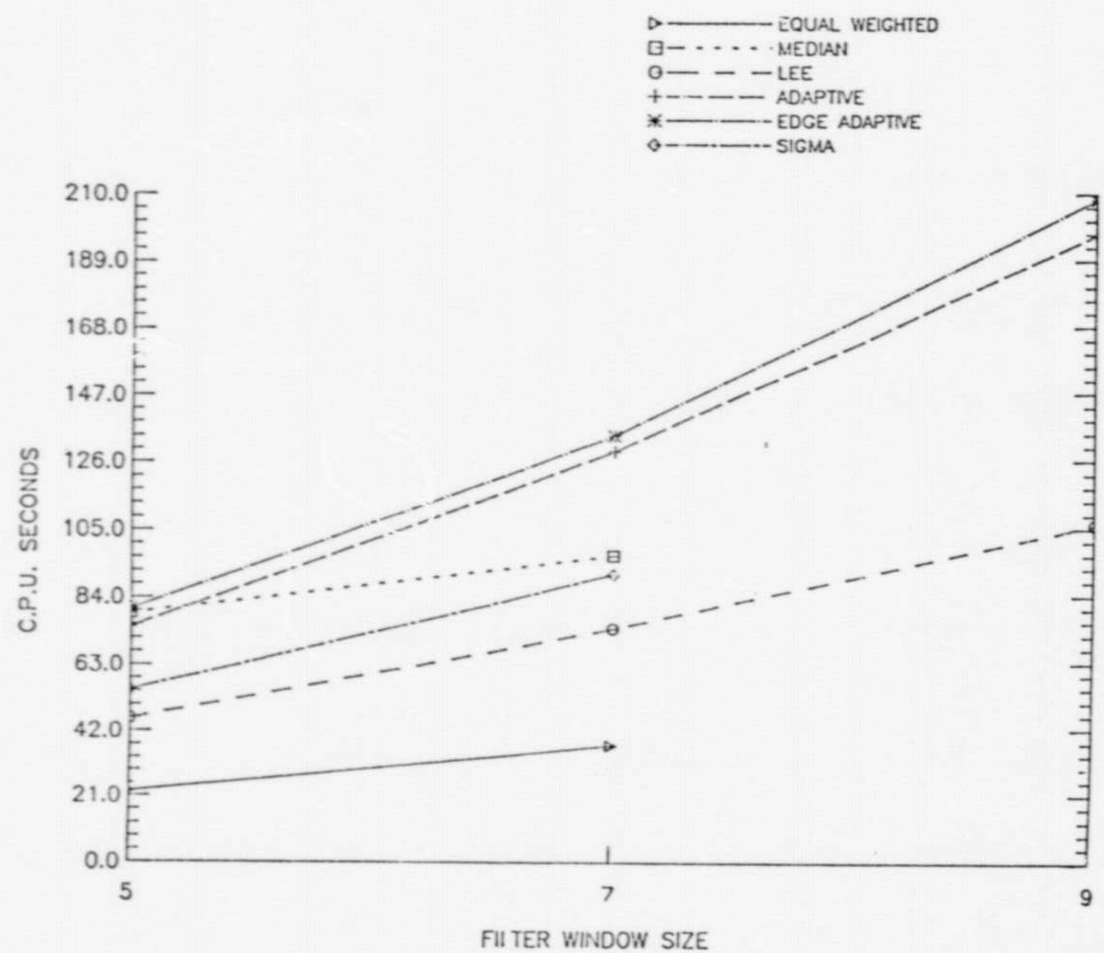

N = 4
WINDOW SIZE = 9
E.F.M. BASED ON OPTIMUM THRESHOLDS
LEE
ADAPTIVE
EDGE ADAPTIVE
% EDGE FIGURE OF MERIT
85.0
76.5
68.0
59.5
51.0
42.5
34.0
25.5
17.0
8.5
0.0
3
9

Figure 5.22



EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
C.P.U. SECONDS
210.0
189.0
168.0
147.0
126.0
105.0
84.0
63.0
42.0
21.0
0.0
5
7
9
FIITER WINDOW SIZE

Figure 5.23

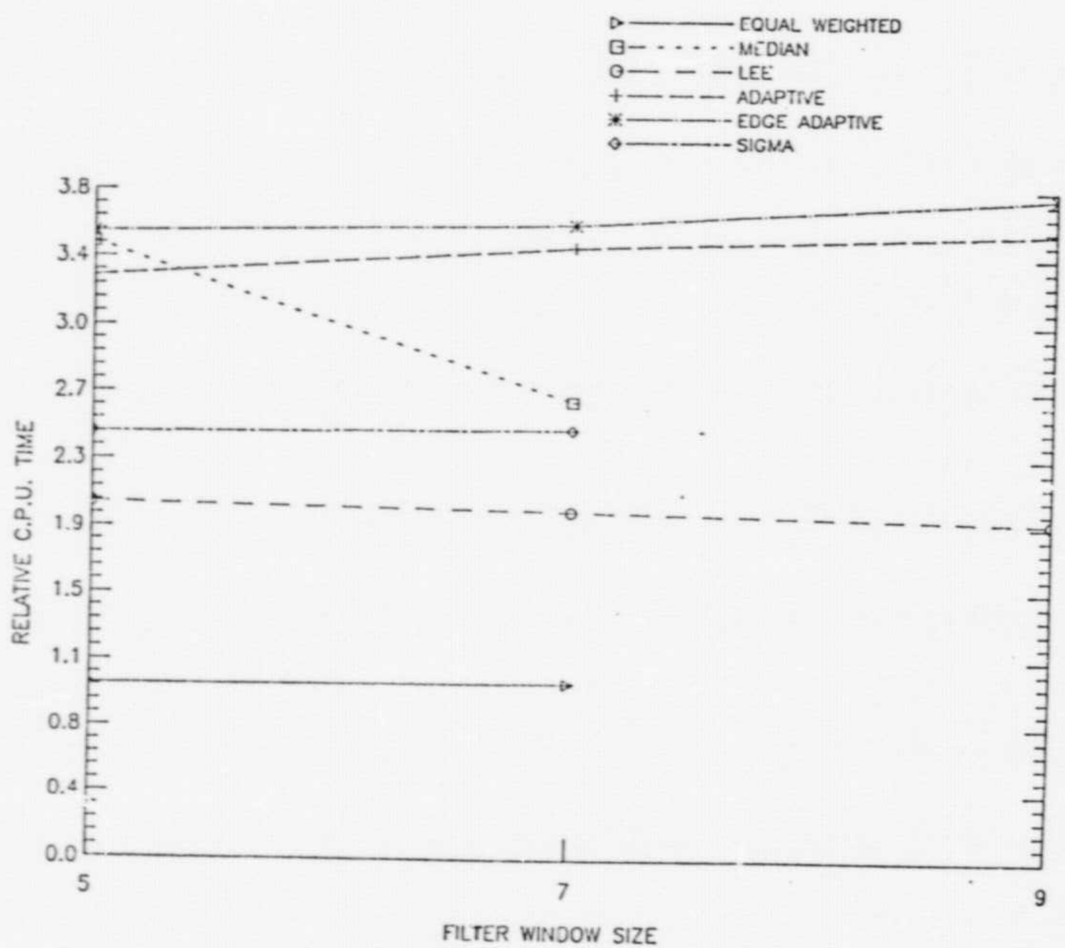

EQUAL WEIGHTED
MEDIAN
LEE
ADAPTIVE
EDGE ADAPTIVE
SIGMA
RELATIVE C.P.U. TIME
3.8
3.4
3.0
2.7
2.3
1.9
1.5
1.1
0.8
0.4
0.0
5
7
9
FILTER WINDOW SIZE

Figure 5.24

FILTER PERFORMANCE RANKINGS

Technique	Low Contrast	High Noise	Small Window	Computer Time
1. Equal Weighted	3	1	3	1
2. Median	6	5	4	4
3. Lee's	4	2	6	3
4. Adaptive	1	3	2	5
5. Edge Adaptive	2	6	1	6
6. Sigma	5	4	5	2

NOTE: The rankings for each column were determined by averaging the results for that parameter. The results for optimized thresholds were not included in the averages.

Fig. 5.25

REFERENCES

1. Victor S. Frost, "A Model for Radar Images and Its Application to Adaptive Digital Filtering of Multiplicative Noise," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-4, No. 2, pp 157-165, Mar. 1982.

2. J.S. Lee, "Refined Filtering of Image Noise Using Local Statistics," Computer Graphics and Image Processing 15, 380-389 (1981).

3. J.S. Lee, "Speckle Analysis and Smoothing of Synthetic Aperture Radar Images," Computer Graphics and Image Processing 17, 24-32 (1981).

4. Alan V. Oppenheim and Ronald W. Schafer, Digital Signal Processing. Chap. 10. New Jersey : Prentice-Hall, 1975.

5. W.K. Pratt, Digital Image Processing. New York : Wiley, 1978.

6. J.S. Lee, "A Simple Speckle Smoothing Algorithm for Synthetic Aperture Radar Images," IEEE Trans. Syst. Man Cybern. SMC-13, No. 1, 85-89 (1983).

7. R.C. Gonzalez and P. Wintz, Digital Image Processing, Massachusetts : Addison-Wesley, 1977.

APPENDIX A

Listings for computer programs

Program Name	Description
CONVO	Example of an equal-weighted filter
DIFOP	Discrete differentiation of an image
EFM	Calculate the edge figure of merit
EQUFLT	Equal-weighted filter
MEDFLT	Median filter
LEEFLT	Lee's edge filter
ADPFLT	Adaptive filter
EADFLT	Adaptive filter with a non-isotropic filter window
SIGFLT	Sigma filter

All required subroutines are listed after each mainline. However, modules that are common to several routines are given at the end of the appendix.

```
C       THIS IS A PROGRAM TO PERFORM A TWO-DIMENSIONAL SPATIAL
C       CONVOLUTION ON A REAL ARRAY.
C
C
        SUBROUTINE  CONVO (QUEUE, WINDOW, OUT, WNDSIZ, SIZE,
     &  DUMMY1, DUMMY2, OUTSIZ)
C
C       THE DUMMY ARGUMENTS ARE USED IN THIS IMPLEMENTATION IN
C       ORDER TO ALLOW VARIABLE ARRAY DIMENSIONS.
C
C
        IMPLICIT INTEGER (A-Z)
        REAL TOTAL, QUEUE (WNDSIZ, SIZE), WINDOW (WNDSIZ,WNDSIZ)
        REAL  OUT (OUTSIZ)
C
C.
C       INITIALIZE CIRCULAR QUEUE WHICH WILL STORE A HORIZONTAL
C       STRIP OF THE INPUT IMAGE.
C
        DO 10  X = 1, WNDSIZ
         READ (1)  (QUEUE (X, WRD) , WRD = 1, SIZE)
10      CONTINUE
C
C
C       INITIALIZE RECORD COUNT AND QUEUE POINTER.
C
        REC = 0
        QREC = 1
C
C       BEGINNING OF OUTERMOST LOOP.  SET THE TEMPORARY QUEUE
C       POINTER EQUAL TO THE FRONT OF THE QUEUE.
C
20      TMPQRC = QREC
C
C       PROCESS A ROW OF THE IMAGE
C
        DO 60  START = 1, OUTSIZ
C
C       PROCESS THE CONTENTS OF THE WINDOW.  INITIALIZE THE SUM
C       OF THE WINDOW PRODUCTS TO ZERO.
C
        TOTAL = 0.0
        DO 80  WREC = 1, WNDSIZ
         WWORD = 1
         DO 50  QWORD = START, START + WNDSIZ - 1
          TOTAL = TOTAL + QUEUE (TMPQRC, QWORD)
     &            * WINDOW (WREC, WWORD)
          WWORD = WWORD + 1
50       CONTINUE
C
C       UPDATE THE QUEUE POINTER
C
         TMPQRC = MOD ( TMPQRC, WNDSIZ ) + 1
C
80      CONTINUE
```

```
C
        OUT (START) = TOTAL
C
60      CONTINUE
C
C       WRITE THE OUTPUT RECORD AND READ IN A NEW ONE
C
        WRITE (2)  (OUT (WRD), WRD = 1, OUTSIZ)
C
        REC = REC + 1
C
        READ (1, END=200)  (QUEUE (QREC, WRD), WRD = 1, SIZE)
C
C       UPDATE THE POINTER TO THE FRONT OF THE QUEUE
C
        QREC = MOD ( QREC, WNDSIZ) + 1
C
C       LOOP BACK FOR ANOTHER RECORD UNTIL AN EOF IS REACHED
C
        GOTO 20
C
200     PRINT,'* * *  A L L D O N E * * *'
        PRINT,'THERE WERE', REC, 'RECORDS WRITTEN'
C
        STOP
        END
```

```
C ------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ------------------------------------------------------------------------
C PROGRAM NAME:DIFOP              AUTHOR:JEFF WATSON        DATE:4/15/83
C ------------------------------------------------------------------------
C PURPOSE :    PERFORMS DISCRETE DIFFERENTIATION OF AN INPUT
C IMAGE USING ROBERTS, PREWITT, OR SOBEL DIFFERENTIAL OPERATORS.
C THE OUTPUT CAN BE EITHER OF GRADIENTS OR OF EDGE ANGLES.
C
C
C
C ------------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ------------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ------------------------------------------------------------------------
C                      NON-LOCAL VARIABLES
C ------------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ------------------------------------------------------------------------
C                      SUBROUTINES REQUIRED
C   NAME           \          DESCRIPTION
C ------------------------------------------------------------------------
C DIFSUB           \PROCESSES THE IMAGE
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C ------------------------------------------------------------------------
C
      PROGRAM  DIFOP
C
      INTEGER  MAXSIZ, MXWSIZ, BUFQUE, BUFWND
C
C      BUFQUE = MAXSIZE * MAXWINDOWSIZE
C      BUFWND = MAXWINDOWSIZE * MAXWINDOWSIZE
C
      PARAMETER (MAXSIZ=512)
      PARAMETER (MXWSIZ=3)
```

```
      PARAMETER (BUFOUL=MAXSIZ*MXWSIZ)
      PARAMETER (BUFWND=MXWSIZ*MXWSIZ)
C
      INTEGER   RECS, TBUF(MAXSIZ), OPNUM, OUTTYPE
      INTEGER  SIZE, OUTSIZ, WNDSIZ, ARRSIZ
      INTEGER  OUBUF(BUFOUL), WINDOW(BUFWND)
C
C
C
      INTEGER*1   INFNM(18), OUTFNM(18)
C
      INTEGER  TTYIN, TTYOUT
      DATA  TTYIN, TTYOUT  /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER THE FILENAME FOR INPUT (MUST BE AN OLD FILE)')
      READ (TTYIN,510) INFNM
510   FORMAT (18A1)
      WRITE (TTYOUT,700)  INFNM
700   FORMAT (1X,18A1)
C
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER THE FILENAME FOR OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUTFNM
      WRITE (TTYOUT,700)  OUTFNM
C
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE INPUT IMAGE ')
      READ (TTYIN,*)  SIZE
      WRITE (TTYOUT,710)  SIZE
710   FORMAT (1X,I4)
C
      IF (SIZE .GT. MAXSIZ) THEN
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
      END IF
C
C
      WRITE (TTYOUT,627)
627   FORMAT (1X,'ENTER THE NUMBER OF THE DIFFERENTIAL OPERATOR',/,
     &       1X,'1 = ROBERTS, 2 = PREWITT, 3 = SOBEL')
      READ (TTYIN, *)  OPNUM
      WRITE (TTYOUT,710)  OPNUM
C
      IF ((OPNUM .GT. 3) .OR. (OPNUM .LT. 0))  THEN
       WRITE (TTYOUT,628)
628    FORMAT (1X,'* * *  E R R O R  - - DIFF OP NUMBERS ARE 1,2,3')
      GOTO 1010
      END IF
C
      IF (OPNUM .EQ. 1) THEN
       WNDSIZ = 2
      ELSE
       WNDSIZ = 3
      END IF
C
```

```
      WRITE(TTYOUT, 629)
  629 FORMAT(1X,'GRADIENT OR ANGLE?  1 = GRADIENT, 2  = ANGLE')
      READ(TTYIN, *) OUTTYPE
      WRITE (TTYOUT,710)  OUTTYPE
      IF((OUTTYPE .NE. 1) .AND. (OUTTYPE .NE. 2))   THEN
       WRITE(TTYOUT, 630)
  630  FORMAT(1X,'ERROR - - - OUTPUT IMAGE TYPE NUMBERS ARE 1 AND 2')
       GO TO 1010
      END IF
13    OUTSIZ = SIZE - WNDSIZ + 1
      ARRSIZ = WNDSIZ * WNDSIZ
      RECS = 0
C
C
C   CALL SUBROUTINE TO DO THE WORK
C
      CALL DIFSUB( INFNM, OUTFNM, ARRSIZ, SIZE, RECS,
     &      WNDSIZ, OUTSIZ, TTYOUT, QUEUE, WINDOW, IBUF, CPNUM, OUTTYPE)
C
C
C
      WRITE (TTYOUT,631)
631   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640)  OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
1010  STOP
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      OUT (START) = TOTAL
C
60    CONTINUE
C
C     WRITE THE OUTPUT RECORD AND READ IN A NEW ONE
C
      WRITE (2)  (OUT (WRD), WRD = 1, OUTSIZ)
C
      REC = REC + 1
C
      READ (1, END=200)  (QUEUE (QREC, WRD), WRD = 1, SIZE)
C
C     UPDATE THE POINTER TO THE FRONT OF THE QUEUE
C
      QREC = MOD ( QREC, WNDSIZ) + 1
C
C     LOOP BACK FOR ANOTHER RECORD UNTIL AN EOF IS REACHED
C
      GOTO 20
C
200   PRINT,'* * *  A L L D O N E * * *'
      PRINT,'THERE WERE', REC, 'RECORDS WRITTEN'
C
      STOP
      END
```

```
C ----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ----------------------------------------------------------------------
C PROGRAM NAME:DIFOP            AUTHOR:JEFF WATSON       DATE:4/15/83
C ----------------------------------------------------------------------
C PURPOSE :    PERFORMS DISCRETE DIFFERENTIATION OF AN INPUT
C IMAGE USING ROBERTS, PREWITT, OR SOBEL DIFFERENTIAL OPERATORS.
C THE OUTPUT CAN BE EITHER OF GRADIENTS OR OF EDGE ANGLES.
C
C
C
C ----------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ----------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ----------------------------------------------------------------------
C                      NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ----------------------------------------------------------------------
C                      SUBROUTINES REQUIRED
C   NAME           \        DESCRIPTION
C ----------------------------------------------------------------------
C DIFSUB           \PROCESSES THE IMAGE
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C ----------------------------------------------------------------------
C
      PROGRAM  DIFOP
C
      INTEGER  MAXSIZ, MXWSIZ, BUFCUE, BUFWND
C
C      BUFCUE = MAXSIZE * MAXWINDOWSIZE
C      BUFWND = MAXWINDOWSIZE * MAXWINDOWSIZE
C
      PARAMETER (MAXSIZ=512)
      PARAMETER (MXWSIZ=3)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      PARAMETER (BUFOUT=MAXSIZ*MXWSIZ)
      PARAMETER (BUFWND=MXWSIZ*MXWSIZ)
C
      INTEGER   RECS, TBUF(MAXSIZ), OPNUM, OPTYPE
      INTEGER  SIZE, OUTSIZ, WNDSIZ, ARRSIZ
      INTEGER  OUTBUF(BUFOUT), WINDOW(BUFWND)
C
C
C
      INTEGER*1   INFNM(18), OUTFNM(18)
C
      INTEGER  TTYIN, TTYOUT
      DATA  TTYIN, TTYOUT  /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER THE FILENAME FOR INPUT (MUST BE AN OLD FILE)')
      READ (TTYIN,510) INFNM
510   FORMAT (18A1)
      WRITE (TTYOUT,700)  INFNM
700   FORMAT (1X,18A1)
C
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER THE FILENAME FOR OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUTFNM
      WRITE (TTYOUT,700)  OUTFNM
C
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE INPUT IMAGE ')
      READ (TTYIN,*)  SIZE
      WRITE (TTYOUT,710)  SIZE
710   FORMAT (1X,I4)
C
      IF (SIZE .GT. MAXSIZ) THEN
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
      END IF
C
C
      WRITE (TTYOUT,627)
627   FORMAT (1X,'ENTER THE NUMBER OF THE DIFFERENTIAL OPERATOR',/,
     &        1X,'1 = ROBERTS, 2 = PREWITT, 3 = SOBEL')
      READ (TTYIN, *)  OPNUM
      WRITE (TTYOUT,710)  OPNUM
C
      IF ((OPNUM .GT. 3) .OR. (OPNUM .LT. 0))  THEN
       WRITE (TTYOUT,628)
628    FORMAT (1X,'* * *  E R R O R  - - DIFF OP NUMBERS ARE 1,2,3')
      GOTO 1010
      END IF
C
      IF (OPNUM .EQ. 1) THEN
       WNDSIZ = 2
      ELSE
       WNDSIZ = 3
      END IF
C
```

```
      WRITE(TTYOUT, 629)
  629 FORMAT(1X,'GRADIENT OR ANGLE?  1 = GRADIENT, 2  = ANGLE')
      READ(TTYIN, *) OUTTYPE
      WRITE (TTYOUT,710)  OUTTYPE
      IF((OUTTYPE .NE. 1) .AND. (OUTTYPE .NE. 2))   THEN
       WRITE(TTYOUT, 630)
  630  FORMAT(1X,'ERROR - - - OUTPUT IMAGE TYPE NUMBERS ARE 1 AND 2')
       GO TO 1010
      END IF
13    OUTSIZ = SIZE - WNDSIZ + 1
      ARRSIZ = WNDSIZ * WNDSIZ
      RECS = 0
C
C
C   CALL SUBROUTINE TO DO THE WORK
C
      CALL DIFSUB( INFNM, OUTFNM, ARRSIZ, SIZE, RECS,
     &      WNDSIZ, OUTSIZ, TTYOUT, QUEUE, WINDOW, IBUF, CPNUM, OUTTYPE)
C
C
C
      WRITE (TTYOUT,631)
631   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640)  OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
1010  STOP
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C ----------------------------------------------------------------------
C UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION S   NCES LAB
C ----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ----------------------------------------------------------------------
C PROGRAM NAME:DIFSUB            AUTHOR:JEFF WATSON       DATE:4/ 5/83
C ----------------------------------------------------------------------
C PURPOSE :     DOES THE PROCESSING FOR THE DIFFERENTIAL OPERATOR
C FILTER AFTER BEING CALLED BY DIFOP.
C
C
C
C
C ----------------------------------------------------------------------
C                          PARAMETER DEFINITION
C  NAME               \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ----------------------------------------------------------------------
C INFILE              \CH*18   \R       \          \INPUT FILENAME
C OUTFILE             \CH*18   \R       \          \OUTPUT FILENAME
C ARRSIZE             \I       \R       \          \NUMBERS OF ELEMENTS IN WINDOW
C NCOLS               \I       \R       \          \COLUMNS IN INPUT IMAGE
C NOUTROW             \I       \W       \          \ROWS IN OUTPUT IMAGE
C WINDSIZE            \I       \R       \          \WINDOWSIZE
C NOUTCOL             \I       \R       \          \COLUMNS IN OUTPUT IMAGE
C TTYOUT              \I       \R       \          \OUTPUT TO TERMINAL FILECODE
C Q                   \I       \R       \          \CIRCULAR QUEUE
C ARRVAL              \I       \R       \          \WINDOW ARRAY
C TBUF                \I       \R       \          \READ IN BUFFER
C OPNUM               \I       \R       \          \OPERATOR NUMBER
C OUTTYPE             \I       \R       \          \ANGLE OR GRADIENT OUTPUT
C                     \        \        \          \
C ----------------------------------------------------------------------
C                          NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C ----------------------------------------------------------------------
C                         SUBROUTINES REQUIRED
C    NAME              \          DESCRIPTION
C ----------------------------------------------------------------------
C GENMSK              \GENERATES DIFF. OP MASKS
C GETPNT              \FINDS POINT VALUE FOR OUTPUT
C OPN                 \OPENS FILE AND ASSIGNS LOGICAL UNIT
C UCLOSE              \CLOSES FILES OPENED WITH OPN
C                      \
C                     \
C                     \
C ----------------------------------------------------------------------
C
C
      SUBROUTINE DIFSUB ( INFILE, OUTFILE, ARRSIZE, NCOLS, NOUTROW,
     $   WINDSIZE, NOUTCOL, TTYOUT, Q, ARRVAL, TBUF, OPNUM, OUTTYPE)
C
      LOGICAL  ERR
C
      INTEGER  ARRSIZE, NCOLS, NOUTCOL, WINDSIZE, OPNUM, OUTTYPE
      INTEGER I, J, K, ERRNUM, START, ARRPOS, OUTCOL, NROWS
      INTEGER  NOUTROW,TTYOUT, ARRVAL(ARRSIZE), TBUF(NCOLS)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      INTEGER   [illegible]DSIZE,NCOLS), INFNUM, OUTNUM, [illegible], VALUE
      INTEGER   [illegible]SK1(9), MSK2(9), G1, G2
      INTEGER*1 INFILE(18), OUTFILE(18)
      REAL   ANGLE, PI
C
      NOUTROW = 0
      PI = 4 * ATAN(1.0)
C
      CALL  GENMSK( MSK1, MSK2, ARRSIZE, OPNUM)
C
      CALL OPN (INFNUM, INFILE, 'OLD', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO 92
      CALL OPN (OUTNUM, OUTFILE, 'NEW', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO 94
C
C          INITIALIZE QUEUE
C
      DO 30  I = 1,WINDSIZE
       READ(INFNUM,END = 98) (TBUF(K), K = 1, NCOLS)
       DO 20  J = 1,NCOLS
        Q(I,J) = TBUF(J)
   20 CONTINUE
   30 CONTINUE
C
C
C                MAIN PROCESSING
C
   40 CONTINUE
       DO 70  START = 1, NOUTCOL
         DO 60  I = 1, WINDSIZE
          DO 50  J = START, START + WINDSIZE - 1
            ARRPOS = WINDSIZE*(I-1) + J-START+1
            ARRVAL(ARRPOS) = Q(I,J)
   50     CONTINUE
   60    CONTINUE
      CALL  GETPNT( MSK1, ARRSIZE, ARRVAL, G1)
      CALL  GETPNT( MSK2, ARRSIZE, ARRVAL, G2)
C
      IF(OUTTYPE .EQ. 1)  THEN
       VALUE = SQRT(REAL(G1**2) + REAL(G2**2))
      ELSE
       IF( G1 .EQ. 0 )     THEN
         IF( G2 .GT. 0 )   ANGLE  = PI/2.0
         IF( G2 .LE. 0 )   ANGLE  = 3 * PI / 2.0
       ELSE
           ANGLE  =  ATAN( REAL(G2)/ REAL(G1) )
       END IF
C
C    ANGLE IS NOW BETWEEN -PI/2 AND PI/2, AND WE WANT IT BETWEEN
C    0 AND 2PI.  MUST CHECK SIGN OF G1 AND G2.
C
       IF( G1 .LT. 0)  ANGLE = ANGLE + PI
       IF(( G2 .LT. 0 ) .AND. ( G1 .GT. 0 )) ANGLE = ANGLE + 2*PI
       IF( OPNUM .EQ. 1 ) THEN
         ANGLE = PI/4.0 + ANGLE
       END IF
       VALUE = 255 * ANGLE / (2.0 * PI)  +   0.5
C
C     FOR ROBERTS, THE ANGLES WERE ROTATED BY PI/4 .  THIS COULD
C     CAUSE PROBLEMS WHEN THE ANGLE GETS ROTATED INTO THE FIRST QUADRANT
```

```
C
      IF( VALUE .GT. 255 )   VALUE = VALUE - 255
      END IF
      OUTCOL = START
      TBUF(OUTCOL) = VALUE
   70  CONTINUE
C
      NOUTROW = NOUTROW + 1
      WRITE(OUTNUM) (TBUF(K), K = 1, NOUTCOL)
C
C         UPDATE QUEUE
C
       DO 80  I = 1 , WINDSIZE
        IF(I .LT. WINDSIZE) THEN
         DO 85  J = 1, NCOLS
           Q(I,J) = Q(I+1,J)
   85    CONTINUE
         ELSE
          READ(INFNUM,END = 98) (TBUF(K), K = 1, NCOLS)
           DO 87  J = 1,NCOLS
             Q(I,J) = TBUF(J)
   87      CONTINUE
         END IF
   80 CONTINUE
      GO TO 40
C
C        WE HAVE REACHED THE END OF THE INPUT FILE
C
   92 CALL FILERR (TTYOUT, INFILE, ERRNUM)
C
      GOTO 99
   94 CALL FILERR (TTYOUT, OUTFILE, ERRNUM)
C
      GOTO 99
C
C
   98 CONTINUE
      CALL UCLOSE(INFNUM)
      CALL UCLOSE(OUTNUM)
   99 CONTINUE
      RETURN
      END
EOF..
?
```

```
C -----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFO                 S LAB
C -----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C -----------------------------------------------------------------------
C PROGRAM NAME:GENMSK          AUTHOR:JEFF WATSON       DATE          3
C -----------------------------------------------------------------------
C PURPOSE :    TO GENERATE THE ROBERTS, PREWITT, OR SOBEL
C  DIFFERENTIAL OPERATOR MASK.
C
C
C
C
C -----------------------------------------------------------------------
C                            PARAMETER DEFINITION
C  NAME               \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C -----------------------------------------------------------------------
C MSK1                \I      \W      \        \DIFFERINTIAL OPERATOR MASK
C MSK2                \I      \W      \        \DIFFERENTIAL OPERATOR MASK
C ARRSIZE             \I      \R      \        \NUMBER OF WINDOW ELEMENTS
C OPNUM               \I      \R      \        \OPERATOR NUMBER
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C -----------------------------------------------------------------------
C                            NON-LOCAL VARIABLES
C -----------------------------------------------------------------------
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C                     \       \       \        \
C -----------------------------------------------------------------------
C                           SUBROUTINES REQUIRED
C    NAME               \           DESCRIPTION
C -----------------------------------------------------------------------
C                       \
C                       \
C                       \
C                       \
C                       \
C                       \
C                       \
C -----------------------------------------------------------------------
      SUBROUTINE  GENMSK( MSK1, MSK2, ARRSIZE, OPNUM)
C
      INTEGER  ARRSIZE, OPNUM, C
      INTEGER  MSK1(ARRSIZE), MSK2(ARRSIZE)
C
      IF (OPNUM .NE. 1) GO TO 10
       MSK1(1) =  0
       MSK1(2) = -1
       MSK1(3) =  1
       MSK1(4) =  0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
        MSK2(1) = -1
        MSK2(2) =  0
        MSK2(3) =  0
        MSK2(4) =  1
       GO TO 20
C
C     PREWITT AND SOBEL DIFF. OPERATORS DIFFER ONLY BY THE CONSTANT
C     C.  C=2 FOR SOBEL, C=1 FOR PREWITT.
C
   10 CONTINUE
      C = 1
      IF (OPNUM .EQ. 3)  C = 2
C
        MSK1(1) =  1
        MSK1(2) =  0
        MSK1(3) = -1
        MSK1(4) =  C
        MSK1(5) =  0
        MSK1(6) = -C
        MSK1(7) =  1
        MSK1(8) =  0
        MSK1(9) = -1
C
        MSK2(1) = -1
        MSK2(2) = -C
        MSK2(3) = -1
        MSK2(4) =  0
        MSK2(5) =  0
        MSK2(6) =  0
        MSK2(7) =  1
        MSK2(8) =  C
        MSK2(9) =  1
C
   20 CONTINUE
      RETURN
      END
EOF..
```

```
C ------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ------------------------------------------------------------------------
C PROGRAM NAME:THDFND          AUTHOR:J. SCOTT GARDNER DATE:02/27/83
C ------------------------------------------------------------------------
C PURPOSE :    THIS IS THE MAINLINE FOR THE THRESHOLD
C SEARCH ROUTINE WHICH USES A QUADRATIC SEARCH
C TO FIND THE BEST THRESHOLD FOR PRODUCING
C AN EDGE MAP BASED ON THE EDGE FIGURE OF MERIT.
C
C
C ------------------------------------------------------------------------
C                     PARAMETER DEFINITION
C  NAME              \ TYPE   \ CLASS\ RANGE   \     DESCRIPTION
C ------------------------------------------------------------------------
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C ------------------------------------------------------------------------
C                      NON-LOCAL VARIABLES
C ------------------------------------------------------------------------
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C                    \        \      \         \
C ------------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME              \          DESCRIPTION
C ------------------------------------------------------------------------
C THDCUD             \DOES THE PROCESSING-- CALLED WITH VARIABLE PARAMETERS
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C ------------------------------------------------------------------------
      PROGRAM  THDFND
C
      INTEGER  MAXSIZ, MXWSIZ
C
      PARAMETER (MAXSIZ=512)
      PARAMETER (MXWSIZ=15)
C
      INTEGER  RECS
      INTEGER  TBUFF(MAXSIZ), SIZE, OUTSIZ, WNDSIZ
C
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      INTEGER*1  IDLFN(18), ACTLFN(18), THRSFN(18)
C
      INTEGER  TTYIN,TTYOUT
C
      LOGICAL  ERR
C
      ERR = .FALSE.
      DATA  TTYIN, TTYOUT  /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER IDEAL IMAGE FILENAME (MUST BE AN CLC FILE)')
      READ (TTYIN,510) IDLFN
510   FORMAT (18A1)
C
      WRITE(TTYOUT,701) IDLFN
  701 FORMAT(1X,18A1)
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER ACTUAL IMAGE FILENAME (MUST BE AN CLC FILE)')
      READ (TTYIN,510) ACTLFN
C
      WRITE(TTYOUT,701) ACTLFN
C
      WRITE (TTYOUT,603)
  603 FORMAT(1X,'FILENAME FOR THE EDGE MAP (MUST BE A NEW FILE)')
      READ (TTYIN,510) THRSFN
C
      WRITE(TTYOUT,701) THRSFN
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE IDEAL IMAGE ')
      READ (TTYIN,*)  SIZE
      WRITE(TTYOUT,*)  SIZE
C
      IF (SIZE .LE. MAXSIZ) GOTO 12
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
C
12    WRITE (TTYOUT,627)
627   FORMAT (1X,'ENTER THE SIZE OF THE FILTERED IMAGE')
      READ (TTYIN, *)  OUTSIZ
      WRITE (TTYOUT, *)  OUTSIZ
C
      IF (OUTSIZ .GT. SIZE)    THEN
       WRITE (TTYOUT,641)
641    FORMAT (1X,'FILTERED IMAGE MUST BE SMALLER THAN IDEAL IMAGE')
       GOTO 1010
      END IF
C
C
C   CALL SUBROUTINE TO DO THE WORK
C
C
       CALL THDCUD (IDLFN, ACTLFN, THRSFN, SIZE, RECS, OUTSIZ,
     &  TTYOUT, TTYIN, TBUFF, ERR)
C
C
```

```
      IF( ERR ) GO TO 1010
C
      WRITE (TTYOUT,630)
630   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640)  OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
C
1010  STOP
      END
EOF..
?
```

```
C --------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C --------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS         REF. # :
C --------------------------------------------------------------------------
C PROGRAM NAME:THDQUD            AUTHOR:JEFF WATSON        DATE:5/19/83
C --------------------------------------------------------------------------
C PURPOSE :   THDQUD IS A QUADRATIC SEARCH.  USING A QUADRATIC APPROX-
C IMATION, IT DETERMINES THE NEXT THRESHOLD TO HAVE ITS EDGE FIGURE
C OF MERIT CALCULATED.  I ASSUMED THAT A GRAPH OF EFM'S VS. THRESHOLD
C WOULD BE APPROXIMATELY A QUADRATIC.  THE SEARCH TERMINATES WHEN
C THE EFM THAT IS APPROXIMATED BY THE QUADRATIC IS WITHIN A SMALL
C AMOUNT (DELTA) OF THE ACTUAL EFM AND THE SEARCH IS OVER A SMALL
C RANGE OF THRESHOLDS, OR WHEN A SET MAXIMUM ITERATIONS OF THE SEARCH
C IS REACHED.
C --------------------------------------------------------------------------
C                        PARAMETER DEFINITION
C    NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C --------------------------------------------------------------------------
C ICFLE              \INT    \READ  \        \IDEAL EDGE MAP
C ACTFLE             \INT    \READ  \        \NOISY EDGE MAP
C THRFLE             \INT    \WRITE \        \BEST THRESHOLD EDGE MAP
C NCOL               \INT    \READ  \  0-512 \NUM COLUMNS IN ICFLE
C NOUTROW            \INT    \WRITE \  0-512 \NUM RECS IN ACTFLE AND THDFLE
C NOUTCOL            \INT    \READ  \  0-512 \NUM COLS IN ACTFLE AND THDFLE
C TTYOUT             \INT    \READ  \  16    \OUTPUT TO TERMINAL FILECODE
C TTYIN              \INT    \READ  \  15    \INPUT FROM TERMINAL FILECODE
C TBUF(NOUTCOL)      \INT    \WRITE \        \TEMPORARY BUFFER ARRAY
C ERR                \LOG    \WRITE \        \FLAG FOR FILE OPENING ERROR
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C --------------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C --------------------------------------------------------------------------
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C --------------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C     NAME           \          DESCRIPTION
C --------------------------------------------------------------------------
C EFM                \CALCULATES EDGE FIGURE OF MERIT FOR A GIVEN THRESHOLD
C THCRFI             \CREATES THRESHOLDED FILE
C THNEWX             \FINDS THE NEXT THRESHOLD
C THA2XO             \FINDS NEW INITIAL THRESHOLD AND SEARCH STEPLENGTH
C THSPEC             \FOR A DIFFERENT IMAGE FINDS EFM AND CREATES AND EDGE
C                    \MAP FOR FIRST IMAGE'S BEST THRESHOLD
C OPN                \OPENS FILES
C UCLOSE             \CLOSES FILES OPENED WITH OPN
C FILERR             \GIVES ERROR MESSAGE FOR FILE OPENING ERROR
C --------------------------------------------------------------------------
C
      SUBROUTINE THDQUD(IDLFLE, ACTFLE, THRFLE, NCOL, NOUTROW,
     &                 NOUTCOL, TTYOUT, TTYIN, TBUF, ERR)
C
       INTEGER  NCOL, NOUTROW, NOUTCOL, TTYOUT, TTYIN, MAXCOUNT
       INTEGER  ERRNUM, TBUF(NOUTCOL), J, IDLNUM, ACTNUM, THRNUM
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      INTEGER   X, XLEFT, XRIGHT, ALFSTAR, T, XMARK, ALPHA
      INTEGER   ALFST, CCUNT, XO, A3, A2, ASTAR, MINTH, MAXTH, ANSWER
C
      INTEGER*1 IDLFLE(18), ACTFLE(18), THRFLE(18)
C
      LOGICAL   RIGHT, ERR, DEBUG, INBOUND
C
      REAL      ERRCAL, DELTA, EFMARR(0:255), HALFSTAR, F1, F2, F3
      REAL      EFMMAX, HALFST, EFM, HSTAR, FSTAR
C
      PARAMETER (MAXCOUNT = 10)
      PARAMETER (DELTA = .01)
      PARAMETER (MINTH = 0)
      PARAMETER (MAXTH = 255)
C
      XLEFT(XMARK,ALPHA)  = XMARK - ALPHA
      XRIGHT(XMARK,ALPHA) = XMARK + ALPHA
      ALFST(F1,F2,F3,T)   = INT((4*F2 - 3*F1 - F3)*T/
     &                      (4*F2 - 2*F3 - 2*F1))
      HALFST(F1,F2,F3)    = F1 - (ABS(4*F2 - 3*F1 - F3))**2/
     &                      (8*(F1 - 2*F2 + F3))
C
      CALL OPN( THRNUM, THRFLE, 'NEW', 'UNF', ERRNUM, ERR)
           IF( ERR )   THEN
                CALL FILERR( TTYOUT, THRFLE, ERRNUM )
                GO TO 99
           END IF
      CALL OPN( ACTNUM, ACTFLE, 'OLD', 'UNF', ERRNUM, ERR)
           IF( ERR )   THEN
                CALL FILERR( TTYOUT, ACTFLE, ERRNUM )
                GO TO 99
           END IF
      CALL OPN( IDLNUM, IDLFLE, 'OLD', 'UNF', ERRNUM  ERR)
           IF( ERR )   THEN
                CALL FILERR( TTYOUT, IDLFLE, ERRNUM )
                GO TO 99
           END IF
C.
      DO 10  J = MINTH, MAXTH
         EFMARR(J) = 0.0
   10 CONTINUE
C
      XO = 0
      A2 = 50
      X  = XRIGHT( XO, A2 )
C
      F1 = EFM( IDLNUM, ACTNUM, NCOL, NOUTCOL, XO, TTYOUT)
      F2 = EFM( IDLNUM, ACTNUM, NCOL, NOUTCOL, X, TTYOUT)
      EFMARR(XO)  = F1
      EFMARR(X) = F2
C
      COUNT = 2
      IF( F2 .GE. F1 )     THEN
        RIGHT = .TRUE.
      ELSE
        RIGHT = .FALSE.
        XO = XRIGHT( XO, A2 )
        F3 = F2
        F2 = F1
        F1 = F3
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C           F3 USED ONLY TO MOVE F1 AND F2 AROUND HERE
      END IF
      DEBUG = .FALSE.
C
C        SEARCH LOOP
C
   20 CONTINUE
   30 CONTINUE
      A3 = 2 * A2
C
C
      CALL THNEWX( RIGHT, XO, A3, MAXTH, MINTH, X, INBOUND)
C
C
      IF( INBOUND )   THEN
          IF(EFMARR(X) .EQ. 0.0 )   THEN
               F3 = EFM( IDLNUM, ACTNUM, NCOL, NOUTCOL, X, TTYOUT)
               EFMARR(X) = F3
          ELSE
               F3 = EFMARR( X )
          END IF
          COUNT = COUNT + 1
      ELSE
          F3 = 0
      END IF
C
      IF( DEBUG )  THEN
        WRITE(TTYOUT,45) F1,F2,F3
   45   FORMAT(1X,'F1= ',F7.5,' F2= ',F7.5,' F3= ',F7.5)
      END IF
C
      IF(F3 .GT. F2)  THEN
       A2 = A3
       F2 = F3
       GO TO 20
      END IF
C
      IF((F1 .EQ. F2) .AND. (F2 .EQ. F3))   THEN
          WRITE( TTYOUT, 50 )
   50     FORMAT( 1X,'ORIGINAL BOUNDS OF SEARCH NEED TO BE CHANGED')
          GO TO 87
      END IF
C
      HSTAR = HALFST(F1,F2,F3)
      ASTAR = ALFST(F1,F2,F3,A2)
      IF( ASTAR .EQ. A2 )   ASTAR = ASTAR + 1
C
      CALL THNEWX( RIGHT, XO, ASTAR, MAXTH, MINTH, X, INBOUND )
C
      IF( INBOUND )   THEN
          IF( EFMARR( X ) .EQ.  0.0 )   THEN
               FSTAR = EFM( IDLNUM, ACTNUM, NCOL, NOUTCOL, X, TTYOUT)
               EFMARR(X) = FSTAR
          ELSE
               FSTAR = EFMARR(X)
          END IF
          COUNT = COUNT + 1
      ELSE
          FSTAR = 0.0
      END IF
```

```
C
       ERRCAL = ABS(PSTAR - FSTAR)
       IF(( (ERRCAL .GT. DELTA) .OR. (A2 .GT. 3) )
     &       .AND. ( CCUNT .LT. MAXCCUNT ))   THEN
C
C          FIND NEW STEP LENGTH AND MOVE X0
C
        CALL THA2XC( RIGHT, X0, ASTAR, A2, F1, F2, FSTAR )
C
        GO TO 30
C
       END IF
C
        X  = MINTH
        EFMMAX = 0.0
        DO 55  J = MINTH, MAXTH
            IF( EFMARR(J) .GT. EFMMAX)  THEN
              X = J
              EFMMAX = EFMARR(J)
            END IF
   55   CONTINUE
C
        WRITE(TTYOUT,70) X, EFMMAX
   70 FORMAT(1X,'BEST THRESHOLD,',I3,' GIVES EDGE FIGURE OF MERIT,'
     &          ,F8.5)
C
       IF (DEBUG)  THEN
         WRITE(TTYOUT,75)  CCUNT
   75    FORMAT(1X,'IT TOOK',I3,' ITERATIONS')
       END IF
C
C          CREATE THRESHOLDED FILE
C
       CALL THORFI( ACTNUM, THRNUM, TBUF, NOUTROW, NOUTCOL,
     &                 X, MINTH, MAXTH )
C
   97 CONTINUE
C
       WHILE( DEBUG )
        REWIND(ACTNUM)
        WRITE(TTYOUT, 90)
   90   FORMAT(1X,'ENTER SPECIFIC THRESHOLD (0-255). THRESHOLD',/,
     &          1X,'OUTSIDE RANGE TO STOP')
        READ(TTYIN, *) X
        WRITE(TTYOUT, *) X
        IF((X .LT. 0) .OR. (X .GT. 255)) GO TO 98
        FSTAR = EFM( IDLNUM, ACTNUM, NCOL, NOUTCOL, X, TTYOUT)
        WRITE(TTYOUT,95) X, FSTAR
   95   FORMAT(1X,'FOR THRESHOLD,',I3,' EFM = ',F7.5)
       END WHILE
C
   98 CONTINUE
       CALL UCLOSE(THRNUM)
       CALL UCLOSE(ACTNUM)
C
C
C     AT ABOVE BEST THRESHOLD FIND EFM AND EDGE MAP FOR DIFFERENT IMAGES
C
       ANSWER = 1
       WHILE (ANSWER .EQ. 1)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      CALL THSPEC( X, IDLNUM, ANSWER, NCOL, NOUTCOL, TBUF,
     &                 MINTH, MAXTH, TTYIN, TTYOUT )
C
      END WHILE
C
      CALL UCLOSE(IDLNUM)
   99 CONTINUE
      RETURN
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C -----------------------------------------------------------------------
C                    UNIVERSITY OF KANSAS REMOTE SENSING LAB
C -----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C -----------------------------------------------------------------------
C PROGRAM NAME:EFM               AUTHOR:JEFF WATSON        DATE:2/4/83
C -----------------------------------------------------------------------
C PURPOSE :
C           THIS ALGORITHM CALCULATES THE EDGE FIGURE OF MERIT
C           OF AN ACTUAL EDGE MAP AS PROPOSED BY PRATT,
C           'QUANTATIVE DESIGN AND EVALUATION OF ENHANCEMENT/THRESHOLD
C           EDGE DETECTORS', PROCEEDINGS OF THE IEEE, MAY 1979.
C           THE EDGE FIGURE OF MERIT IS DEFINED BY
C           F = (1.0/MAX(JCOUNT,ICOUNT)) * SUM(1.0/(1.0 + A*DSQ))
C           THIS FIGURE OF MERIT CAN BE USED TO EVALUATE EITHER
C           THE PERFORMANCE OF EDGE DETECTORS OR THE RELATIVE
C           QUALITY OF VARIOUS IMAGE ENHANCEMENT ALGORITHMS
C           WITH RESPECT TO EDGE QUALITY.  HERE AN IDEAL EDGE MAP
C           DEFINES THE EXACT LOCATION OF THE EDGE.  IN THIS
C           PROGRAM I ASSUME THAT THERE IS ONLY ONE EDGE AND IT IS
C           VERTICAL.
C
C -----------------------------------------------------------------------
C                    PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \    DESCRIPTION
C -----------------------------------------------------------------------
C IDLNUM           \I       \R      \ 17-40  \IDEAL FILE NUMBER
C ACTNUM           \I       \R      \ 17-40  \ACTUAL FILE NUMBER
C NCOL             \I       \R      \ 0-512  \NUM COLUMNS IN IDEAL FILE
C NCUT             \I       \R      \ 0-512  \NUM COLUMNS IN ACTUAL FILE
C THR   LD         \I       \R      \ 0-255  \THRESHHOLD
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C -----------------------------------------------------------------------
C                    NON-LOCAL VARIABLES
C -----------------------------------------------------------------------
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C -----------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME           \          DESCRIPTION
C -----------------------------------------------------------------------
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C -----------------------------------------------------------------------
C
```

```
      FUNCTION              , ACTNUM, NCOL, NOUTCOL, T    OLD, TTYOUT)
C
      INTEGER               ACTNUM, NCOL, NOUTCOL, THRH   D, I
      INTEGER      C      AG,IEP,AEP,ACROW(512),IDROW(512)
      INTEGER      JC     , ICOUNT, TTYOUT
      REAL         EFM,  NUM, A, DSQ, DSUM
      LOGICAL      DEBUG
      DATA         FLAG/0/
C
C      SET ERROR CONSTANT
C
      A = 1.0/9.0
C
C
C         SET INITIAL CONDITIONS
C
      JCOUNT = 0
      ICOUNT = 0
      DSUM   = 0.0
      DEBUG  = .TRUE.
C
C         MAIN LOOP FOR PROCESSING DATA
C            THIS LOOP PROCESSES ONE LINE OF THE EDGE IMAGES
C            AT A TIME
C
   10 CONTINUE
        READ(IDLNUM,END=98)(IDROW(I),I = 1,NCOL)
        READ(ACTNUM,END=98)(ACROW(I),I = 1,NOUTCOL)
C       THIS LOOP FINDS THE IDEAL EDGE POINTS FOR THIS LINE
C       (I ASSUMED ONLY ONE EDGE POINT PER LINE IN THE IDEAL IMAGE)
C       AND STORES THEIR LOCATION AND COUNTS THEM
C
       DO 20 COL = 1,NCOL
         IF (IDROW(COL) .EQ. FLAG)    THEN
           IEP = COL - (NCOL - NOUTCOL)/2
           JCOUNT = JCOUNT + 1
           GO TO 25
         END IF
   20  CONTINUE
   25 CONTINUE
C
C
C           THE FOLLOWING LOOP FINDS THE ACTUAL EDGE POINT (AEP)
C           COUNTS THEM, CALCULATES THE SQUARED DISTANCE BETWEEN
C           IT AND THE IEP AND CALCULATES ITS CONTRIBUTION TO
C           THE FIGURE OF MERIT.
C
       DO 30 COL = 1,NOUTCOL
         IF (ACROW(COL) .GE. THRHOLD)    THEN
           AEP     = COL
           ICOUNT  = ICOUNT + 1
           DSQ     = REAL((IEP - AEP)*(IEP - AEP))
           DSUM    = DSUM + 1.0/(1.0 + A*DSQ)
         END IF
   30  CONTINUE
      GO TO 10
C
   98 CONTINUE
C
C         CALCULATE THE EDGE FIGURE OF MERIT
```

```
C
      XNUM    = REAL(ICOUNT)
      IF( JCOUNT .GT. ICOUNT )  XNUM = REAL(JCOUNT)
      EFM = DSUM/XNUM
C
      IF( DEBUG )   THEN
          WRITE(TTYOUT,40) THRHOLD, EFM
   40     FORMAT(1X,'THRESHOLD= ',I4,' EFM= ',F5.3)
      END IF
C
      REWIND(IDLNUM)
      REWIND(ACTNUM)
      RETURN
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C ---------------------------------------------------------------------------
C                     UNIVERSITY OF KANSAS REMOTE SENSING LAB
C ---------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ---------------------------------------------------------------------------
C PROGRAM NAME:THORFI           AUTHOR:JEFF WATSON        DATE:5/25/83
C ---------------------------------------------------------------------------
C PURPOSE :   CREATES THRESHOLDED FILE
C
C
C
C
C
C
C ---------------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ---------------------------------------------------------------------------
C ACTNUM           \INT    \READ  \17-40   \LFN OF ACTUAL IMAGE
C THRNUM           \INT    \READ  \17-40   \LFN OF THRESHOLDED FILE
C TBUF(NOUTCOL)    \INT    \WRITE \        \BUFFER FOR PROCESSING 1
C                  \       \      \        \RECORD AT A TIME
C NOUTROW          \INT    \WRITE \0-512   \NUMBER OF ROWS IN EDGEMAP
C NOUTCOL          \INT    \READ  \0-512   \NUMBER OF COLUMNS IN ACTUAL
C                  \       \      \        \IMAGE
C THRESH           \INT    \READ  \0-255   \THRESHOLD
C MINTH            \INT    \READ  \0       \BLACK
C MAXTH            \INT    \READ  \255     \WHITE
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ---------------------------------------------------------------------------
C                      NON-LOCAL VARIABLES
C ---------------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ---------------------------------------------------------------------------
C                      SUBROUTINES REQUIRED
C    NAME          \          DESCRIPTION
C ---------------------------------------------------------------------------
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C ---------------------------------------------------------------------------
      SUBROUTINE  THORFI( ACTNUM, THRNUM, TBUF, NOUTROW, NOUTCOL,
     &                    THRESH, MINTH, MAXTH )
C
      INTEGER  ACTNUM, THRNUM, THRESH, NOUTROW, NOUTCOL
      INTEGER  MINTH, MAXTH, J, TBUF( NOUTCOL )
C
      NOUTROW = 0
   10 CONTINUE
      READ(ACTNUM, END = 30) (TBUF(J), J = 1, NOUTCOL)
      NOUTROW = NOUTROW + 1
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      DO 20  J = 1, NOUTCOL
       IF( TBUF(J) .GE. THRESH )  THEN
          TBUF(J) = MAXTH
        ELSE
          TBUF(J) = MINTH
       END IF
   20 CONTINUE
      WRITE( THRNUM) (TBUF(J), J = 1, NOUTCOL)
      GO TO 10
C
   30 CONTINUE
      RETURN
      END
EOF..
?
```

```
C ----------------------------------------------------------------------
C                    UNIVERSITY OF KANSAS REMOTE SENSING LAB
C ----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS         REF. # :
C ----------------------------------------------------------------------
C PROGRAM NAME:THNEWX            AUTHOR:JEFF WATSON        DATE:5/25/83
C ----------------------------------------------------------------------
C PURPOSE :    FINDS THE NEXT THRESHOLD TO HAVE ITS EFM CALCULATED
C AND DETERMINES WHETHER THIS THRESHOLD IS WITHIN THE ALLOWABLE
C RANGE OF THRESHOLDS.
C
C
C
C ----------------------------------------------------------------------
C                    PARAMETER DEFINITION
C  NAME              \ TYPE  \ CLASS\ RANGE   \     DESCRIPTION
C ----------------------------------------------------------------------
C RIGHT              \LOG    \READ  \         \PRESENT DIRECTION OF SEARCH
C XO                 \INT    \READ  \0-255    \INITIAL THRESHOLD
C ASTEP              \INT    \READ  \0-255    \DISTANCE FROM XO TO
C                    \       \      \         \THRESHOLD BEING TESTED
C MAXTH              \INT    \READ  \255      \MAXIMUM THRESHOLD
C MINTH              \INT    \READ  \0        \MINIMUM THRESHOLD
C X                  \INT    \WRITE \-255-510\NEXT THRESHOLD TESTED
C INBOUND            \LOG    \W     \         \TRUE IF 0 < X < 255
C                    \       \      \         \
C                    \       \      \         \
C                    \       \      \         \
C                    \       \      \         \
C                    \       \      \         \
C                    \       \      \         \
C ----------------------------------------------------------------------
C                    NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                    \       \      \         \
C                    \       \      \         \
C                    \       \      \         \
C                    \       \      \         \
C ----------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME            \            DESCRIPTION
C ----------------------------------------------------------------------
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C ----------------------------------------------------------------------
      SUBROUTINE  THNEWX( RIGHT, XO, ASTEP, MAXTH, MINTH, X, INBOUND )
C
      INTEGER  XO, ASTEP, MAXTH, MINTH, X
      LOGICAL  RIGHT, INBOUND
C
      IF( RIGHT ) THEN
           X = XO + ASTEP
      ELSE
           X = XO - ASTEP
      END IF
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      IF(( X .LT. MINTH ) .OR. ( X .GT. MAXTH ))   THEN
            INBOUND = .FALSE.
      ELSE
            INBOUND = .TRUE.
      END IF
      RETURN
      END
EOF..
?
```

```
C ----------------------------------------------------------------------
C                    UNIVERSITY OF KANSAS REMOTE SENSING LAB
C ----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ----------------------------------------------------------------------
C PROGRAM NAME:THA2XO          AUTHOR:JEFF WATSON       DATE:5/25/83
C ----------------------------------------------------------------------
C PURPOSE :    TO FIND NEW STEP LENGTH AND MOVE XO FOR QUADRATIC SEARCH
C
C
C
C
C
C ----------------------------------------------------------------------
C                    PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ----------------------------------------------------------------------
C RIGHT            \LOG     \MOD    \        \DIRECTION OF SEARCH
C XO               \INT     \MOD    \0-255   \INITIAL THRESHOLD
C ASTAR            \INT     \READ   \0-255   \DISTANCE FROM XO TO
C                  \        \       \        \QUAD'S BEST GUESS
C A2               \INT     \READ   \0-255   \STEP LENGTH OF SEARCH
C F1               \REAL    \MOD    \0-1     \EFM AT XO
C F2               \REAL    \MOD    \0-1     \EFM AT A2 FROM XO
C FSTAR            \REAL    \READ   \0-1     \EFM AT ASTAR FROM XO
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C ----------------------------------------------------------------------
C                    NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C                  \        \       \        \
C ----------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME            \          DESCRIPTION
C ----------------------------------------------------------------------
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C ----------------------------------------------------------------------
      SUBROUTINE  THA2XO( RIGHT, XO, ASTAR, A2, F1, F2, FSTAR )
C
      INTEGER  XO, ASTAR, A2, XRIGHT, XLEFT, XMARK, ALPHA
      LOGICAL  RIGHT
      REAL     F1, F2, FSTAR
C
      XRIGHT( XMARK, ALPHA ) = XMARK + ALPHA
      XLEFT( XMARK, ALPHA ) = XMARK - ALPHA
C
C
      IF(ASTAR .GE. A2)  THEN
          IF(F2 .GT. FSTAR) THEN
```

```
C
C            MAX IS BETWEEN A1 AND ASTAR
C
              A2 = ASTAR - A2
              IF(RIGHT)  THEN
                    RIGHT = .FALSE.
                    XO    = XRIGHT(XO,ASTAR)
              ELSE
                    RIGHT = .TRUE.
                    XO    = XLEFT(XO,ASTAR)
              END IF
              F1 = FSTAR
C
          ELSE
C
C                  MAX IS BETWEEN A2 AND A3
C
              IF(RIGHT)  XO = XRIGHT(XO,A2)
              IF(.NOT. RIGHT)  XO = XLEFT(XO,A2)
              A2 = ASTAR - A2
              F1 = F2
              F2 = FSTAR
          END IF
C
      ELSE
C
C            ASTAR BETWEEN A1 AND A2
C
           IF(F2 .GT. FSTAR)  THEN
C
C             MAX BETWEEN ASTAR AND A3
C
               IF(RIGHT)   XO = XRIGHT(XO,ASTAR)
               IF(.NOT. RIGHT)   XO = XLEFT(XO,ASTAR)
               A2 = A2 - ASTAR
               F1 = FSTAR
C
           ELSE
C
C            MAX BETWEEN A1 AND A2
C
               IF(RIGHT)  THEN
                     RIGHT = .FALSE.
                     XO    =  XRIGHT(XO,A2)
               ELSE
                     RIGHT = .TRUE.
                     XO    = XLEFT(XO,A2)
               END IF
               A2 = A2 - ASTAR
               F1 = F2
               F2 = FSTAR
           END IF
      END IF
C
      RETURN
      END
ECF..
?
```

```
C -----------------------------------------------------------------------
C                   UNIVERSITY OF KANSAS REMOTE SENSING LAB
C -----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C -----------------------------------------------------------------------
C PROGRAM NAME:THSPEC           AUTHOR:JEFF WATSON        DATE:5/26/83
C -----------------------------------------------------------------------
C PURPOSE :   CREATES A THRESHOLDED FILE AND FINDS THE EFM FOR AN IMAGE
C OF THE SAME SIZE AS THE PREVIOUS 'ACTUAL FILE' .  IT USES THE PREVIOUS
C FILE'S BEST THRESHOLD AS ITS THRESHOLD.
C
C
C
C -----------------------------------------------------------------------
C                        PARAMETER DEFINITION
C  NAME             \ TYPE  \ CLASS\ RANGE  \      DESCRIPTION
C -----------------------------------------------------------------------
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C -----------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C -----------------------------------------------------------------------
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C -----------------------------------------------------------------------
C                        SUBROUTINES REQUIRED
C    NAME            \           DESCRIPTION
C -----------------------------------------------------------------------
C THCRFI            \CREATES THE THRESHOLDED FILE
C OPN               \OPENS FILES AND ASSIGNS LOGICAL FILE NUMBERS
C UCLOSE            \CLOSES FILES OPENED BY OPN
C FILERR            \REPORTS ERROR IN OPENING FILE WITH OPN
C EFM               \FINDS THE EDGE FIGURE OF MERIT FOR A GIVEN THRESHOLD
C                   \
C                   \
C -----------------------------------------------------------------------
      SUBROUTINE  THSPEC( THRESH, IDLNUM, ANSWER, NCOL,
     &        NOUTCOL, TBUF, MINTH, MAXTH, TTYIN, TTYOUT )
C
C
      INTEGER   IDLNUM, ACTNUM, THRNUM, THRESH, NOUTCOL
      INTEGER   TBUF(NOUTCOL), ANSWER, NCOL, MINTH, MAXTH
      INTEGER   TTYIN, TTYOUT, ERRNUM, RECS
C
      INTEGER*1  ACTLFN(18), THRSFN(18)
C
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      LOGICAL  ERR
C
      REAL F, EFM
C
      ERR = .FALSE.
C
C
      WRITE( TTYOUT, 10 )
   10 FORMAT(/,' IS THERE ANOTHER FILTERED IMAGE OF THE SAME SIZE',/,
     &     ,1X,'THAT YOU WOULD LIKE AN EFM AND EDGE MAP OF FOR THIS '
     &     'THRESHOLD?',/,1X,'1 = YES, 0 = NO')
      READ( TTYIN, * ) ANSWER
      WRITE( TTYOUT, * ) ANSWER
C
      IF ( ANSWER .EQ. 0 )   GO TO 60
C
C
      WRITE (TTYOUT, 20)
   20 FORMAT (1X,'ENTER ACTUAL IMAGE FILENAME (MUST BE AN OLD FILE)')
      READ (TTYIN, 30) ACTLFN
   30 FORMAT(18A1)
C
      WRITE(TTYOUT, 40) ACTLFN
   40 FORMAT(1X,18A1)
C
      CALL OPN( ACTNUM, ACTLFN, 'OLD', 'UNF', ERRNUM, ERR )
      IF ( ERR )   THEN
            CALL FILERR( TTYOUT, ACTLFN, ERRNUM )
            GO TO  70
      END IF
C
C
      WRITE (TTYOUT, 45)
   45 FORMAT(1X,'FILENAME FOR THE EDGE MAP (MUST BE A NEW FILE)')
      READ (TTYIN, 30) THRSFN
C
      WRITE(TTYOUT, 40) THRSFN
C
      CALL OPN( THRNUM, THRSFN, 'NEW', 'UNF', ERRNUM, ERR )
      IF ( ERR )   THEN
            CALL FILERR( TTYOUT, THRSFN, ERRNUM )
            GO TO  70

C
C
C       FIND EFM FOR THE SPECIFIC THRESHOLD
C
C
      F = EFM( IOLNUM, ACTNUM, NCOL, NOUTCOL, THRESH, TTYOUT )
      WRITE( TTYOUT, 50 )  THRESH, F
   50 FORMAT(1X, 'THRESHOLD,',I4,' GIVES EDGE FIGURE OF MERIT,',F8.5)
C
C
C       CREATE THRESHOLDED FILE
C
      CALL THORFI (ACTNUM, THRNUM, TBUF, RECS, NOUTCOL,
     &             THRESH, MINTH, MAXTH)
C
      CALL UCLOSE( ACTNUM )
      CALL UCLOSE( THRNUM )
```

```
C
   60 CCNTINUE
   70 CCNTINUE
      RETURN
      END
ECF..
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR ...

```
C ------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ------------------------------------------------------------------------
C PROGRAM NAME:EQUFLT          AUTHOR:J. SCOTT GARDNER DATE:02/27/83
C ------------------------------------------------------------------------
C PURPOSE :    THIS IS THE MAINLINE FOR THE EQUAL WEIGHTED
C FILTER ROUTINE WHICH USES A FILTER
C WINDOW WITH AN EQUAL WEIGHTING FUNCTION.
C
C
C
C ------------------------------------------------------------------------
C                        PARAMETER DEFINITION
C  NAME               \ TYPE   \ CLASS\ RANGE   \      DESCRIPTION
C ------------------------------------------------------------------------
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C ------------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C ------------------------------------------------------------------------
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C                     \        \      \         \
C ------------------------------------------------------------------------
C                        SUBROUTINES REQUIRED
C    NAME               \          DESCRIPTION
C ------------------------------------------------------------------------
C EQUSUB              \DOES THE PROCESSING-- CALLED WITH VARIABLE PARAMETERS
C                     \
C                     \
C                     \
C                     \
C                     \
C                     \
C ------------------------------------------------------------------------
      PROGRAM  EQUFLT
C
      INTEGER  MAXSIZ, MXWSIZ, BUFQUE
C
C
C   BUFQUE = MAXSIZE * MAXWINDOWSIZE
C
      PARAMETER (MAXSIZ=800)
      PARAMETER (MXWSIZ=15)
      PARAMETER (BUFQUE=MAXSIZ*MXWSIZ)
```

```
C
      INTEGER   RECS
      INTEGER   SIZE, OUTSIZ, WNDSIZ
      INTEGER   IN(MAXSIZ), OUT(MAXSIZ)
C
      REAL   QUEUE(BUFQUE)
C
      INTEGER*1   INFNM(18), OUTFNM(18)
C
      INTEGER   TTYIN,TTYOUT,INFC,OUTFC
C
      DATA   TTYIN, TTYOUT   /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER THE FILENAME FOR INPUT (MUST BE AN CLD FILE)')
      READ (TTYIN,510) INFNM
      WRITE (TTYOUT,700) INFNM
700   FORMAT (1X,18A1)
510   FORMAT (18A1)
C
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER THE FILENAME FOR OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUTFNM
      WRITE (TTYOUT,700)  OUTFNM
C
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE INPUT IMAGE ')
      READ (TTYIN,*)  SIZE
      WRITE (TTYOUT,705)  SIZE
705   FORMAT (1X,I4)
C
      IF (SIZE .LE. MAXSIZ) GOTO 12
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
C
C
12    WRITE (TTYOUT,627)
627   FORMAT (1X,'ENTER THE SIZE OF THE FILTER WINDOW')
      READ (TTYIN, *)  WNDSIZ
      WRITE (TTYOUT,705)  WNDSIZ
C
C
20    IF (WNDSIZ .LE. MXWSIZ) GOTO 13
       WRITE (TTYOUT,641) MXWSIZ
641    FORMAT (1X,'* * *  E R R O R  - - MAXIMUM WINDOW SIZE = ',I5)
      GOTO 1010
C
13    OUTSIZ = SIZE - WNDSIZ + 1
C
C
C
C   CALL SUBROUTINE TO DO THE WORK
C
      CALL ECUSUB (INFNM, OUTFNM, QUEUE,
     &  IN, OUT, SIZE, RECS, WNDSIZ, OUTSIZ, TTYOUT)
C
```

```
C
C
      WRITE (TTYOUT,630)
630   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640)  OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
C
      GOTO 1010
C
C
C
1010  STOP
      END
EOF..
?
```

```
C ------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS         REF. # :
C ------------------------------------------------------------------------
C PROGRAM NAME:EQUSUB         AUTHOR:J. SCOTT GARDNER DATE:3/4/83
C ------------------------------------------------------------------------
C PURPOSE :    THIS IS THE SUBROUTINE TO PERFORM
C THE ACTUAL PROCESSING FOR THE EQUAL WEIGHTED
C FILTER ROUTINE.
C
C
C
C ------------------------------------------------------------------------
C                        PARAMETER DEFINITION
C  NAME             \ TYPE   \ CLASS\ RANGE   \     DESCRIPTION
C ------------------------------------------------------------------------
C INFNM             \CH*18   \R      \         \INPUT FILENAME
C OUTFNM            \CH*18   \R      \         \OUTPUT FILENAME
C QUEUE             \R       \R      \         \IMAGE DATA QUEUE
C IN                \I       \W      \         \INPUT BUFFER
C OUT               \I       \W      \         \OUTPUT BUFFER
C SIZE              \I       \R      \         \IMAGE SIZE
C RECS              \I       \W      \         \NUMBER OF RECORDS IN IMAGE
C WNDSIZ            \I       \R      \         \SIZE OF LOCAL AREA WINDOW
C OUTSIZ            \I       \R      \         \SIZE OF OUTPUT IMAGE
C TTYOUT            \I       \R      \         \OUTPUT TO TERMINAL FILECODE
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C ------------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C ------------------------------------------------------------------------
C                   \        \       \         \
C                   \        \       \         \
C                   \        \       \         \
C ------------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C    NAME            \           DESCRIPTION
C ------------------------------------------------------------------------
C OPN               \OPEN FILE AND ASSIGN FILECODE
C UCLOSE            \CLOSE FILES OPENED WITH OPN
C FILERR            \REPORT TYPE OF FILE ERROR
C ------------------------------------------------------------------------
      SUBROUTINE  EQUSUB (INFNM, OUTFNM, QUEUE,
     &  IN, OUT, SIZE, RECS, WNDSIZ, OUTSIZ, TTYOUT)
C
C
C
      INTEGER  SIZE, OUTSIZ, WNDSIZ
C
      INTEGER  ERRNUM
      INTEGER  IN(SIZE), OUT(OUTSIZ)
      INTEGER  REC, WRD, WORD, RECS, CREC, TMPCRC, START
```

```
      INTEGER   WREC, WWORD, QWORD
C
      REAL   QUEUE(WNDSIZ,SIZE)
      REAL   WNDPTS, WNDWGT, TOTALM, XESTMT
C
      INTEGER*1  INFNM(18), OUTFNM(18)
C
      LOGICAL   ERR
C
      INTEGER   TTYOUT,INFC,OUTFC
C
C
C
C
C    OPEN FILES AND CHECK FOR ERRORS
C
      CALL OPN ( INFC, INFNM, 'OLD', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  1009
C
C
      CALL OPN ( OUTFC, OUTFNM, 'NEW', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  2009
C
C
C
      WNDPTS = WNDSIZ * WNDSIZ
C
C
C    CALCULATE THE WEIGHTING FACTOR
C
      WNDWGT = 1.0 / WNDPTS
C
C    INITIALIZE THE CIRCULAR QUEUE
C
      DO 40  REC=1,WNDSIZ
C
       READ (INFC,ERR=4009) (IN(WRD),WRD=1,SIZE)
         DO 30  WORD=1,SIZE
          QUEUE(REC,WORD) = IN(WORD)
30       CONTINUE
40    CONTINUE
C
C
C    BEGIN PROCESSING
C
      RECS = 0
      QREC = 1
C
50    TMPORC = QREC
C
      DO 90  START=1,OUTSIZ
C
      TOTALM = 0.0
C
C
      DO 20  WREC=1,WNDSIZ
       WWORD = 1
       DO 10  QWORD=START, START+WNDSIZ-1
        TOTALM = TOTALM + QUEUE(TMPORC,QWORD) * WNDWGT
       WWORD = WWORD + 1
```

```
10      CCNTINUE
C
C
C
C
        TMPCRC = MCC (TMPCRC, WNDSIZ) + 1
20      CCNTINUE
C
C
C
        XESTMT = TOTALM
C
C    PUT THE FILTEREC VALUE IN THE CUTPUT BUFFER
C
        CUT(START) = INT (XESTMT)
C
C
90        CCNTINUE
C
C
C    WRITE THIS RECORD AND UPDATE THE RECCRC CCUNTER
C
C
        WRITE (CUTFC) (OUT(WRD),WRD=1,CUTSIZ)
C    DCN'T BCTHER TC CHECK FCR A WRITE ERRCR
C.
        RECS = RECS + 1
C
C
C    READ A NEW RECCRD INTC THE QUEUE AND UPDATE THE FRONT-END PCINTER
C
         READ (INFC,END=200,ERR=4009) (IN(WRD),WRD=1,SIZE)
C
        CC 100  WCRD=1,SIZE
          QUEUE(QREC,WORD) = IN(WORD)
100     CCNTINUE
C
        QREC = MCD (QREC, WNDSIZ) + 1
C
        GCTC 50
C
C
C
C    WE'VE REACHED AN END-CF-FILE SC THE CUTPUT NUMBER CF
C    RECCRCS WILL BE WNDSIZ-1 LESS THAN THE NUMBER INPUT
C    WRAP IT ALL UP AND QUIT
C
C
200     CALL UCLOSE (INFC)
        CALL UCLOSE (CUTFC)
C
C
C
        GCTC 1010
C
C
C
1009    CALL  FILERR (TTYOUT, INFNM, ERRNUM)
        GCTC 1010
C
```

```
2009  CALL FILERR (TTYOUT, OUTFNM, ERRNUM)
      GOTO 1010
C
C
4009  WRITE (TTYOUT,660)
660   FORMAT (1X,'* * * ERROR IN READING INPUT IMAGE * * *')
C
1010  RETURN
      END
EOF..
?
```

```
C -----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C -----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C -----------------------------------------------------------------------
C PROGRAM NAME:MEDFLT           AUTHOR:JEFF WATSON        DATE:3/4/83
C -----------------------------------------------------------------------
C PURPOSE :        THIS PROGRAM CREATES AN OUTPUT IMAGE OF MEDIANS OF
C                  VARIABLE SIZED WINDOWS ON THE INPUT IMAGE
C
C
C
C
C -----------------------------------------------------------------------
C                        PARAMETER DEFINITION
C  NAME             \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C -----------------------------------------------------------------------
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C -----------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C -----------------------------------------------------------------------
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C                   \       \      \        \
C -----------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C   NAME              \          DESCRIPTION
C -----------------------------------------------------------------------
C MEDIAN            \PROCESSES THE IMAGE
C                   \
C                   \
C                   \
C                   \
C                   \
C                   \
C -----------------------------------------------------------------------
C
C
      PROGRAM  MEDFLT
C
      INTEGER  MAXSIZ, MXWSIZ, BUFQUE, BUFWND
C
C      BUFQUE = MAXSIZE * MAXWINDOWSIZE
C      BUFWND = MAXWINDOWSIZE * MAXWINDOWSIZE
C
      PARAMETER (MAXSIZ=512)
```

ORIGINAL PAGE IS OF POOR QUALITY

```
      ...WSIZ=20)
      PARAMETER (BUFQUE=MAXSIZ*MXWSIZ)
      PARAMETER (BUFWND=MXWSIZ*MXWSIZ)
C
      INTEGER    RECS, TBUF(MAXSIZ)
      INTEGER  SIZE, OUTSIZ, WNDSIZ, ARRSIZ
      INTEGER  QUEUE(BUFQUE), WINDOW(BUFWND)
C
C
C
      INTEGER*1    INFNM(18), OUTFNM(18)
C
      INTEGER  TTYIN, TTYOUT
      DATA  TTYIN, TTYOUT  /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER THE FILENAME FOR INPUT (MUST BE AN OLD FILE)')
      READ (TTYIN,510) INFNM
510   FORMAT (18A1)
C
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER THE FILENAME FOR OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUTFNM
C
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE INPUT IMAGE ')
      READ (TTYIN,*)  SIZE
C
      IF (SIZE .GT. MAXSIZ) THEN
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
      END IF
C
C
      WRITE (TTYOUT,627)
627   FORMAT (1X,'ENTER THE SIZE OF THE FILTER WINDOW')
      READ (TTYIN, *)  WNDSIZ
C
      IF (WNDSIZ .GT. MXWSIZ) THEN
       WRITE (TTYOUT,628) MXWSIZ
628    FORMAT (1X,'* * *  E R R O R  - - MAXIMUM WINDOW SIZE = ',I5)
      GOTO 1010
      END IF
C
13    OUTSIZ = SIZE - WNDSIZ + 1
      ARRSIZ = WNDSIZ * WNDSIZ
      RECS = 0
C
C
C    CALL SUBROUTINE TO DO THE WORK
C
      CALL MEDIAN( INFNM, OUTFNM, ARRSIZ, SIZE, RECS,
     &      WNDSIZ, OUTSIZ, TTYOUT, QUEUE, WINDOW, TBUF)
C
C
C
```

```
      WRITE (TTYOUT,630)
630   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640)  OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
1010  STOP
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C --------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C --------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C --------------------------------------------------------------------------
C PROGRAM NAME:MEDIAN            AUTHOR:JEFF WATSON        DATE:3/5/83
C --------------------------------------------------------------------------
C PURPOSE :     DOES THE PROCESSING FOR THE MEDIAN FILTER AFTER BEING
C CALLED BY PROGRAM MEDFLT.
C
C
C
C --------------------------------------------------------------------------
C                         PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C --------------------------------------------------------------------------
C INFILE           \CH*18  \R      \         \INPUT FILENAME
C OUTFILE          \CH*18  \R      \         \OUTPUT FILENAME
C ARRSIZE          \I      \R      \         \NUMBER OF ELEMENTS IN WINDOW
C NCOLS            \I      \R      \         \COLUMNS IN INPUT IMAGE
C NOUTROW          \I      \W      \         \ROWS IN OUTPUT IMAGE
C WINDSIZE         \I      \R      \         \WINDOWSIZE
C NOUTCOL          \I      \R      \         \COLUMNS IN OUTPUT IMAGE
C TTYOUT           \I      \R      \         \OUTPUT TO TERMINAL FILECODE
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C --------------------------------------------------------------------------
C                         NON-LOCAL VARIABLES
C --------------------------------------------------------------------------
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C --------------------------------------------------------------------------
C                         SUBROUTINES REQUIRED
C     NAME           \         DESCRIPTION
C --------------------------------------------------------------------------
C MEDFNC           \FINDS MEDIAN OF LOCAL WINDOW
C OPN              \OPENS FILE AND ASSIGNS LOGICAL UNIT
C UCLOSE           \CLOSES FILES OPENED WITH OPN
C                  \
C                  \
C                  \
C                  \
C --------------------------------------------------------------------------
C
C
      SUBROUTINE MEDIAN ( INFILE, OUTFILE, ARRSIZE, NCOLS, NOUTROW,
     &           WINDSIZE, NOUTCOL, TTYOUT, Q, ARRVAL, TBUF)
C
      LOGICAL  ERR
C
      INTEGER  ARRSIZE, NCOLS, NOUTCOL, WINDSIZE
      INTEGER I, J, K, ERRNUM, START, ARRPOS, OUTCOL, NROWS
      INTEGER  NOUTROW,TTYOUT, ARRVAL(ARRSIZE), TBUF(NCOLS)
      INTEGER  Q(WINDSIZE,NCOLS), INFNUM, OUTNUM, MED, MEDPOS
      INTEGER*1 INFILE(18), OUTFILE(18)
```

```
C
      NOUTROW = 0
C
      CALL OPN (INFNUM, INFILE, 'OLD', 'UNF', ERRNUM, ERR)
      CALL OPN (OUTNUM, OUTFILE, 'NEW', 'UNF', ERRNUM, ERR)
C
C          INITIALIZE QUEUE
C
      DO 30  I = 1,WINDSIZE
       READ(INFNUM,END = 98) (TBUF(K), K = 1, NCOLS)
       DO 20  J = 1,NCOLS
        Q(I,J) = TBUF(J)
   20 CONTINUE
   30 CONTINUE
C
C
C                MAIN PROCESSING
C
C
      MEDPOS = ARRSIZE/2
C
   40 CONTINUE
       DO 70  START = 1, NOUTCOL
         DO 60  I = 1, WINDSIZE
          DO 50  J = START, START + WINDSIZE - 1
            ARRPOS = WINDSIZE*(I-1) + J-START+1
            ARRVAL(ARRPOS) = Q(I,J)
   50     CONTINUE
   60    CONTINUE
      CALL MEDFNC(ARRVAL,MED,ARRSIZE, MEDPOS)
      OUTCOL = START
      TBUF(OUTCOL) = MED
   70  CONTINUE
C
      NOUTROW = NOUTROW + 1
      WRITE(OUTNUM) (TBUF(K), K = 1, NOUTCOL)
C
C          UPDATE QUEUE
C
       DO 80  I = 1 , WINDSIZE
        IF(I .LT. WINDSIZE) THEN
         DO 85  J = 1, NCOLS
           Q(I,J) = Q(I+1,J)
   85    CONTINUE
         ELSE
          READ(INFNUM,END = 98) (TBUF(K), K = 1, NCOLS)
           DO 87  J = 1,NCOLS
             Q(I,J) = TBUF(J)
   87      CONTINUE
         END IF
   80 CONTINUE
      GO TO 40
C
C         WE HAVE REACHED THE END OF THE INPUT FILE
C
   98 CONTINUE
      CALL UCLOSE(INFNUM)
      CALL UCLOSE(OUTNUM)
      RETURN
      END
```

```
C ----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ----------------------------------------------------------------------
C PROGRAM NAME:MEDFNC           AUTHOR:JEFF WATSON        DATE:3/9/83
C ----------------------------------------------------------------------
C PURPOSE :       FINDS THE MEDIAN VALUE OF THE ARRAY PASSED TO IT.
C HERE THE ARRAY IS THE LOCAL AREA WINDOW
C
C
C
C
C ----------------------------------------------------------------------
C                         PARAMETER DEFINITION
C  NAME              \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ----------------------------------------------------------------------
C ARR                \I       \R      \        \ARRAY OF ELEMENTS IN WINDOW
C MED                \I       \W      \0-255   \MEDIAN OF ARRAY
C SIZE               \I       \R      \1-400   \NUMBER OF ELEMENTS IN ARRAY
C MEDPOS             \INT     \READ   \1-200   \POSITION OF MEDIAN VALUE IN
C                    \        \       \        \ORDERED ARRAY OF PIXEL VALUES
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C ----------------------------------------------------------------------
C                         NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C                    \        \       \        \
C ----------------------------------------------------------------------
C                         SUBROUTINES REQUIRED
C    NAME             \          DESCRIPTION
C ----------------------------------------------------------------------
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C ----------------------------------------------------------------------
C
      SUBROUTINE  MEDFNC(ARR, MED, SIZE, MEDPOS)
C
      INTEGER SIZE, I, MED, MEDPOS
      INTEGER  ARR(SIZE), PIXARR( 0: 255 ), COUNT
C
      FOR  I = 0, 255
          PIXARR( I ) = 0
      END FOR
C
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      FOR  I = 1, SIZE
         PIXARR( ARR( I ) ) = PIXARR( ARR( I ) ) + 1
      END FOR
C
      I = -1
      COUNT = 0
      DO
         I = I + 1
         COUNT = COUNT + PIXARR( I )
      UNTIL ( COUNT .GT. MEDPOS )
C
      MED = I
C
      RETURN
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
C ------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ------------------------------------------------------------------------
C PROGRAM NAME:LEEFLT          AUTHOR:J. SCOTT GARDNER DATE:02/12/83
C ------------------------------------------------------------------------
C PURPOSE :   THIS PROGRAM INCORPORATES LEE'S MULTIPLICATIVE
C NOISE MODEL INTO HIS EDGE PRESERVING ADAPTIVE FILTER.
C
C
C
C
C ------------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME                \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ------------------------------------------------------------------------
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C ------------------------------------------------------------------------
C                       NON-LOCAL VARIABLES
C ------------------------------------------------------------------------
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C                      \       \      \        \
C ------------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C    NAME               \         DESCRIPTION
C ------------------------------------------------------------------------
C LEESUB               \CALLED WITH VARIABLE PARAMETERS
C                      \
C                      \
C                      \
C                      \
C                      \
C                      \
C                      \
C ------------------------------------------------------------------------
      PROGRAM  LEEFLT
C
      INTEGER  MAXSIZ, MXWSIZ, BUFCUE, BUFWND, BUFMWN
C
C
C   BUFCUE = MAXSIZE * MAXWINDOWSIZE
C   BUFMWN = 8 * MAXWINDOWSIZE * MAXWINDOWSIZE
C   BUFWND = MAXWINDOWSIZE * MAXWINDOWSIZE
C
      PARAMETER (MAXSIZ=512)
```

```
      IF (WNDSIZ .LE. MXWSIZ) GOTO 13
       WRITE (TTYOUT,628) MXWSIZ
628    FORMAT (1X,'* * *  E R R O R  - - MAXIMUM WINDOW SIZE = ',I5)
      GOTO 1010
C
13    OUTSIZ = SIZE - WNDSIZ + 1
C
C
C   CALL SUBROUTINE TO DO THE WORK
C
      CALL LEESUB (INFNM, OUTFNM, QUEUE, WINDOW, MSKWND, IN, OUT,
     &  SIZE, RECS, WNDSIZ, OUTSIZ, THSHLD, NUMLKS, TTYOUT)
C
C
C
      WRITE (TTYOUT,630)
630   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640)  OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
C
      GOTO 1010
C
C
C
1010  STOP
      END
EOF..
?
```

```
C ------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ------------------------------------------------------------------------
C PROGRAM NAME:LEESUB          AUTHOR:J. SCOTT GARDNER DATE:02/12/83
C ------------------------------------------------------------------------
C PURPOSE :   THIS SUBROUTINE DOES THE PROCESSING
C FOR LEE'S EDGE FILTER AFTER BEING CALLED
C BY PROGRAM LEEFLT WITH VARIABLE ARRAY DIMENSIONS.
C
C
C
C ------------------------------------------------------------------------
C                        PARAMETER DEFINITION
C  NAME              \ TYPE   \ CLASS\ RANGE   \      DESCRIPTION
C ------------------------------------------------------------------------
C INFNM             \CH*18   \R       \         \INPUT FILENAME
C OUTFNM            \CH*18   \R       \         \OUTPUT FILENAME
C QUEUE             \R       \R       \         \IMAGE DATA QUEUE
C WINDOW            \R       \R/W     \         \LOCAL AREA WINDOW
C MSKWND            \I       \W       \         \EDGE TEMPLATES
C IN                \I       \W       \         \INPUT BUFFER
C OUT               \I       \W       \         \OUTPUT BUFFER
C SIZE              \I       \R       \         \IMAGE SIZE
C RECS              \I       \W       \         \NUMBER OF RECORDS IN IMAGE
C WNDSIZ            \I       \R       \         \SIZE OF LOCAL AREA WINDOW
C OUTSIZ            \I       \R       \         \SIZE OF OUTPUT IMAGE
C THSHLD            \R       \R       \         \EDGE THRESHOLD
C NUMLKS            \R       \R       \         \NUMBER OF LOOKS
C TTYOUT            \I       \R       \         \OUTPUT TO TERMINAL FILECODE
C ------------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C ------------------------------------------------------------------------
C                   \        \        \         \
C                   \        \        \         \
C                   \        \        \         \ .
C ------------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C    NAME             \           DESCRIPTION
C ------------------------------------------------------------------------
C GNMSKS            \GENERATES MASKS
C LCSTAT            \COMPUTES LOCAL STATISTICS OF WINDOW
C SUBMSK            \GETS A 3X3 LOCAL MEAN FROM WINDOW
C FNDEDG            \GETS EDGE ORIENTATION AND SELECTS A MASK
C EGSTAT            \COMPUTES LOCAL STATS OUTSIDE THE EDGE
C ESTMTE            \FINDS AN ESTIMATE FOR THE SIGNAL
C OPN               \OPENS FILE AND ASSIGNS LOGICAL UNIT
C UCLOSE            \CLOSES FILES OPENED WITH OPN
C ------------------------------------------------------------------------
      SUBROUTINE  LEESUB (INFNM, OUTFNM, QUEUE, WINDOW, MSKWND,
     &  IN, OUT, SIZE, RECS, WNDSIZ, OUTSIZ, THSHLD, NUMLKS, TTYOUT)
C
C
C
      INTEGER  SIZE, OUTSIZ, WNDSIZ
C
      INTEGER  MSK3X3(3,3,4), SBAREA(3,3), MSKWND(8,WNDSIZ,WNDSIZ)
      INTEGER  IN(SIZE), OUT(OUTSIZ)
      INTEGER  REC, WRD, WORD, RECS, OREC, TMPORC, START, IERR
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      INTEGER  MSKNUM, I, J, K, ERRNUM
C
      REAL  QUEUE(WNDSIZ,SIZE), WINDOW(WNDSIZ,WNDSIZ)
      REAL  WNDPTS, SUBPTS, Z, ZMEAN, VARZ, NUMEKS, THSHLD, XESTMT
      REAL  THRSMX, THRSMI, TTLTMN, TTLTSD, TTLHMN
      REAL  EGTHRS, NUMPTS, NMHPTS
C
      INTEGER*1  INFNM(18), OUTFNM(18)
C
      LOGICAL  ERR
C
      INTEGER  TTYOUT,INFC,OUTFC
C
C
C     INITIALIZE THE 3X3 GRADIENT WINDOWS
C
C
      DATA  (((MSK3X3(I,J,K), I=1,3), J=1,3), K=1,4)
     &  /1,1,1,0,0,0,-1,-1,-1, 0,-1,-1,1,0,-1,1,1,0,
     &   1,0,-1,1,0,-1,1,0,-1, 1,1,0,1,0,-1,0,-1,-1/
C
C
C     INITIALIZE ZMEAN*ZMEAN/VARZ  MIN AND MAX AND TOTALS
C
      THRSMI = 100000
      THRSMX = -100000
      TTLTMN = 0.0
      TTLTSD = 0.0
      TTLHMN = 0.0
      NMHPTS = 0.0
C
C     DETERMINE THE NUMBER OF POINTS IN IMAGE (ASSUMED SQUARE)
C
      NUMPTS = OUTSIZ * OUTSIZ
C
C
C
C     OPEN FILES AND CHECK FOR ERRORS
C
      CALL OPN ( INFC, INFNM, 'OLD', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  1009
C
C
      CALL OPN ( OUTFC, OUTFNM, 'NEW', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  2009
C
C
C
C     FIRST, GENERATE THE EDGE TEMPLATES
      CALL  GNMSKS (MSKWND, WNDSIZ)
C
C
      WNDPTS = WNDSIZ * WNDSIZ
      SUBPTS = (WNDSIZ/2+1) * WNDSIZ
C
C     INITIALIZE THE CIRCULAR QUEUE
C
      DO 40  REC=1,WNDSIZ
C
       READ (INFC,ERR=3009)  (IN(WRD),WRD=1,SIZE)
```

```
          DO 30  WORD=1,SIZE
            QUEUE(REC,WORD) = IN(WORD)
30        CONTINUE
40      CONTINUE
C
C
C     BEGIN PROCESSING
C
        RECS = 0
        QREC = 1
C
50      TMPQRC = QREC
C
        DO 90  START=1,OUTSIZ
C
C     GET THE LOCAL STATISTICS FOR THE AREA DEFINED BY THE WINDOW
C     AND FILL THE WINDOW ARRAY TO BE USED LATER IF AN EDGE IS FOUND
C
        CALL LCSTAT (QUEUE, WINDOW, TMPQRC, START, WNDSIZ, WNDPTS,
     &  SIZE, ZMEAN, VARZ, Z)
C
C
C     FIND THE EDGE THRESHOLD VALUE
C
        EGTHRS = ZMEAN*ZMEAN/VARZ
C
C
C     UPDATE RUNNING SUMS FOR DETERMINING THRESHOLD
C
        TTLTMN = TTLTMN + EGTHRS/NUMPTS
        TTLTSD = TTLTSD + EGTHRS*EGTHRS/NUMPTS
        THRSMI = MIN (THRSMI, EGTHRS)
        THRSMX = MAX (THRSMX, EGTHRS)
C
C     DETERMINE IF AN EDGE EXISTS BASED ON THE LOCAL STATISTICS,
C     THE NUMBER OF LOOKS AND THE USER SPECIFIED THRESHOLD.
C
        IF (EGTHRS .LE. NUMLKS-THSHLD)  GOTO 44
        TTLHMN=TTLHMN + EGTHRS
        NMHPTS = NMHPTS + 1.0
        GOTO 60
C
C
C     WE HAVE AN EDGE, SO PROCEED WITH THE EDGE FILTERING
C
C     GET THE 3X3 SUBAREA LOCAL MEAN
C
44      CALL  SUBMSK  (WINDOW, SBAREA, WNDSIZ)
C
C
C     FIND THE EDGE ORIENTATION AND DETERMINE WHICH EDGE
C     TEMPLATE TO USE IN CALCULATING OUR NEW LOCAL MEAN
C
        CALL  FNDEDG  (SBAREA, MSK3X3, MSKNUM)
C
C
C     COMPUTE A NEW LOCAL MEAN AND VARIANCE USING THE APPROPRIATE
C     TEMPLATE
C
        CALL EGSTAT  (WINDOW, MSKWND, WNDSIZ, MSKNUM,
```

```
     5   ZMEAN, VARZ, Z)
C
C
C    FIND AN ESTIMATE FOR THE SIGNAL, XESTMT
C
60     CALL  ESTMTE  (ZMEAN, VARZ, Z, NUMLKS, XESTMT)
C
       IF (XESTMT .GT. 255.0)  XESTMT=255.0
C
C    PUT THE FILTERED VALUE IN THE OUTPUT BUFFER
C
       OUT(START) = INT (XESTMT)
C
C
90       CONTINUE
C
C
C    WRITE THIS RECORD AND UPDATE THE RECORD COUNTER
C
       WRITE (OUTFC) (OUT(WRD),WRD=1,OUTSIZ)
C
C    DON'T BOTHER TO CHECK FOR A WRITE ERROR
C
       RECS = RECS + 1
C
C
C    READ A NEW RECORD INTO THE QUEUE AND UPDATE THE FRONT-END POINTER
C
        READ (INFC,END=200,ERR=3009) (IN(WRD),WRD=1,SIZE)
C
       DO 100  WORD=1,SIZE
         QUEUE(QREC,WORD) = IN(WORD)
100    CONTINUE
C
       QREC = MOD (QREC, WNDSIZ) + 1
C
       GOTO 50
C
C
C
C    WE'VE REACHED AN END-OF-FILE SO THE OUTPUT NUMBER OF
C    RECORDS WILL BE WNDSIZ-1 LESS THAN THE NUMBER INPUT
C    WRAP IT ALL UP AND QUIT
C
C
200    CALL UCLOSE (INFC)
       CALL UCLOSE (OUTFC)
C
       WRITE (TTYOUT,635) THRSMI,THRSMX,TTLTMN,TTLTSD-TTLTMN*TTLTMN
635    FORMAT (1X,'EDGE THRESHOLD MIN = ',F9.3,
      &  /1X,'EDGE THRESHOLD MAX = ',F9.3,
      &  /1X,'EDGE THRESHOLD MEAN = ',F9.3,
      &  /1X,'EDGE THRESHOLD STANDARD DEVIATION = ',F9.3)
C
       WRITE (TTYOUT,637) TTLHMN/NMHPTS
637    FORMAT (1X,'HOMOGENEOUS AREA MEAN = ',F9.3)
       WRITE (TTYOUT,638) NMHPTS*100.0/NUMPTS
638    FORMAT (1X,F9.3,' PERCENT OF THE IMAGE WAS HOMOGENEOUS')
C
       GOTO 1010
```

```
C
C
C
1009  CALL  FILERR (TTYOUT, INFNM, ERRNUM)
      GOTO 1010
C
2009  CALL FILERR (TTYOUT, OUTFNM, ERRNUM)
      GOTO 1010
C
3009  WRITE (TTYOUT,660)
660   FORMAT (1X,'* * * ERROR IN READING INPUT IMAGE * * *')
C
1010  RETURN
      END
EOF..
?
```

```
C ---------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ---------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ---------------------------------------------------------------------
C PROGRAM NAME:ESTMTE          AUTHOR:J. SCOTT GARDNER DATE:02/14/83
C ---------------------------------------------------------------------
C PURPOSE :   THIS SUBROUTINE ESTIMATES THE SIGNAL FROM THE
C LOCAL MEAN AND VARIANCE.
C
C
C
C
C ---------------------------------------------------------------------
C                     PARAMETER DEFINITION
C  NAME          \ TYPE  \ CLASS\ RANGE   \     DESCRIPTION
C ---------------------------------------------------------------------
C ZMEAN          \R      \R     \         \LOCAL MEAN
C VARZ           \R      \R     \         \LOCAL VARIANCE
C Z              \R      \R     \         \VALUE OF CENTER PIXEL
C XESTMT         \R      \W     \         \ESTIMATE FOR SIGNAL
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C ---------------------------------------------------------------------
C                     NON-LOCAL VARIABLES
C ---------------------------------------------------------------------
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C                \       \      \         \
C ---------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME          \         DESCRIPTION
C ---------------------------------------------------------------------
C                \
C                \
C                \
C                \
C                \
C                \
C                \
C ---------------------------------------------------------------------
      SUBROUTINE  ESTMTE  (ZMEAN, VARZ, Z, NUMLKS, XESTMT)
C
C
      REAL  ZMEAN, VARV, VARZ, Z, XESTMT, XMEAN, VARX, K
      REAL  NUMLKS
C
C
C   SOURCE - -  SPECKLE ANALYSIS AND SMOOTHING OF SYNTHETIC
C   APERTURE RADAR IMAGES
C   JONG-SEN LEE   COMPUTER GRAPHICS AND IMAGE PR  ESSING 17,24-32(1981)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C    SIGNAL MEAN  =  IMAGE MEAN  /  NOISE MEAN
C    ASSUME  NOISE MEAN  = 1
C
      XMEAN = ZMEAN
C
C
      VARV = 1.0 / NUMLKS
C
C
      VARX = (VARZ + ZMEAN*ZMEAN) / (VARV + 1.0) - XMEAN*XMEAN
C
C
C    Z CAN BE LINEARIZED BY THE FIRST ORDER TAYLOR SERIES
C    EXPANSION ABOUT (XMEAN, VMEAN)
C      Z = VMEAN*X + XMEAN * (V - VMEAN)
C
C   FROM THIS AN ESTIMATE FOR X IS DEVELOPED
C
C
      K = VARX / (XMEAN * XMEAN * VARV + VARX)
C
      XESTMT = XMEAN + K * (Z - XMEAN)
C
C
      RETURN
      END
EOF..
?
```

```
C ------------------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATI...            LAB
C ------------------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ------------------------------------------------------------------------------------
C PROGRAM NAME:ADPFLT            AUTHOR:J. SCOTT GARDNER DATE:02/.../
C ------------------------------------------------------------------------------------
C PURPOSE :    THIS IS THE MAINLINE FOR THE ADAPTIVE FILTER
C ROUTINE WHICH USES A FILTER
C WINDOW WITH AN EXPONENTIAL WEIGHTING FUNCTION.
C
C
C
C ------------------------------------------------------------------------------------
C                         PARAMETER DEFINITION
C  NAME              \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ------------------------------------------------------------------------------------
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C ------------------------------------------------------------------------------------
C                         NON-LOCAL VARIABLES
C ------------------------------------------------------------------------------------
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C                    \       \      \        \
C ------------------------------------------------------------------------------------
C                        SUBROUTINES REQUIRED
C   NAME              \           DESCRIPTION
C ------------------------------------------------------------------------------------
C ADPSUB             \DOES THE PROCESSING-- CALLED WITH VARIABLE PARAMETERS
C                    \
C                    \
C                    \
C                    \
C                    \
C                    \
C ------------------------------------------------------------------------------------
      PROGRAM  ADPFLT
C
      INTEGER  MAXSIZ, MXWSIZ, BUFQUE, BUFWND, MXFLTS, BUFFWN
C
C
C   BUFQUE = MAXSIZE * MAXWINDOWSIZE
C   BUFWND = MAXWINDOWSIZE * MAXWINDOWSIZE
C   BUFFWN IS THE BUFFER FOR THE FILTERS ARRAY
C
      PARAMETER (MAXSIZ=512)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
              TER (MXWSIZ=15)
              TER (MXFLTS=25)
      PARAMETER (BUFQUE=MAXSIZ*MXWSIZ)
      PARAMETER  (BUFWND=MXWSIZ*MXWSIZ)
      PARAMETER  (BUFFWN=MXFLTS * MXWSIZ * MXWSIZ)
C
      INTEGER  RECS, FLTNUM
      INTEGER  IN(MAXSIZ), OUT(MAXSIZ), SIZE, OUTSIZ, WNDSIZ
      INTEGER  FLTHST (MXFLTS)
C
      REAL  QUEUE(BUFQUE), WINDOW(BUFWND), FLTRS(BUFFWN)
      REAL  NUMLKS, NMAX, NMIN, DIST(BUFWND), EQURES
C
      INTEGER*1  INFNM(18), OUTFNM(18), OUT2FN(18)
C
      INTEGER  TTYIN,TTYOUT,INFC,OUTFC, OUT2FC
C
      DATA  TTYIN, TTYOUT  /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER THE FILENAME FOR INPUT (MUST BE AN OLD FILE)')
      READ (TTYIN,510) INFNM
510   FORMAT (18A1)
      WRITE (TTYOUT,700)  INFNM
700   FORMAT (1X,18A1)
C
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER THE FILENAME FOR OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUTFNM
      WRITE (TTYOUT,700)  OUTFNM
C
      WRITE (TTYOUT,603)
603   FORMAT (1X,'FILENAME FOR FILTER OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUT2FN
      WRITE (TTYOUT,700)  OUT2FN
C
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE INPUT IMAGE ')
      READ (TTYIN,*)  SIZE
      WRITE (TTYOUT,705)  SIZE
705   FORMAT (1X,I4)
C
      IF (SIZE .LE. MAXSIZ) GOTO 12
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
C
12    WRITE (TTYOUT,625)
625   FORMAT (1X,'ENTER THE NUMBER OF LOOKS ')
      READ (TTYIN,*)  NUMLKS
      WRITE (TTYOUT,710)  NUMLKS
710   FORMAT (1X,F6.3)
C
      WRITE (TTYOUT,637)
637   FORMAT (1X,'ENTER THE EQUIVALENT RESOLUTION ')
      READ (TTYIN,*)  EQURES
      WRITE (TTYOUT,710)  EQURES
```

```
      WRITE (TTYOUT,645)
      FORMAT (1X,'ENTER THE MINIMUM NUMBER OF LOCKS ')
      READ (TTYIN,*) NMIN
      WRITE (TTYOUT,710) NMIN
C
      WRITE (TTYOUT,647)
647   FORMAT (1X,'ENTER THE MAXIMUM NUMBER OF LOCKS ')
      READ (TTYIN,*) NMAX
      WRITE (TTYOUT,710) NMAX
C
      WRITE (TTYOUT,649)
649   FORMAT (1X,'ENTER THE NUMBER OF FILTERS ')
      READ (TTYIN,*) FLTNUM
      WRITE (TTYOUT,705) FLTNUM
C
C
      WRITE (TTYOUT,627)
627   FORMAT (1X,'ENTER THE SIZE OF THE FILTER WINDOW',
     & /10X,'(THIS PARAMETER MUST BE ODD)')
      READ (TTYIN, *) WNDSIZ
      WRITE (TTYOUT,705) WNDSIZ
C
      IF (MOD(WNDSIZ,2) .NE. 0) GOTO 20
      WNDSIZ = WNDSIZ + 1
      WRITE (TTYOUT,632) WNDSIZ
632   FORMAT (1X,'THAT IS NOT AN ODD NUMBER. I WILL USE ',I2,' INSTEAD')
C
20    IF (WNDSIZ .LE. MXWSIZ) GOTO 13
       WRITE (TTYOUT,641) MXWSIZ
641    FORMAT (1X,'* * *  E R R O R  - - MAXIMUM WINDOW SIZE = ',I5)
      GOTO 1010
C
13    OUTSIZ = SIZE - WNDSIZ + 1
C
C
C
C    CALL SUBROUTINE TO DO THE WORK
C
      CALL ADPSUB (INFNM, OUTFNM, OUT2FN, QUEUE, WINDOW, FLTRS, DIST,
     &  IN, OUT, FLTHST, SIZE, FLTNUM, RECS, WNDSIZ, OUTSIZ, NUMLKS,
     &  NMIN, NMAX, EQURES, TTYOUT)
C
C
C
      WRITE (TTYOUT,630)
630   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640) OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
C
      GOTO 1010
C
C
C
1010  STOP
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C -----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C -----------------------------------------------------------------------
C  PROGRAM SUITE : NOISE FILTERS         REF. # :
C -----------------------------------------------------------------------
C  PROGRAM NAME:ADPSUB            AUTHOR:J. SCOTT GARDNER DATE:3/4/83
C -----------------------------------------------------------------------
C  PURPOSE :    THIS IS THE SUBROUTINE TO PERFORM
C  THE ACTUAL PROCESSING FOR THE ADAPTIVE WEINER
C  FILTER ROUTINE.
C
C -----------------------------------------------------------------------
C                     PARAMETER DEFINITION
C   NAME          \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C -----------------------------------------------------------------------
C  INFNM          \CH*18  \R      \        \INPUT FILENAME
C  OUTFNM         \CH*18  \R      \        \OUTPUT FILENAME
C  OUT2FN         \CH*8   \R      \        \FILTER OUTPUT FILENAME
C  QUEUE          \R      \R      \        \IMAGE DATA QUEUE
C  WINDOW         \R      \R/W    \        \LOCAL AREA WINDOW
C  FLTRS          \R      \R/W    \        \FILTERS ARRAY
C  DIST           \R      \R/W    \        \DISTANCE ARRAY
C  IN             \I      \W      \        \INPUT BUFFER
C  OUT            \I      \W      \        \OUTPUT BUFFER
C  FLTHST         \I      \R      \        \FILTER USAGE HISTOGRAM
C  SIZE           \I      \R      \        \IMAGE SIZE
C  FLTNUM         \I      \R      \        \NUMBER OF FILTERS
C  RECS           \I      \W      \        \NUMBER OF RECORDS IN IMAGE
C  WNDSIZ         \I      \R      \        \SIZE OF LOCAL AREA WINDOW
C  OUTSIZ         \I      \R      \        \SIZE OF OUTPUT IMAGE
C  NUMLKS         \R      \R      \        \NUMBER OF LOOKS
C  NMIN           \R      \R      \        \MINIMUM NUMBER OF LOOKS
C  NMAX           \R      \R      \        \MAXIMUM NUMBER OF LOOKS
C  EQURES         \R      \R      \        \EQUIVALENT RESOLUTION
C  TTYOUT         \I      \R      \        \OUTPUT TO TERMINAL FILECODE
C -----------------------------------------------------------------------
C                     NON-LOCAL VARIABLES
C -----------------------------------------------------------------------
C                 \       \       \        \
C -----------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C     NAME          \          DESCRIPTION
C -----------------------------------------------------------------------
C  GENFLT         \GENERATE THE FILTERS AND PRINT THEM
C  LCSTAT         \GETS LOCAL STATS AND FILLS A WINDOW ARRAY FROM QUEUE
C  FILTER         \APPLY THE PROPER FILTER TO THE LOCAL AREA
C  OPN            \OPEN FILE AND ASSIGN FILECODE
C  UCLOSE         \CLOSE FILES OPENED WITH OPN
C  FILERR         \REPORT TYPE OF FILE ERROR
C -----------------------------------------------------------------------
      SUBROUTINE  ADPSUB (INFNM, OUTFNM, OUT2FN, QUEUE, WINDOW, FLTRS,
     &  DIST, IN, OUT, FLTHST, SIZE, FLTNUM, RECS, WNDSIZ, OUTSIZ,
     &  NUMLKS, NMIN, NMAX, EQURES, TTYOUT)
C
C
C
      INTEGER  SIZE, OUTSIZ, WNDSIZ
C
      INTEGER  FLTNUM, FLT, ERRNUM
      INTEGER  IN(SIZE), OUT(OUTSIZ), FLTHST (FLTNUM)
```

```
      INTEGER  REC, WRD, WORD, RECS, OREC, TMPREC, START, IERR
C
      REAL  QUEUE(WNDSIZ,SIZE), WINDOW(WNDSIZ,WNDSIZ)
      REAL  FLTRS (FLTNUM, WNDSIZ, WNDSIZ), DIST(WNDSIZ,WNDSIZ)
      REAL  WNDPTS, Z, ZMEAN, VARZ, NUMLKS, THSHLD, XESTMT
      REAL  THRSMX, THRSMI, TTLTMN, TTLTSD, DELTAN, NMIN, NMAX
      REAL  FGTHRS, NUMPTS, EQURES
C
      INTEGER*1  INFNM(18), OUTFNM(18), OUT2FN(18)
C
      LOGICAL  ERR, RNGERR
C
      INTEGER  TTYOUT,INFC,OUTFC,OUT2FC
C
C
C
C     INITIALIZE ZMEAN*ZMEAN/VARZ  MIN AND MAX AND TOTALS
C
      THRSMI = 100000
      THRSMX = -100000
      TTLTMN = 0.0
      TTLTSD = 0.0
C
C
      RNGERR = .FALSE.
C
C     ZERO HIST ARRAY
C
      DO 5 FLT=1,FLTNUM
       FLTHST(FLT) = 0
5     CONTINUE
C
C
C     DETERMINE THE NUMBER OF POINTS IN IMAGE (ASSUMED SQUARE)
C
      NUMPTS = OUTSIZ * OUTSIZ
C
C     CALCULATE THE INCREMENTAL NUMBER OF LOOKS
C     TO BE USED IN INDEXING THE FILTERS
C
      DELTAN = (NMAX-NMIN) / FLOAT(FLTNUM)
C
C
C     OPEN FILES AND CHECK FOR ERRORS
C
      CALL OPN ( INFC, INFNM, 'OLD', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  1009
C
C
      CALL OPN ( OUTFC, OUTFNM, 'NEW', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  2009
C
      CALL OPN ( OUT2FC, OUT2FN, 'NEW', 'FOR', ERRNUM, ERR)
      IF (ERR) GOTO  3009
C
C
C
      WNDPTS = WNDSIZ * WNDSIZ
C
C
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C     GENERATE THE FILTERS
C
        CALL GENFLT (FLTRS, DIST, FLTNUM, WNDSIZ, DELTAN, NUMLKS, WNDPTS,
     &      ECURES, CUT2FC, TTYCUT)
C
C     INITIALIZE THE CIRCULAR QUEUE
C
        DO 40  REC=1,WNDSIZ
C
         READ (INFC,ERR=4009) (IN(WRD),WRD=1,SIZE)
           DO 30  WORD=1,SIZE
            QUEUE(REC,WORD) = IN(WORD)
30         CONTINUE
40      CONTINUE
C
C
C     BEGIN PROCESSING
C
        RECS = 0
        OREC = 1
C
50      TMPORC = OREC
C
        DO 90  START=1,OUTSIZ
C
C     GET THE LOCAL STATISTICS FOR THE AREA DEFINED BY THE WINDOW
C     AND FILL THE WINDOW ARRAY
C
        CALL LCSTAT (QUEUE, WINDOW, TMPORC, START, WNDSIZ, WNDPTS,
     &  SIZE, ZMEAN, VARZ, Z)
C
C
        EQTHRS = ZMEAN * ZMEAN / VARZ
C
C     CHECK TO SEE IF LOCAL NUMBER OF LOOKS IS OUT OF THE USER RANGE
C
        IF (EQTHRS .LT. NMIN .OR. EQTHRS .GT. NMAX)  RNGERR = .TRUE.
C
C
C
C     UPDATE RUNNING SUMS FOR DETERMINING THRESHOLD
C
        TTLTMN = TTLTMN + EQTHRS/NUMPTS
        TTLTSD = TTLTSD + EQTHRS*EQTHRS/NUMPTS
        THRSMI = MIN (THRSMI, EQTHRS)
        THRSMX = MAX (THRSMX, EQTHRS)
C
C
C     CALCULATE WHICH FILTER TO USE
C
        FLT = FLTNUM - INT((EQTHRS-NMIN) / DELTAN)
        IF (FLT .LT. 1) FLT = 1
        IF (FLT .GT. FLTNUM) FLT = FLTNUM
C
C     UPDATE HISTOGRAM
C
        FLTHST (FLT) = FLTHST (FLT)   + 1
C
C
C     PERFORM THE FILTERING
```

```
C
65     CALL  FILTER (WINDOW, FLTRS, WNDSIZ, FLTNUM, FLT,
      &   ZMEAN, VARZ, Z)
C
C
       XESTMT = ZMEAN
C
C    PUT THE FILTERED VALUE IN THE OUTPUT BUFFER
C
       IF (XESTMT .GT. 255.0)  XESTMT = 255.0
       OUT(START) = INT (XESTMT)
C
C
90        CONTINUE
C
C
C    WRITE THIS RECORD AND UPDATE THE RECORD COUNTER
C
C
       WRITE (OUTFC) (OUT(WRD),WRD=1,OUTSIZ)
C    DON'T BOTHER TO CHECK FOR A WRITE ERROR
C
       RECS = RECS + 1
C
C
C    READ A NEW RECORD INTO THE QUEUE AND UPDATE THE FRONT-END POINTER
C
        READ (INFC,END=200,ERR=4009) (IN(WRD),WRD=1,SIZE)
C
       DO 100  WORD=1,SIZE
         QUEUE(QREC,WORD) = IN(WORD)
100    CONTINUE
C
       QREC = MOD (QREC, WNDSIZ) + 1
C
       GOTO 50
C
C
C
C    WE'VE REACHED AN END-OF-FILE SO THE OUTPUT NUMBER OF
C    RECORDS WILL BE WNDSIZ-1 LESS THAN THE NUMBER INPUT
C    WRAP IT ALL UP AND QUIT
C
C
200    CALL UCLOSE (INFC)
       CALL UCLOSE (OUTFC)
C
       WRITE (TTYOUT,635) THRSMI,THRSMX,TTLTMN,TTLTSD-TTLTMN*TTLTMN
635    FORMAT (1X,'MINIMUM NUMBER OF LOOKS = ',F9.3,
      &  /1X,'MAXIMUM NUMBER OF LOOKS = ',F9.3,
      &  /1X,'MEAN NUMBER OF LOOKS = ',F9.3,
      &  /1X,'STANDARD DEVIATION OF NUMBER OF LOOKS = ',F9.3)
C
       IF (RNGERR) WRITE (TTYOUT,645)
645    FORMAT (1X,'* * * E R R O R -- ENCOUNTERED LOCAL NUMBER'
      & /5X,' OF LOOKS WHICH WERE OUTSIDE THE USER SPECIFIED RANGE.'
      & /5X,' THE FIRST OR LAST FILTERS WERE USED IN THESE AREAS')
C
C
C
```

```
      WRITE (CUT2FC, 650)
650   FCRMAT (/5X,'- - -   F I L T E R   U S A G E   - - [illegible]   ER #',
     &  5X,'% USAGE')
C
      CC 150  FLT = 1,FLTNUM
       WRITE (CUT2FC,700) FLT,FLCAT(FLTHST(FLT))*100.0/NUMPTS
700    FORMAT (1X,I3,6X,F6.3)
150   CCNTINUE
C
C
      GCTC 1010
C
C
C
1009  CALL  FILERR (TTYCUT, INFNM, ERRNUM)
      GCTC 1010
C
2009  CALL FILERR (TTYOUT, CUTFNM, ERRNUM)
      GCTC 1010
C
3009  CALL FILERR (TTYOUT, OUT2FN, ERRNUM)
      GCTC 1010
C
4009  WRITE (TTYCUT,660)
660   FCRMAT (1X,'* * * ERRCR IN REACING INPUT IMAGE * * *')
C
1010  RETURN
      END
ECF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C -----------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIE   S LAB
C -----------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C -----------------------------------------------------------------------------
C PROGRAM NAME:EADFLT            AUTHOR:J. SCOTT GARDNER DATE:02/27 83
C -----------------------------------------------------------------------------
C PURPOSE :    THIS IS THE MAINLINE FOR THE ADAPTIVE FILTER
C ROUTINE WHICH USES A NON-ISOTROPIC FILTER
C WINDOW WITH AN EXPONENTIAL WEIGHTING FUNCTION.
C
C
C
C -----------------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C -----------------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C -----------------------------------------------------------------------------
C                       NON-LOCAL VARIABLES
C -----------------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C -----------------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C     NAME          \          DESCRIPTION
C -----------------------------------------------------------------------------
C EADSUB            \DOES THE PROCESSING-- CALLED WITH VARIABLE PARAMETERS
C                   \
C                   \
C                   \
C                   \
C                   \
C                   \
C -----------------------------------------------------------------------------
      PROGRAM  EADFLT
C
      INTEGER  MAXSIZ, MXWSIZ, BUFCUE, BUFWND, MXFLTS, BUFFWN
      INTEGER  BUFMWN
C
C
C   BUFCUE = MAXSIZE * MAXWINDOWSIZE
C   BUFWND = MAXWINDOWSIZE * MAXWINDOWSIZE
C   BUFFWN IS THE BUFFER FOR THE FILTERS ARRAY
C
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      PARAMETER (MAXSIZ=512)
      PARAMETER (MXWSIZ=9)
      PARAMETER (MXFLTS=20)
      PARAMETER (BUFCUE=MAXSIZ*MXWSIZ)
      PARAMETER  (BUFWND=MXWSIZ*MXWSIZ)
      PARAMETER  (BUFMWN=8*MXWSIZ*MXWSIZ)
      PARAMETER  (BUFFWN=MXFLTS * MXWSIZ * MXWSIZ)
C
      INTEGER  RECS, FLTNUM, MSKWND(BUFMWN)
      INTEGER  SIZE, OUTSIZ, WNDSIZ, OUT(MAXSIZ)
      INTEGER  IN(MAXSIZ)
      INTEGER  FLTHST (MXFLTS)
C
      REAL  QUEUE(BUFCUE), WINDOW(BUFWND), FLTRS(BUFFWN)
      REAL  NUMLKS, NMAX, NMIN, DIST(BUFWND), EQURES
      REAL  THSHLD
C
      INTEGER*1  INFNM(18), OUTFNM(18), OUT2FN(18)
C
      INTEGER  TTYIN,TTYOUT,INFC,OUTFC, OUT2FC
C
      DATA  TTYIN, TTYOUT  /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER THE FILENAME FOR INPUT (MUST BE AN OLD FILE)')
      READ (TTYIN,510) INFNM
510   FORMAT (18A1)
      WRITE (TTYOUT,700)  INFNM
700   FORMAT (1X,18A1)
C
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER THE FILENAME FOR OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUTFNM
      WRITE (TTYOUT,700)  OUTFNM
C
      WRITE (TTYOUT,603)
603   FORMAT (1X,'FILENAME FOR FILTER OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUT2FN
      WRITE (TTYOUT,700)  OUT2FN
C
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE INPUT IMAGE ')
      READ (TTYIN,*)  SIZE
      WRITE (TTYOUT,710)  SIZE
710   FORMAT (1X,I5)
C
      IF (SIZE .LE. MAXSIZ) GOTO 12
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
C
12    WRITE (TTYOUT,625)
625   FORMAT (1X,'ENTER THE NUMBER OF LOOKS ')
      READ (TTYIN,*)  NUMLKS
      WRITE (TTYOUT,720) NUMLKS
720   FORMAT (1X,F10.3)
C
```

```
      WRITE (TTYCUT,637)
637   FCRMAT (1X,'ENTER THE ECUIVALENT RESCLUTION ')
      READ (TTYIN,*)  EQURES
      WRITE (TTYCUT,720)  EQURES
C
      WRITE (TTYCUT,626)
626   FCRMAT (1X,'ENTER THE EDGE THRESHCLD VALUE ')
      READ (TTYIN,*)  THSHLD
      WRITE (TTYCUT,720)  THSHLD
C
      WRITE (TTYCUT,645)
645   FCRMAT (1X,'ENTER THE MINIMUM NUMBER OF LOCKS ')
      READ (TTYIN,*)  NMIN
      WRITE (TTYCUT,720)  NMIN
C
      WRITE (TTYCUT,647)
647   FCRMAT (1X,'ENTER THE MAXIMUM NUMBER OF LOCKS ')
      READ (TTYIN,*)  NMAX
      WRITE (TTYCUT,720)  NMAX
C
      WRITE (TTYOUT,649)
649   FCRMAT (1X,'ENTER THE NUMBER CF FILTERS ')
      READ (TTYIN,*)  FLTNUM
      WRITE (TTYOUT,710)  FLTNUM
C .
C
      WRITE (TTYOUT,627)
627   FCRMAT (1X,'ENTER THE SIZE CF THE FILTER WINDOW',
     &  /10X,'(THIS PARAMETER MUST BE ODD)')
      READ (TTYIN, *)  WNDSIZ
      WRITE (TTYOUT,710)  WNDSIZ
C
      IF (MOD(WNDSIZ,2) .NE. 0) GOTO 20
      WNDSIZ = WNDSIZ + 1
      WRITE (TTYOUT,632) WNDSIZ
632   FCRMAT (1X,'THAT IS NOT AN ODD NUMBER. I WILL USE ',I2,' INSTEAD')
C
20    IF (WNDSIZ .LE. MXWSIZ) GOTO 13
       WRITE (TTYCUT,641) MXWSIZ
641    FORMAT (1X,'* * *  E R R O R  - - MAXIMUM WINDOW SIZE = ',I5)
      GOTO 1010
C
13    OUTSIZ = SIZE - WNDSIZ + 1
C
C
C
C   CALL SUBROUTINE TO DO THE WORK
C
      CALL EACSUB (INFNM, OUTFNM, OUT2FN, QUEUE, WINDOW, MSKWND, FLTRS,
     &  DIST, IN, OUT, FLTHST, SIZE, FLTNUM, RECS, WNDSIZ,
     &  OUTSIZ, NUMLKS, THSHLD, NMIN, NMAX, EQURES, TTYOUT)
C
C
C
      WRITE (TTYOUT,630)
630   FCRMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYCUT,640)  OUTSIZ, RECS
640   FCRMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
C
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      [illegible]C 1010
C
C
C
1010  STOP
      END
EOF..
?
```

```
C -------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C -------------------------------------------------------------------------
C  PROGRAM SUITE : NOISE FILTERS          REF. # :
C -------------------------------------------------------------------------
C  PROGRAM NAME:EADSUB          AUTHOR:J. SCOTT GARDNER DATE:3/4/83
C -------------------------------------------------------------------------
C  PURPOSE :    THIS IS THE SUBROUTINE TO PERFORM
C  THE ACTUAL PROCESSING FOR THE ADAPTIVE WEINER
C  NON-ISOTROPIC FILTER ROUTINE.
C
C
C
C -------------------------------------------------------------------------
C                        PARAMETER DEFINITION
C   NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C -------------------------------------------------------------------------
C  INFNM            \CH*18  \R      \        \INPUT FILENAME
C  OUTFNM           \CH*18  \R      \        \OUTPUT FILENAME
C  OUT2FN           \CH*18  \R      \        \FILTER OUTPUT FILENAME
C  QUEUE            \R      \R      \        \IMAGE DATA QUEUE
C  WINDOW           \R      \R/W    \        \LOCAL AREA WINDOW
C  MSKWND           \R      \R/W    \        \EDGE TEMPLATES
C  FLTRS            \R      \R/W    \        \FILTERS ARRAY
C  DIST             \R      \R/W    \        \DISTANCE ARRAY
C  IN               \I      \R/W    \        \INPUT BUFFER
C  OUT              \I      \R/W    \        \OUTPUT BUFFER
C  FLTHST           \I      \R      \        \FILTER USAGE HISTOGRAM
C  SIZE             \I      \R      \        \IMAGE SIZE
C  FLTNUM           \I      \R      \        \NUMBER OF FILTERS
C  RECS             \I      \W      \        \NUMBER OF RECORDS IN IMAGE
C  WNDSIZ           \I      \R      \        \SIZE OF LOCAL AREA WINDOW
C  OUTSIZ           \I      \R      \        \SIZE OF OUTPUT IMAGE
C  NUMLKS           \R      \R      \        \NUMBER OF LOOKS
C  THSHLD           \R      \R      \        \EDGE THRESHOLD
C  NMIN             \R      \R      \        \MINIMUM NUMBER OF LOOKS
C  NMAX             \R      \R      \        \MAXIMUM NUMBER OF LOOKS
C  EQURES           \R      \R      \        \EQUIVALENT RESOLUTION
C  TTYOUT           \I      \R      \        \OUTPUT TO TERMINAL FILECODE
C -------------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C -------------------------------------------------------------------------
C                   \       \       \        \
C -------------------------------------------------------------------------
C                        SUBROUTINES REQUIRED
C     NAME            \          DESCRIPTION
C -------------------------------------------------------------------------
C  GENFLT           \GENERATE THE FILTERS AND PRINT THEM
C  GNMSKS           \GENERATES MASKS
C  LCSTAT           \GETS LOCAL STATS AND FILLS A WINDOW ARRAY FROM QUEUE
C  SUBMSK           \GETS A 3X3 LOCAL MEAN FROM WINDOW
C  FNDEDG           \GETS EDGE ORIENTATION AND SELECTS A MASK
C  FILTER           \APPLY THE PROPER FILTER TO THE LOCAL AREA
C  EGSTAT           \GET THE EDGE STATISTICS
C  EDGFLT           \COMPUTES LOCAL STATS OUTSIDE THE EDGE
C  OPN              \OPEN FILE AND ASSIGN FILECODE
C  UCLOSE           \CLOSE FILES OPENED WITH OPN
C  FILERR           \REPORT TYPE OF FILE ERROR
C -------------------------------------------------------------------------
      SUBROUTINE  EADSUB (INFNM, OUTFNM, OUT2FN, QUEUE, WINDOW, MSKWND,
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
     &  FLTRS, [illegible] IN, OUT, FLTHST, SIZE, FLTNUM, RECS, WNDSIZ,
     &  OUTSIZ, [illegible]KS, THSHLD, NMIN, NMAX, EQURES, TTYOUT)
C
C
C
      INTEGER  SIZE, OUTSIZ, WNDSIZ
      INTEGER  IN(SIZE),  OUT(OUTSIZ), WRD
      INTEGER  MSK3X3(3,3,4), SBAREA(3,3), MSKWND(8,WNDSIZ,WNDSIZ)
C
      INTEGER  FLTNUM, FLT, ERRNUM
      INTEGER  FLTHST (FLTNUM)
      INTEGER  REC, WORD, RECS, CREC, TMPORC, START, IERR
      INTEGER  MSKNUM, I, J, K
C
      REAL  QUEUE(WNDSIZ,SIZE), WINDOW(WNDSIZ,WNDSIZ)
      REAL  FLTRS (FLTNUM, WNDSIZ, WNDSIZ), DIST(WNDSIZ,WNDSIZ)
      REAL  WNDPTS, Z, ZMEAN, VARZ, NUMLKS, THSHLD, XESTMT
      REAL  THRSMX, THRSMI, TTLTMN, TTLTSD, DELTAN, NMIN, NMAX
      REAL  EGTHRS, NUMPTS, EQURES, TTLHMN, NMHPTS
C
      INTEGER*1  INFNM(18), OUTFNM(18), OUT2FN(18)
C
      LOGICAL  ERR, RNGERR
C
      INTEGER  TTYOUT,INFC,OUTFC,OUT2FC
C
C
C
C    INITIALIZE THE 3X3 GRADIENT WINDOWS
C
C
      DATA  (((MSK3X3(I,J,K), I=1,3), J=1,3), K=1,4)
     &  /1,1,1,0,0,0,-1,-1,-1, 0,-1,-1,1,0,-1,1,1,0,
     &    1,0,-1,1,0,-1,1,0,-1, 1,1,0,1,0,-1,0,-1,-1/
C
C
C    INITIALIZE ZMEAN*ZMEAN/VARZ  MIN AND MAX AND TOTALS
C
      THRSMI = 100000
      THRSMX = -100000
      TTLTMN = 0.0
      TTLTSD = 0.0
      TTLHMN = 0.0
      NMHPTS = 0.0
C
      RNGERR = .FALSE.
C
C    ZERO HIST ARRAY
C
      DO 5 FLT=1,FLTNUM
       FLTHST(FLT) = 0
5     CONTINUE
C
C    DETERMINE THE NUMBER OF POINTS IN IMAGE (ASSUMED SQUARE)
C
      NUMPTS = OUTSIZ * OUTSIZ
C
C    CALCULATE THE INCREMENTAL NUMBER OF LOOKS
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      TO BE USED IN INDEXING THE FILTERS

        DELTAN = (NMAX-NMIN) / FLOAT(FLTNUM)

C     OPEN FILES AND CHECK FOR ERRORS
C
        CALL OPN ( INFC, INFNM, 'OLD', 'UNF', ERRNUM, ERR)
        IF (ERR) GOTO  1009
C

        CALL OPN ( OUTFC, OUTFNM, 'NEW', 'UNF', ERRNUM, ERR)
        IF (ERR) GOTO  2009

        CALL OPN ( OUT2FC, OUT2FN, 'NEW', 'FOR', ERRNUM, ERR)
        IF (ERR) GOTO  3009

      FIRST, GENERATE THE EDGE TEMPLATES
        CALL  GNMSKS (MSKWND, WNDSIZ)
C


        WNDPTS = WNDSIZ * WNDSIZ


C     GENERATE THE FILTERS
C
        CALL GENFLT (FLTRS, DIST, FLTNUM, WNDSIZ, DELTAN, NUMLKS, WNDPTS,
       &     EQURES, OUT2FC, TTYOUT)
C
      INITIALIZE THE CIRCULAR QUEUE

        DO 40  REC=1,WNDSIZ

         READ (INFC,END=4009,ERR=4009) (IN(WRD),WRD=1,SIZE)
           DO 30  WORD=1,SIZE
            QUEUE(REC,WORD)=IN(WORD)
 0         CONTINUE
 0      CONTINUE
C

      BEGIN PROCESSING
C
        RECS = 0
        QREC = 1

50      TMPQRC = QREC

        DO 90  START=1,OUTSIZ
C
      GET THE LOCAL STATISTICS FOR THE AREA DEFINED BY THE WINDOW
      AND FILL THE WINDOW ARRAY

        CALL LCSTAT (QUEUE, WINDOW, TMPQRC, START, WNDSIZ, WNDPTS,
       &  SIZE, ZMEAN, VARZ, Z)
C
        IF (VARZ .NE. 0.0) THEN
         EGTHRS = ZMEAN * ZMEAN / VARZ
        ELSE
```

```
       EGTHRS = NMAX
      END IF
C
C     CHECK TO SEE IF LOCAL NUMBER OF LOOKS IS OUT OF THE USER RANGE
C
      IF (EGTHRS .LT. (NMIN-.01*NUMLKS) .OR. EGTHRS .GT.
     &      (NMAX+.01*NUMLKS) ) RNGERR = .TRUE.
C
C
C
C     UPDATE RUNNING SUMS FOR DETERMINING THRESHOLD
C
      TTLTMN = TTLTMN + EGTHRS/NUMPTS
      TTLTSD = TTLTSD + EGTHRS*EGTHRS/NUMPTS
      THRSMI = MIN (THRSMI, EGTHRS)
      THRSMX = MAX (THRSMX, EGTHRS)
C
C
C     DETERMINE IF AN EDGE EXISTS BASED ON THE LOCAL
C     NUMBER OF LOOKS
C
      IF (EGTHRS .LE. NUMLKS - THSHLD)  GOTO 44
       TTLHMN = TTLHMN + EGTHRS
       NMHPTS = NMHPTS + 1.0
      GOTO 60
C
C
C     WE HAVE AN EDGE, SO PROCEED WITH THE EDGE FILTERING
C
C     GET THE 3X3 SUBAREA LOCAL MEAN
C
44    CALL  SUBMSK  (WINDOW, SBAREA, WNDSIZ)
C
C
C     FIND THE EDGE ORIENTATION AND DETERMINE WHICH EDGE
C     TEMPLATE TO USE IN CALCULATING OUR NEW LOCAL MEAN
C
      CALL  FNDEDG  (SBAREA, MSK3X3, MSKNUM)
C
C
C     COMPUTE A NEW LOCAL MEAN AND VARIANCE USING THE APPROPRIATE
C     TEMPLATE
C
      CALL EGSTAT  (WINDOW, MSKWND, WNDSIZ, MSKNUM,
     &  ZMEAN, VARZ, Z)
C
      IF (VARZ .NE. 0.0) THEN
       EGTHRS = ZMEAN * ZMEAN / VARZ
      ELSE
       EGTHRS = NMAX
      END IF
C
C
C
C     CALCULATE WHICH FILTERS TO USE
C
      FLT = INT((EGTHRS-NMIN) / DELTAN) + 1
      IF (FLT .LT. 1) FLT = 1
      IF (FLT .GT. FLTNUM) FLT = FLTNUM
      FLT = FLTNUM - FLT + 1
```

```
C
C
C     UPDATE HISTOGRAM
C
        FLTHST (FLT) = FLTHST (FLT)  + 1
C
C     PERFORM THE EDGE FILTERING
C
        CALL  EDGFLT (WINDOW, FLTRS, MSKWND, WNDSIZ, FLTNUM,
     &    FLT, MSKNUM, ZMEAN, VARZ, Z)
C
C
        GOTO 65
C
C
C     THIS AREA IS HOMOGENEOUS
C     CALCULATE WHICH FILTER TO USE
C
60      FLT = INT((EGTHRS-NMIN) / DELTAN) + 1
        IF (FLT .LT. 1) FLT = 1
        IF (FLT .GT. FLTNUM) FLT = FLTNUM
        FLT = FLTNUM - FLT + 1
C
C     UPDATE HISTOGRAM
C
        FLTHST (FLT) = FLTHST (FLT)  + 1
C
C
C     PERFORM THE FILTERING
C
        CALL  FILTER (WINDOW, FLTRS, WNDSIZ, FLTNUM, FLT,
     &    ZMEAN, VARZ, Z)
C
C
65      XESTMT = ZMEAN
C
C     PUT THE FILTERED VALUE IN THE OUTPUT BUFFER
C
        IF (XESTMT .GT. 255.0)  XESTMT = 255.0
        OUT(START) = INT (XESTMT)
C
C
90         CONTINUE
C
C
C     WRITE THIS RECORD AND UPDATE THE RECORD COUNTER
C
C
        WRITE (OUTFC) (OUT(WRD),WRD=1,OUTSIZ)
C
C
        RECS = RECS + 1
C
C
C     READ A NEW RECORD INTO THE QUEUE AND UPDATE THE FRONT-END POINTER
C
          READ (INFC,END=200,ERR=4009) (IN(WRD),WRD=1,SIZE)
           DO 110  WORD=1,SIZE
            QUEUE(QREC,WORD) = IN(WORD)
110        CONTINUE
```

```
C
      QREC = MOD (QREC, WNDSIZ) + 1
C
      GOTO 50
C
C
C
C    WE'VE REACHED AN END-OF-FILE SO THE OUTPUT NUMBER OF
C    RECORDS WILL BE WNDSIZ-1 LESS THAN THE NUMBER INPUT
C    WRAP IT ALL UP AND QUIT
C
C
200   CALL UCLOSE (INFC)
      CALL UCLOSE (OUTFC)
C
      WRITE (TTYOUT,635) THRSMI,THRSMX,TTLTMN,TTLTSD-TTLTMN*TTLTMN
635   FORMAT (1X,'MINIMUM NUMBER OF LOOKS = ',F9.3,
     & /1X,'MAXIMUM NUMBER OF LOOKS = ',F9.3,
     & /1X,'MEAN NUMBER OF LOOKS = ',F9.3,
     & /1X,'STANDARD DEVIATION OF NUMBER OF LOOKS = ',F9.3)
C
C
      WRITE (TTYOUT,637) TTLHMN/NMHPTS
637   FORMAT (1X,'HOMOGENEOUS AREA MEAN = ',F9.3)
      WRITE (TTYOUT,638) NMHPTS*100.0/NUMPTS
638   FORMAT (1X,F9.3,' PERCENT OF THE IMAGE WAS HOMOGENEOUS')
C
      IF (RNGERR) WRITE (TTYOUT,645)
645   FORMAT (1X,'* * * E R R O R -- ENCOUNTERED LOCAL NUMBER'
     & /5X,' OF LOOKS WHICH WERE OUTSIDE THE USER SPECIFIED RANGE.'
     & /5X,' THE FIRST OR LAST FILTERS WERE USED IN THESE AREAS')
C
C
C
      WRITE (OUT2FC, 650)
650   FORMAT (/5X,'- - -  F I L T E R  U S A G E  - - -',/1X,'FILTER #',
     & 5X,'% USAGE')
C
      DO 150  FLT = 1,FLTNUM
       WRITE (OUT2FC,700) FLT,FLOAT(FLTHST(FLT))*100.0/NUMPTS
700    FORMAT (1X,I3,6X,F6.3)
150   CONTINUE
C
C
      GOTO 1010
C
C
C
1009  CALL  FILERR (TTYOUT, INFNM, ERRNUM)
      GOTO 1010
C
2009  CALL FILERR (TTYOUT, OUTFNM, ERRNUM)
      GOTO 1010
C
3009  CALL FILERR (TTYOUT, OUT2FN, ERRNUM)
      GOTO 1010
C
4009  WRITE (TTYOUT,660)
660   FORMAT (1X,'* * * ERROR IN READING INPUT IMAGE * * *')
C
```

```
1010  RETURN
      END
$CF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
C  ------------------------------------------------------------------------
C   UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C  ------------------------------------------------------------------------
C  PROGRAM SUITE : NOISE FILTERS        REF. # :
C  ------------------------------------------------------------------------
C  PROGRAM NAME:EDGFLT           AUTHOR:J. SCOTT GARDNER DATE:02/14/83
C  ------------------------------------------------------------------------
C  PURPOSE :   THIS SUBROUTINE COMPUTES THE LOCAL STATISTICS FOR
C  A WINDOW AFTER APPLYING A FILTER WHICH MUST
C  PREVIOUSLY HAVE BEEN NORMALIZED.
C
C
C
C  ------------------------------------------------------------------------
C                       PARAMETER DEFINITION
C   NAME             \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C  ------------------------------------------------------------------------
C  WINDOW            \R      \R      \       \LOCAL AREA WINDOW
C  FLTRS             \R      \R      \       \FILTERS ARRAY
C  MSKWND            \I      \R      \       \EDGE TEMPLATES
C  WNDSIZ            \I      \R      \       \SIZE OF WINDOW
C  NUMFLT            \I      \R      \       \NUMBER OF FILTERS
C  FLT               \I      \R      \       \NUMBER OF FILTER  TO USE
C  MSKNUM            \I      \R      \       \NUMBER OF MASK TO USE
C  ZMEAN             \R      \W      \       \LOCAL MEAN
C  VARZ              \R      \W      \       \LOCAL VARIANCE
C  Z                 \R      \W      \       \VALUE OF CENTER PIXEL
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C  ------------------------------------------------------------------------
C                       NON-LOCAL VARIABLES
C  ------------------------------------------------------------------------
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C                    \       \       \       \
C  ------------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C     NAME           \          DESCRIPTION
C  ------------------------------------------------------------------------
C                    \
C                    \
C                    \
C                    \
C  ------------------------------------------------------------------------
      SUBROUTINE  EDGFLT (WINDOW, FLTRS, MSKWND, WNDSIZ,
     &  NUMFLT, FLT, MSKNUM, ZMEAN, VARZ, Z)
C
      INTEGER  WNDSIZ, FLT, REC, WORD, NUMFLT
      INTEGER  MSKWND (8,WNDSIZ,WNDSIZ), MSKNUM, FLTNDX
C
      REAL  FLTRS(NUMFLT,WNDSIZ,WNDSIZ)
      REAL  WINDOW(WNDSIZ,WNDSIZ)
      REAL  ZMEAN, VARZ, Z, TOTALM, TOTALV, FLTVAL
```

```
C
C
      TCTALM = 0.0
      TCTALV = 0.0
C
      DC 20  REC = 1,WNDSIZ
       DC 10  WCRD = 1,WNDSIZ
        IF(MSKWND(MSKNUM,REC,WORD).EQ.0) THEN
         FLTNDX = NUMFLT - FLT + 1
        ELSE
         FLTNDX = FLT
        END IF
        FLTVAL = WINDOW(REC,WCRD) * FLTRS (FLTNDX,REC,WORD)
        TCTALM = TCTALM + FLTVAL
        TCTALV = TCTALV + FLTVAL * FLTVAL
10     CONTINUE
20    CCNTINUE
C
      ZMEAN = TOTALM
      VARZ = TCTALV  - ZMEAN*ZMEAN
C
      RETURN
      END
EOF..
?
```

```
C ----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS         REF. # :
C ----------------------------------------------------------------------
C PROGRAM NAME:SIGFLT           AUTHOR:J. SCOTT GARDNER DATE:04/17/83
C ----------------------------------------------------------------------
C PURPOSE :    THIS ROUTINE PERFORMS LEE'S SIGMA FILTER.
C
C
C
C
C
C ----------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ----------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ----------------------------------------------------------------------
C                       NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ----------------------------------------------------------------------
C                      SUBROUTINES REQUIRED
C    NAME           \          DESCRIPTION
C ----------------------------------------------------------------------
C SIGSUB           \DOES THE PROCESSING -- CALLED WITH VARIABLE PARAMETERS
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C ----------------------------------------------------------------------
      PROGRAM  SIGFLT
C
C
      INTEGER  MAXSIZ, MXWSIZ, BUFQUE
C
C
C   BUFQUE = MAXSIZE * MAXWINDOWSIZE
C
      PARAMETER (MAXSIZ=512)
      PARAMETER (MXWSIZ=15)
```

```
      PARAMETER (BUFQUE=MAXSIZ*MXWSIZ)
C
      INTEGER  RECS
      INTEGER  IN(MAXSIZ), OUT(MAXSIZ), SIZE, OUTSIZ, WNDSIZ
C
      REAL  QUEUE(BUFQUE)
      REAL  NUMLKS, K
C
      INTEGER*1  INFNM(18), OUTFNM(18)
C
      INTEGER  TTYIN,TTYOUT,INFC,OUTFC
C
      DATA  TTYIN, TTYOUT  /15, 16/
C
C
      WRITE (TTYOUT,601)
601   FORMAT (1X,'ENTER THE FILENAME FOR INPUT (MUST BE AN OLD FILE)')
      READ (TTYIN,510) INFNM
510   FORMAT (18A1)
      WRITE (TTYOUT,700) INFNM
700   FORMAT (18A1)
C
C
      WRITE (TTYOUT,602)
602   FORMAT (1X,'ENTER THE FILENAME FOR OUTPUT (MUST BE A NEW FILE)')
      READ (TTYIN,510) OUTFNM
      WRITE (TTYOUT,700)  OUTFNM
C
C
C
      WRITE (TTYOUT,610)
610   FORMAT (1X,'ENTER THE SIZE OF THE INPUT IMAGE ')
      READ (TTYIN,*)  SIZE
      WRITE (TTYOUT,710)  SIZE
710   FORMAT (1X,I5)
C
      IF (SIZE .LE. MAXSIZ) GOTO 12
       WRITE (TTYOUT,615) MAXSIZ
615    FORMAT (1X,'* * * E R R O R - - THE MAXIMUM SIZE = ',I5)
       GOTO 1010
C
12    WRITE (TTYOUT,625)
625   FORMAT (1X,'ENTER THE NUMBER OF LOOKS ')
      READ (TTYIN,*)  NUMLKS
      WRITE (TTYOUT,720)  NUMLKS
720   FORMAT (1X,F10.3)
C
C
      WRITE (TTYOUT,628)
628   FORMAT (1X,'ENTER THE SIGMA THRESHOLD ')
      READ (TTYIN,*)  K
      WRITE (TTYOUT,720)  K
C
C
      WRITE (TTYOUT,627)
627   FORMAT (1X,'ENTER THE SIZE OF THE FILTER WINDOW',
     & /10X,'(THIS PARAMETER MUST BE ODD)')
      READ (TTYIN, *)  WNDSIZ
      WRITE (TTYOUT,710)  WNDSIZ
C
```

ORIGINAL PAGE IS OF POOR QUALITY

```
      IF (MOD(WNDSIZ,2) .NE. 0) GOTO 20
      WNDSIZ = WNDSIZ + 1
      WRITE (TTYOUT,632) WNDSIZ
632   FORMAT (1X,'THAT IS NOT AN ODD NUMBER. I WILL USE ',I2,' INSTEAD')
C
20    IF (WNDSIZ .LE. MXWSIZ) GOTO 13
       WRITE (TTYOUT,641) MXWSIZ
641    FORMAT (1X,'* * *  E R R O R  - - MAXIMUM WINDOW SIZE = ',I5)
      GOTO 1010
C
13    OUTSIZ = SIZE - WNDSIZ + 1
C
C
C
C   CALL SUBROUTINE TO DO THE WORK
C
      CALL SIGSUB (INFNM, OUTFNM, QUEUE, IN, OUT,
     &   SIZE, RECS, WNDSIZ, OUTSIZ, NUMLKS, K, TTYOUT)
C
C
C
      WRITE (TTYOUT,630)
630   FORMAT (1X,'* * *  A L L  D O N E  * * *')
      WRITE (TTYOUT,640)  OUTSIZ, RECS
640   FORMAT (1X,'THE OUTPUT IMAGE IS ',I5,' WORDS BY ',I5,' RECORDS')
C
C
      GOTO 1010
C
C
C
1010  STOP
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C ---------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ---------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ---------------------------------------------------------------------
C PROGRAM NAME:SIGSUB          AUTHOR:J. SCOTT GARDNER DATE:3/4/83
C ---------------------------------------------------------------------
C PURPOSE :    THIS IS THE SUBROUTINE TO PERFORM
C THE ACTUAL PROCESSING FOR LEE'S SIGMA FILTER.
C
C
C
C
C ---------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME           \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ---------------------------------------------------------------------
C INFNM          \CH*13  \R      \        \INPUT FILENAME
C OUTFNM         \CH*13  \R      \        \OUTPUT FILENAME
C QUEUE          \R      \R      \        \IMAGE DATA QUEUE
C IN             \I      \W      \        \INPUT BUFFER
C OUT            \I      \W      \        \OUTPUT BUFFER
C SIZE           \I      \R      \        \IMAGE SIZE
C RECS           \I      \W      \        \NUMBER OF RECORDS IN IMAGE
C WNDSIZ         \I      \R      \        \SIZE OF LOCAL AREA WINDOW
C OUTSIZ         \I      \R      \        \SIZE OF OUTPUT IMAGE
C NUMLKS         \R      \R      \        \NUMBER OF LOOKS
C K              \R      \R      \        \SIGMA THRESHOLD
C TTYOUT         \I      \R      \        \OUTPUT TO TERMINAL FILECODE
C                \       \       \        \
C                \       \       \        \
C                \       \       \        \
C                \       \       \        \
C                \       \       \        \
C ---------------------------------------------------------------------
C                      NON-LOCAL VARIABLES
C ---------------------------------------------------------------------
C                \       \       \        \
C ---------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME          \          DESCRIPTION
C ---------------------------------------------------------------------
C OPN            \OPEN FILE AND ASSIGN FILECODE
C UCLOSE         \CLOSE FILES OPENED WITH OPN
C FILERR         \REPORT TYPE OF FILE ERROR
C ---------------------------------------------------------------------
      SUBROUTINE  SIGSUB (INFNM, OUTFNM, QUEUE,  IN, OUT,
     &   SIZE, RECS, WNDSIZ, OUTSIZ, NUMLKS, K, TTYOUT)
C
C
C
      INTEGER  SIZE, OUTSIZ, WNDSIZ
C
      INTEGER   ERRNUM
      INTEGER  IN(SIZE), OUT(OUTSIZ)
      INTEGER  REC, WORD, RECS, OREC, MIDREC, TMPORC, START, IERR
      INTEGER  OWORD, MIDWRD, WRD, WREC, NXTORC, INDEX
C
      REAL  QUEUE(WNDSIZ,SIZE)
      REAL  NUMLKS, XESTMT, K, USRCNT
```

```
      REAL  LWRLIM(6), UPRLIM(6), LOWER, UPPER, LWR, UPR
      REAL  VALUE, SUBPTS, TOTALM, CENTER, USRCNT
C
      INTEGER*1  INFNM(18), OUTFNM(18)
C
      LOGICAL  ERR
C
      INTEGER  TTYOUT,INFC,OUTFC
C
C
C    INITIALIZE UPPER AND LOWER LIMIT ARRAYS
C
      DATA  LWRLIM  /.025,.121,.207,.273,.367,.48/
      DATA  UPRLIM  /3.69,2.785,2.408,2.191,1.945,1.709/
C
C    THE FOLLOWING DATA STATEMENTS ARE FOR A GAUSSIAN
C     ASSUMPTION
C
C      DATA  LWRLIM /0.0,0.0,0.0,0.0,.18,.37/
C      DATA  UPRLIM /3.0,2.41,2.15,2.0,1.82,1.63/
C
      INDEX = NUMLKS + 0.5
      IF (INDEX .GT. 4)  THEN
       IF (INDEX .LE. 8)  THEN
        INDEX = 5
       ELSE
        INDEX = 6
       END IF
      END IF
C
      LOWER = LWRLIM (INDEX)
      UPPER = UPRLIM (INDEX)
C
      USRCNT = 0.0
C
C
C    OPEN FILES AND CHECK FOR ERRORS
C
      CALL OPN ( INFC, INFNM, 'OLD', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  1009
C
C
      CALL OPN ( OUTFC, OUTFNM, 'NEW', 'UNF', ERRNUM, ERR)
      IF (ERR) GOTO  2009
C
C
C
C
C
C
C    INITIALIZE THE CIRCULAR QUEUE
C
      DO 40  REC=1,WNDSIZ
C
       READ (INFC,ERR=4009) (IN(WRD),WRD=1,SIZE)
         DO 30  WORD=1,SIZE
          QUEUE(REC,WORD) = IN(WORD)
30       CONTINUE
40    CONTINUE
C
```

```
C
C     BEGIN PROCESSING
C
         RECD = 0
         CREC = 1
         MIDREC = WNDSIZ/2 + 1
C
50       TMPCRC = CREC
C
         DO 90  START=1,OUTSIZ
C
C
         TOTALM = 0.0
         SUBPTS = 0.0
         MIDWRD=START+WNDSIZ/2
C
         CENTER = QUEUE (MIDREC,MIDWRD)
         LWR = LOWER * CENTER
         UPR = UPPER * CENTER
C
         DO 80  WREC = 1,WNDSIZ
          DO 70  CWCRD = START,START+WNDSIZ-1
C
           VALUE = QUEUE(TMPQRC,CWCRD)
           IF (VALUE.GE.LWR .AND. VALUE.LE.UPR)  THEN
            TOTALM = TOTALM + VALUE
            SUBPTS = SUBPTS + 1
           END IF
70          CONTINUE
C
         TMPCRC = MOD (TMPQRC,WNDSIZ) + 1
80       CONTINUE
C
C
         IF (SUBPTS .LE. K)  THEN
          TOTALM = CENTER + QUEUE (MIDREC,MIDWRD+1)
          NXTQRC = MOD (MIDREC,WNDSIZ) + 1
          TOTALM = TOTALM + QUEUE(NXTQRC,MIDWRD) + QUEUE(NXTQRC,MIDWRD+1)
          XESTMT = TOTALM / 4.0
          USECNT = USECNT + 1.0
         ELSE
          XESTMT = TOTALM / SUBPTS
         END IF
C
C
C
C     PUT THE FILTERED VALUE IN THE OUTPUT BUFFER
C
         IF (XESTMT .GT. 255.0)  XESTMT = 255.0
         OUT(START) = INT (XESTMT)
C
C
90          CONTINUE
C
C
C     WRITE THIS RECORD AND UPDATE THE RECORD COUNTER
C
         WRITE (OUTFC) (OUT(WRD),WRD=1,OUTSIZ)
C
C     DON'T BOTHER TO CHECK FOR A WRITE ERROR
```

```
      RECS = RECS + 1
C
C
C    READ A NEW RECORD INTO THE QUEUE AND UPDATE THE FRONT-END POINTER
C
      READ (INFC,END=200,ERR=4009) (IN(WRD),WRD=1,SIZE)
C
      DO 100  WORD=1,SIZE
        QUEUE(QREC,WORD) = IN(WORD)
100   CONTINUE
C
      QREC = MOD (QREC, WNDSIZ) + 1
      MIDREC = MOD (MIDREC, WNDSIZ) + 1
C
      GOTO 50
C
C
C
C    WE'VE REACHED AN END-OF-FILE SO THE OUTPUT NUMBER OF
C    RECORDS WILL BE WNDSIZ-1 LESS THAN THE NUMBER INPUT
C    WRAP IT ALL UP AND QUIT
C
C
200   CALL UCLOSE (INFC)
      CALL UCLOSE (OUTFC)
C
C
C
C
      WRITE (TTYOUT, 650) 100.0 * USECNT / FLOAT(OUTSIZ*OUTSIZ)
650   FORMAT (/5X,'THE SIGMA THRESHOLD WAS APPLIED TO ',F6.2,
     &  /1X,'PERCENT OF THE IMAGE')
C
C
      GOTO 1010
C
C
C
1009  CALL  FILERR (TTYOUT, INFNM, ERRNUM)
      GOTO 1010
C
2009  CALL FILERR (TTYOUT, OUTFNM, ERRNUM)
      GOTO 1010
C
C
4009  WRITE (TTYOUT,660)
660   FORMAT (1X,'* * * ERROR IN READING INPUT IMAGE * * *')
C
1010  RETURN
      END
EOF..
```

```
C ----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ----------------------------------------------------------------------
C  PROGRAM SUITE : NOISE FILTERS          REF. # :
C ----------------------------------------------------------------------
C  PROGRAM NAME:GNMSKS          AUTHOR:J. SCOTT GARDNER DATE:02/13/83
C ----------------------------------------------------------------------
C  PURPOSE :    THIS IS A SUBROUTINE TO GENERATE THE
C   EDGE TEMPLATES FOR AN ADAPTIVE EDGE FILTER.
C
C
C
C
C ----------------------------------------------------------------------
C                       PARAMETER DEFINITION
C   NAME           \ TYPE  \ CLASS\ RANGE  \      DESCRIPTION
C ----------------------------------------------------------------------
C  MSK3X3          \I       \W      \         \ARRAY OF 3X3 MASKS
C  MSKWND          \I       \W      \         \ARRAY OF EDGE TEMPLATES
C  WNDSIZ          \I       \R      \         \SIZE OF SQUARE WINDOWS
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C ----------------------------------------------------------------------
C                       NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C                  \        \       \         \
C ----------------------------------------------------------------------
C                      SUBROUTINES REQUIRED
C     NAME         \           DESCRIPTION
C ----------------------------------------------------------------------
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C ----------------------------------------------------------------------
      SUBROUTINE  GNMSKS  (MSKWND, WNDSIZ)
C
      INTEGER  WNDSIZ, LINE, CLM, START
      INTEGER  MSKWND(8,WNDSIZ,WNDSIZ)
C
C
C
C
C    FILL THE EDGE TEMPLATES
C
```

```
      START = 1
      DC 200  LINE=1,WNDSIZ
        DC 190  CLM=1,WNDSIZ
C
          IF(CLM .LT. WNDSIZ/2+1)  GCTC 10
          MSKWND(1,LINE,CLM) = 1
          GCTC 15
10        MSKWND(1,LINE,CLM)=0
C
15        IF (CLM .LE. WNDSIZ/2+1)  GCTC 20
           MSKWND(5,LINE,CLM) = 0
           GCTC 25
20        MSKWND(5,LINE,CLM) = 1
C
25        IF (CLM .GE. START)  GOTC 30
           MSKWND(2,LINE,CLM) = 0
           GCTC 35
30        MSKWND(2,LINE,CLM) = 1
C
35        IF (CLM .GT. START)  GOTO 40
           MSKWND(6,LINE,CLM) = 1
           GCTC 45
40        MSKWND(6,LINE,CLM) = 0
C
45        IF (LINE .LE. WNDSIZ/2+1)  GCTC 50
           MSKWND(3,LINE,CLM) = 0
           GCTC 55
50        MSKWND(3,LINE,CLM) = 1
C
55        IF (LINE .LE. WNDSIZ/2)  GOTC 60
           MSKWND(7,LINE,CLM) = 1
           GCTC 65
60        MSKWND(7,LINE,CLM) = 0
C
65        IF (CLM .LE. WNDSIZ-START+1)  GCTC 70
           MSKWND(4,LINE,CLM) = 0
           GCTC 75
70        MSKWND(4,LINE,CLM) = 1
C
75        IF (CLM .LE. WNDSIZ-START)  GOTC 80
           MSKWND(8,LINE,CLM) = 1
           GCTO 190
80        MSKWND(8,LINE,CLM) = 0
C
190     CONTINUE
        START = START + 1
200   CCNTINUE
C
C
      RETURN
      END
EOF..
?
```

```
C ---------------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ---------------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ---------------------------------------------------------------------------
C PROGRAM NAME:LCSTAT           AUTHOR:J. SCOTT GARDNER DATE:02/13/83
C ---------------------------------------------------------------------------
C PURPOSE :    THIS SUBROUTINE COMPUTES THE LOCAL STATISTICS FOR A
C WINDOW IN A CIRCULAR QUEUE AND FILLS A WINDOW ARRAY
C FOR LATER PROCESSING.
C
C
C
C ---------------------------------------------------------------------------
C                     PARAMETER DEFINITION
C  NAME           \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ---------------------------------------------------------------------------
C QUEUE          \R      \R      \        \DATA QUEUE
C WINDOW         \R      \W      \        \WINDOW FILLED FROM QUEUE
C TMPORC         \I      \R/W    \        \FRONT OF QUEUE
C START          \I      \R      \        \START OF WINDOW IN QUEUE
C WNDSIZ         \I      \R      \        \SIZE OF WINDOW
C WNDPTS         \R      \R      \        \NUMBER OF POINTS IN WINDOW
C SIZE           \I      \R      \        \SIZE OF QUEUE
C ZMEAN          \R      \W      \        \LOCAL MEAN
C VARZ           \R      \W      \        \LOCAL VARIANCE
C Z              \R      \W      \        \VALUE OF CENTER PIXEL
C                \       \       \        \
C                \       \       \        \
C                \       \       \        \
C                \       \       \        \
C ---------------------------------------------------------------------------
C                     NON-LOCAL VARIABLES
C ---------------------------------------------------------------------------
C                \       \       \        \
C                \       \       \        \
C                \       \       \        \
C                \       \       \        \
C ---------------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME          \        DESCRIPTION
C ---------------------------------------------------------------------------
C               \
C               \
C               \
C               \
C               \
C               \
C               \
C ---------------------------------------------------------------------------
      SUBROUTINE  LCSTAT  (QUEUE, WINDOW, TMPORC, START, WNDSIZ,
     &  WNDPTS, SIZE, ZMEAN, VARZ, Z)
C
      INTEGER  WREC, WWORD, OWORD, START, WNDSIZ, SIZE, TMPORC
C
      REAL  QUEUE(WNDSIZ,SIZE), WNDPTS, ZMEAN, VARZ, Z
      REAL  TOTALM, TOTALV, WINDOW(WNDSIZ,WNDSIZ)
C
C
      TOTALM = 0.0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      TOTALV = 0.0
C
      DO 20  WREC=1,WNDSIZ
       WWORD = 1
       DO 10  OWORD=START, START+WNDSIZ-1
        TOTALM = TOTALM + QUEUE(TMPORC,OWORD)
        TOTALV = TOTALV + QUEUE(TMPORC,OWORD)**2
        WINDOW(WREC,WWORD) = QUEUE(TMPORC,OWORD)
       WWORD = WWORD + 1
10     CONTINUE
C
C
C
C
      TMPORC = MOD (TMPORC, WNDSIZ) + 1
20    CONTINUE
C
C
      ZMEAN = TOTALM / WNDPTS
      VARZ = TOTALV / WNDPTS  - ZMEAN*ZMEAN
C
C    FIND THE CENTER PIXEL
C
      Z = WINDOW(WNDSIZ/2+1, WNDSIZ/2+1)
C
      RETURN
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C ----------------------------------------------------------------------
C                      UNIVERSITY OF KANSAS REMOTE SENSING LAB
C ----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ----------------------------------------------------------------------
C PROGRAM NAME:SUBMSK            AUTHOR:J. SCOTT GARDNER DATE:02/13/83
C ----------------------------------------------------------------------
C PURPOSE :    THIS IS A SUBROUTINE TO GENERATE A 3X3 WINDOW OF LOCAL
C MEANS FROM A WINDOW OF DATA POINTS.
C
C
C
C
C ----------------------------------------------------------------------
C                         PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \      DESCRIPTION
C ----------------------------------------------------------------------
C WINDOW           \R      \R      \         \WINDOW OF AREA OF INTEREST
C SBAREA           \I      \W      \         \3X3 LOCAL MEANS
C WNDSIZ           \I      \R      \         \SIZE OF WINDOW
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C ----------------------------------------------------------------------
C                         NON-LOCAL VARIABLES
C ----------------------------------------------------------------------
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C                  \       \       \         \
C ----------------------------------------------------------------------
C                        SUBROUTINES REQUIRED
C    NAME          \             DESCRIPTION
C ----------------------------------------------------------------------
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C                  \
C ----------------------------------------------------------------------
      SUBROUTINE  SUBMSK (WINDOW, SBAREA, WNDSIZ)
C
      INTEGER  WNDSIZ, SBWSIZ, OVRLAP, SBAREA(3,3)
      INTEGER  STARTR, STARTW, MREC, MWORD, REC, WORD
C
      REAL  WINDOW(WNDSIZ,WNDSIZ)
      REAL NUMPTS, TOTAL
C
C
C   CALCULATE THE SUBAREA SIZE AND OVRLAP SIZE
```

```
C
      CVRLAP = MOD (WNDSIZ, 3)
      SBWSIZ = WNDSIZ/3 + OVRLAP
C
      NUMPTS = SBWSIZ * SBWSIZ
      STARTR = 1
      STARTW = 1
C
      DO 40  MREC = 1,3
       DO 30  MWORD = 1,3
        TOTAL = 0.0
        DO 20  REC = STARTR, STARTR+SBWSIZ-1
         DO 10  WORD = STARTW, STARTW+SBWSIZ-1
          TOTAL = TOTAL + WINDOW(REC,WORD)
10       CONTINUE
20      CONTINUE
C
        SBAREA(MREC,MWORD) = TOTAL / NUMPTS
        STARTW = STARTW + SBWSIZ - OVRLAP
30     CONTINUE
       STARTW = 1
       STARTR = STARTR + SBWSIZ - OVRLAP
40    CONTINUE
C
C
      RETURN
      END
EOF..
?
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C -----------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C -----------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS         REF. # :
C -----------------------------------------------------------------------
C PROGRAM NAME:FNCEDG         AUTHOR:J. SCOTT GARDNER DATE:02/14/83
C -----------------------------------------------------------------------
C PURPOSE :    THIS SUBROUTINE USES A 3X3 GRADIENT MASK TO
C DETERMINE THE EDGE ORIENTATION AND THEN THRESHOLDS
C TO FIND WHICH SIDE OF THE EDGE THE CENTER PIXEL
C LIES. FROM THIS A MASK NUMBER IS SELECTED FOR SUBSEQUENT
C EDGE FILTERING.
C
C -----------------------------------------------------------------------
C                        PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C -----------------------------------------------------------------------
C SBAREA           \I      \R      \        \3X3 SUBAREA MEANS
C MSK3X3           \I      \R      \        \GRADIENT MASKS
C MSKNUM           \I      \W      \        \NUMBER OF MASK TO USE
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C -----------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C -----------------------------------------------------------------------
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C                  \       \       \        \
C -----------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C    NAME           \          DESCRIPTION
C -----------------------------------------------------------------------
C                   \
C                   \
C                   \
C                   \
C                   \
C                   \
C                   \
C -----------------------------------------------------------------------
      SUBROUTINE  FNCEDG  (SBAREA, MSK3X3, MSKNUM)
C
      INTEGER   SBAREA(3,3), MSK3X3(3,3,4), MSKNUM, MAXVAL
      INTEGER   CENTER, MSKNDX, ROW, CLM, SUM
      INTEGER   THRSH1, THRSH2
C
C
      CENTER = SBAREA(2,2)
      MAXVAL = -10000
C
```

```
C
      DO 50  MSKNDX = 1,4
      SUM = 0
       DO 20  ROW = 1,3
        DO 10  CLM = 1,3
         SUM = SUM + SBAREA(ROW,CLM) * MSK3X3(ROW,CLM,MSKNDX)
10      CONTINUE
20     CONTINUE
       IF (ABS(SUM) .LT. MAXVAL)  GOTO 50
        MAXVAL = ABS(SUM)
        MSKNUM = MSKNDX
50    CONTINUE
C
C
      IF (MSKNUM .NE. 1)  GOTO 60
       THRSH1 = SBAREA(2,3) - CENTER
       THRSH2 = SBAREA(2,1) - CENTER
       GOTO 90
C
60    IF (MSKNUM .NE. 2)  GOTO 70
       THRSH1 = SBAREA(1,3) - CENTER
       THRSH2 = SBAREA(3,1) - CENTER
       GOTO 90
C
70    IF (MSKNUM .NE. 3)  GOTO 80
       THRSH1 = SBAREA(1,2) - CENTER
       THRSH2 = SBAREA(3,2) - CENTER
       GOTO 90
C
80    THRSH1 = SBAREA(1,1) - CENTER
      THRSH2 = SBAREA(3,3) - CENTER
C
C
90    IF (ABS(THRSH1) .GT. ABS(THRSH2))  MSKNUM = MSKNUM + 4
C
      RETURN
      END
EOF..
?
```

```
C ---------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ---------------------------------------------------------------------
C  PROGRAM SUITE : NOISE FILTERS        REF. # :
C ---------------------------------------------------------------------
C  PROGRAM NAME:EGSTAT          AUTHOR:J. SCOTT GARDNER DATE:02/14/83
C ---------------------------------------------------------------------
C  PURPOSE :   THIS SUBROUTINE COMPUTES THE LOCAL STATISTICS FOR
C  A EDGE WINDOW AFTER APPLYING A MASK WHICH MUST
C  PREVIOUSLY HAVE BEEN NORMALIZED.
C
C
C
C ---------------------------------------------------------------------
C                     PARAMETER DEFINITION
C  NAME          \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ---------------------------------------------------------------------
C WINDOW         \R       \R      \          \LOCAL AREA WINDOW
C MSKWND         \R       \R      \          \EDGE TEMPLATES
C WNDSIZ         \I       \R      \          \SIZE OF WINDOW
C MSKNUM         \I       \R      \          \NUMBER OF MASK  TO USE
C ZMEAN          \R       \W      \          \LOCAL MEAN
C VARZ           \R       \W      \          \LOCAL VARIANCE
C Z              \R       \W      \          \VALUE OF CENTER PIXEL
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C ---------------------------------------------------------------------
C                     NON-LOCAL VARIABLES
C ---------------------------------------------------------------------
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C                \        \       \          \
C ---------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME        \          DESCRIPTION
C ---------------------------------------------------------------------
C                \
C                \
C                \
C                \
C ---------------------------------------------------------------------
      SUBROUTINE  EGSTAT (WINDOW, MSKWND, WNDSIZ,
     &  MSKNUM, ZMEAN, VARZ, Z)
C
       INTEGER  WNDSIZ, MSKNUM, REC, WORD
       INTEGER  MSKWND(8,WNDSIZ,WNDSIZ)
C
       REAL  WINDOW(WNDSIZ,WNDSIZ), SUBPTS
       REAL  ZMEAN, VARZ, Z, TOTALM, TOTALV, FLTVAL
C
C
       TOTALM = 0.0
       TOTALV = 0.0
```

```
      SUBPTS = (WNDSIZ/2+1) * WNDSIZ
C
      DO 20  REC = 1,WNDSIZ
       DO 10  WORD = 1,WNDSIZ
        FLTVAL = WINDOW(REC,WORD) * FLOAT(MSKWND (MSKNUM,REC,WORD))
        TOTALM = TOTALM + FLTVAL
        TOTALV = TOTALV + FLTVAL * FLTVAL
10     CONTINUE
20    CONTINUE
C
      ZMEAN = TOTALM / SUBPTS
      VARZ = TOTALV / SUBPTS  - ZMEAN*ZMEAN
C
      RETURN
      END
EOF..
?
```

```
C ---------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ---------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS          REF. # :
C ---------------------------------------------------------------------
C PROGRAM NAME:GENFLT          AUTHOR:J. SCOTT GARDNER DATE:3/23/83
C ---------------------------------------------------------------------
C PURPOSE :    THIS SUBROUTINE GENERATES THE
C FILTERS FOR THE WEINER FILTER.
C
C
C
C
C ---------------------------------------------------------------------
C                          PARAMETER DEFINITION
C  NAME               \ TYPE   \ CLASS\ RANGE   \     DESCRIPTION
C ---------------------------------------------------------------------
C FLTRS               \R       \R/W     \          \FILTERS ARRAY
C DIST                \R       \R/W     \          \DISTANCE ARRAY
C FLTNUM              \I       \R       \          \NUMBER OF FILTERS
C WNDSIZ              \I       \R       \          \SIZE OF THE FILTER WINDOW
C DELTAN              \R       \R       \          \INCREMENTAL NUMBER OF LOOKS
C NUMLKS              \R       \R       \          \NUMBER OF LOOKS
C WNDPTS              \R       \R       \          \NUMBER OF PTS IN WINDOW
C EQURES              \R       \R       \          \EQUIVALENT NUMBER OF LOOKS
C OUT2FC              \I       \R       \          \FILTER OUTPUT FILECODE
C TTYOUT              \I       \R       \          \TERMINAL OUTPUT FILECODE
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C ---------------------------------------------------------------------
C                          NON-LOCAL VARIABLES
C ---------------------------------------------------------------------
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C                     \        \        \          \
C ---------------------------------------------------------------------
C                          SUBROUTINES REQUIRED
C     NAME              \            DESCRIPTION
C ---------------------------------------------------------------------
C                     \
C                     \
C                     \
C                     \
C                     \
C                     \
C ---------------------------------------------------------------------
      SUBROUTINE  GENFLT  (FLTRS, DIST, FLTNUM, WNDSIZ, DELTAN, NUMLKS,
     &  WNDPTS, EQURES, OUT2FC, TTYOUT)
C
      INTEGER  FLTNUM, WNDSIZ, MID, ROW, CLM, FLT
      INTEGER  OUT2FC, TTYOUT
C
      REAL  DELTAN, FLTRS (FLTNUM,WNDSIZ,WNDSIZ)
      REAL  XDIFF2, YDIFF2, ALPHA, SUM, EQURES
      REAL  DIST (WNDSIZ,WNDSIZ), WNDPTS, TEMP
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      REAL  NUMLKS
C
C
C    FIRST, FIND THE MIDPOINT AND FILL THE DISTANCE ARRAY
C
      MID = WNDSIZ / 2  + 1
C
C
      DO 20   ROW = 1, WNDSIZ
       YDIFF2 = FLOAT ((ROW-MID) * (ROW-MID))
       DO 10   CLM = 1,WNDSIZ
        XDIFF2 = FLOAT ((CLM-MID) * (CLM-MID))
        DIST(ROW,CLM) = SQRT(XDIFF2 + YDIFF2)
10      CONTINUE
20    CONTINUE
C
C
C    CALCULATE FILTERS 2 THROUGH NUMFILTS - 1
C
      DO 90   FLT=2,FLTNUM-1
C
C    CALCULATE ALPHA BASED ON THE RELATIONSHIP THAT
C    ALPHA = 2 / W      WHERE W IS THE EQUIVALENT RESOLUTION FOR A 5X5
C    BOX FILTER.
C
C    QUANTIZING ALPHA GIVES      ALPHA = K * INDEX     WHERE K IS
C    A CONSTANT EVALUATED FOR THE CASE WHERE  ALPHA=.5 WHEN N=NUMLKS
C
C    THIS YIELDS   K = (2/W) * DELTAN / NUMLKS
C
C
      ALPHA = (2.0/EQURES) * DELTAN * FLOAT(FLT) / NUMLKS
C
C
C
C    NEXT CALCULATE THE VALUE OF EACH ELEMENT OF THIS FILTER
C
      SUM = 0.0
      DO 60  ROW = 1,WNDSIZ
       DO 50 CLM = 1,WNDSIZ
       TEMP = EXP (-ALPHA * DIST(ROW,CLM))
        SUM = SUM + TEMP
        FLTRS(FLT,ROW,CLM) = TEMP
50     CONTINUE
60    CONTINUE
C
C    NORMALIZE THE FILTER
C
      DO 80  ROW = 1,WNDSIZ
       DO 70  CLM = 1,WNDSIZ
        FLTRS(FLT,ROW,CLM) = FLTRS(FLT,ROW,CLM) / SUM
70     CONTINUE
80    CONTINUE
C
C
90     CONTINUE
C
C
C    CALCULATE THE FIRST AND LAST FILTERS
C
```

```
      TEMP = 1.C / WNDPTS
C
      DC 110  ROW = 1,WNDSIZ
       DO 100   CLM = 1,WNDSIZ
        FLTRS(1,ROW,CLM) = TEMP
        FLTRS(FLTNUM,ROW,CLM) = C.0
100    CONTINUE
110   CCNTINUE
C
      FLTRS(FLTNUM,MID,MID) = 1.C
C
C
C   PRINT OUT THE FILTERS
C
      DC 13C  FLT = 1,FLTNUM
       WRITE (CUT2FC,600) FLT
600   FCRMAT (/,/,/,10X,'THIS IS FILTER # ',I3,/)
      DC 12C  RCW = 1,WNDSIZ
       WRITE (CUT2FC,601) (FLTRS(FLT,ROW,CLM), CLM=1,WNDSIZ)
601   FCRMAT (1X,31(F9.6,2X))
120   CCNTINUE
130   CCNTINUE
C
C
      RETURN
      END
EOF..
?
```

```
C ---------------------------------------------------------------------
C  UNIVERSITY OF KANSAS TELECOMMUNICATIONS AND INFORMATION SCIENCES LAB
C ---------------------------------------------------------------------
C PROGRAM SUITE : NOISE FILTERS        REF. # :
C ---------------------------------------------------------------------
C PROGRAM NAME:FILTER          AUTHOR:J. SCOTT GARDNER DATE:02/14/83
C ---------------------------------------------------------------------
C PURPOSE :   THIS SUBROUTINE COMPUTES THE LOCAL STATISTICS FOR
C A WINDOW AFTER APPLYING A FILTER WHICH MUST
C PREVIOUSLY HAVE BEEN NORMALIZED.
C
C
C
C ---------------------------------------------------------------------
C                        PARAMETER DEFINITION
C  NAME            \ TYPE  \ CLASS\ RANGE  \     DESCRIPTION
C ---------------------------------------------------------------------
C WINDOW           \R      \R     \        \LOCAL AREA WINDOW
C FLTRS           \R      \R     \        \FILTERS ARRAY
C WNDSIZ           \I      \R     \        \SIZE OF WINDOW
C NUMFLT           \I      \R     \        \NUMBER OF FILTERS
C FLT              \I      \R     \        \NUMBER OF FILTER  TO USE
C ZMEAN            \R      \W     \        \LOCAL MEAN
C VARZ             \P      \W     \        \LOCAL VARIANCE
C Z                \R      \W     \        \VALUE OF CENTER PIXEL
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ---------------------------------------------------------------------
C                        NON-LOCAL VARIABLES
C ---------------------------------------------------------------------
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C                  \       \      \        \
C ---------------------------------------------------------------------
C                       SUBROUTINES REQUIRED
C    NAME            \          DESCRIPTION
C ---------------------------------------------------------------------
C                  \
C                  \
C                  \
C                  \
C ---------------------------------------------------------------------
      SUBROUTINE  FILTER (WINDOW, FLTRS, WNDSIZ,
     &  NUMFLT, FLT, ZMEAN, VARZ, Z)
C
      INTEGER  WNDSIZ, FLT, REC, WORD, NUMFLT
C
      REAL  FLTRS(NUMFLT,WNDSIZ*WNDSIZ)
      REAL  WINDOW(WNDSIZ,WNDSIZ)
      REAL  ZMEAN, VARZ, Z, TOTALM, TOTALV, FLTVAL
C
C
      TOTALM = 0.0
```

```
      TOTALV = 0.0
C
      DO 20  REC = 1,WNDSIZ
       DO 10  WORD = 1,WNDSIZ
        FLTVAL = WINDOW(REC,WORD) * FLTRS (FLT,REC,WORD)
        TOTALM = TOTALM + FLTVAL
        TOTALV = TOTALV + FLTVAL * FLTVAL
10     CONTINUE
20    CONTINUE
C
      ZMEAN = TOTALM
      VARZ = TOTALV  - ZMEAN*ZMEAN
C
      RETURN
      END
EOF..
?
```

APPENDIX B

Maximum Likelihood Classification of Synthetic Aperture Radar Imagery

[Paper for Computer Vision, Graphics, and Image Processing]

Maximum Likelihood Classification of Synthetic Aperture Radar Imagery

V.S. Frost, L.S. Yurovsky

Telecommunications and Information Sciences Laboratory
University of Kansas Center for Research, Inc.
Lawrence, Kansas 66045

Abstract

Classification of synthetic aperture radar (SAR) images has important applications in geology, agriculture and the military. A statistical model for SAR images is reviewed and a maximum likelihood classification algorithm developed for the classification of agricultural fields based on the model. It is first assumed that the target feature information is known *a priori*. The performance of the algorithm is then evaluated in terms of the probability of incorrect classification. A technique is also presented to extract the needed feature information from a SAR image; then both the feature extraction and the maximum likelihood classification algorithms are tested on a SEASAT-A SAR image.

Manuscript Received ______________________, Revised ________________.

This work was supported by NASA Headquarters Grant No. NAGW-381.

1.0 Introduction

Synthetic Aperture Radar (SAR) imaging systems have been used to obtain images of the Earth's surface. This paper deals with the problem of automatic classification of SAR imagery.

According to the definition in [1], classification of an object in the image is the decision regarding the category to which the object belongs. For example, if the image is of an agricultural area containing a number of fields and there are three categories, or classes of fields, such as corn, wheat, and soybeans, classification is the process of assigning a category to each field.

Each object (target) class is characterized by a unique set of measurable properties, or features. The classification is done by assigning the target to a class based on how closely the observed set of target features matches the set of features for that particular class. The assignment of an object to one of several classes can be done on the basis of a maximum likelihood criteria, that is, a classification decision is made such that the probability of incorrect classification is minimized. The problem of maximum likelihood classification has been presented in general form in [9], and solved for the specific case where the observed feature vector is distributed as multivariate normal random variable. In our case (SAR images), the classification is based on one observed feature, gray level, which is not normally distributed.The major problem with processing SAR images is that coherent nature of the microwave illumination gives rise to a phenomenon called speckle. Speckle noise seriously degrades the quality of an image [12]. It is signal-dependent. Because of the dependency of the noise on the signal, processing techniques designed for additive Gaussian noise fail. Thus other statistical models are needed, and one which has been successfully used in the past [15,16,20] will be introduced in the next section.

As has been mentioned above, a target in a SAR image can be described by a single feature - gray level mean. This is especially true for agricultural targets, where textural features do not offer much discrimination. Here, a maximum likelihood classifier based on gray level mean is designed. This classifier assigns every pixel of an image to one of several target classes. It is originally assumed that the gray level means are known a priori for all target classes. A method is then presented to estimate those values automatically. To evaluate the performance of the classifier, the

C-3

probability of classification error is derived, and the classifier performance is tested using radar image simulation and SEASAT-A SAR images.

2.0 Statistical Model for Intensities of Pixels in Synthetic Aperture Radar Images

It has been confirmed [16] that heuristic image processing techniques, such as gradient edge detection algorithms do not work well for SAR images. Thus a statistical model is needed, one which would incorporate the physics of the SAR image formation process. This model is the basis for image processing techniques developed later.

The main characteristic of SAR images is that the noise variance σ_x^2 of the pixel intensity X is proportional to μ_x^2, the square of the mean of this pixel. The true intensity μ_x is proportional to power return from the pixel in the absence of noise. The ramifications of proportionality of mean and variance is that targets with higher intensity have higher noise variance, and one can conclude that the noise power is the function of the signal power. This precludes the use of an additive white Gaussian noise (AWGN) model, where signal power and noise power are assumed to be independent.

Synthetic aperture radar images belong to the category of speckled images. The statistical characteristics of speckle noise have been developed in [13] and applied to synthetic aperture radar in [14,15]. The complete noise model was developed in [16,20]. According to this model, let R be a random variable representing observed pixel intensity from the SAR image. It has been shown that the probability density function (pdf) of R is exponential with the mean of μ where μ is the true intensity of the target to which this pixel belongs. It has been shown in [15] that pixel intensities of neighboring pixels can be assumed to be uncorrelated. Here we also assume that they are independent. In most SAR imaging systems more than one independent sample of detected power is obtained for each pixel. The number of independent samples, α is often referred to as "number of looks per pixel". Now pixel intensity X is obtained by averaging these independent samples:

$$X = \frac{1}{\alpha} \sum_{i=1}^{\alpha} R_i \tag{1.2}$$

The probability density function of X can thus be expressed by the formula for gamma distribution:

$$f_X(x) = \frac{x^{\alpha-1}\beta^{-\alpha}e^{-x/\beta}}{\Gamma(\alpha)} \qquad (1.3)$$

where,

$\beta = \mu/\alpha$

$\Gamma(\alpha) = (\alpha-1)!$ for α an integer.

Mean and variance of pixel intensity X can be calculated using the properties of gamma distribution. They are equal to μ and μ^2/α, respectively. The ratio of signal power to noise power (signal-to-noise ratio) [4] can be calculated as

$$\frac{S}{N} = \frac{E^2[X]}{Var[X]} = \alpha \qquad (1.6)$$

Thus it has been shown that noise power is proportional to signal power with coefficient of proportionality of $1/\alpha$.

This relates directly to the observation about dependency of noise on signal, made at the beginning of this chapter. Statistical noise model for SAR imagery can thus be best characterized as "multiplicative noise model", where signal and noise power are proportional to each other:

$$X = S \cdot N \qquad (1.7)$$

where X is defined as before, S is noise-free signal, and N is random variable, representing multiplicative noise. One should note that signal S is a continuous random variable and its probability density function is continuous.

There are restrictions to the model, worth mentioning:

(1) The statistical model for pixel intensities introduced above is only valid for a type of region that can be defined as homogeneous. A homogeneous region is an area where all pixels belong to the same target, best characterized by a single feature: mean μ. If more than one target is present in the region the distribution of pixel intensities does not follow gamma distribution, and more sophisticated model is needed [16]. Such regions are called edge regions, and since the model is not valid for them, they should be excluded in any analysis.

(2) There are several conditions to be imposed on the target for multiplicative noise model to be valid for SAR imagery. These conditions were outlined in [13]. One of them is that the surface roughness of the target should be large, compared with wavelength of radar signal. There is a variety

of target classes that would meet such conditions. An example is a target that contains an agricultural area, such as corn field.

In summary, the statistical model that incorporates properties of SAR image formation process was introduced, and its limitations mentioned. This model will be used throughout this paper.

3.0 Maximum Likelihood Classification of Regions in the SAR Images

This section presents a solution to the problem of the classification of pixels in a SAR image, based on gray level. Also, an expression for probability of incorrect classification is derived, and a test to exclude edge regions is presented.

In this section it is assumed that the image contains M targets. The true target means are known a priori. (An algorithm to extract target means from the SAR image will be presented later). A sliding KxK processing window is applied to the image, and sample mean is calculated for each window position. Then, based on the outcome of the test to be described, the center pixel of the window is assigned the value of the target mean, to which the region, defined by the window, is most likely to belong.

A. Maximum Likelihood Approach to Classification

For a given window position, let $X_1, X_2, X_3, \ldots, X_N$ (N=KxK) be the gray level intensities of the pixels within a window. According to our model, we can assume that $X_1, \ldots, X_N$ are independent and uncorrelated.

Let $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$ be the observed mean, and $X_{L_1}, X_{L_2}, \ldots, X_{L_M}$ be true means of all M targets in the image, known a priori, as previously assumed. It is assumed that all the pixels in the window belong to the same homogeneous region and that the region is characterized by the mean . Regions that contain a mixture of targets (edge regions) will be excluded by a test to be described later. Now consider the following set of hypotheses:

$$
\begin{aligned}
&H_1: \ \mu = X_{L_1} \ \text{(region belongs to Target 1)}\\
&H_2: \ \mu = X_{L_2} \ \text{(region belongs to Target 2)}\\
&\ \cdot\\
&\ \cdot\\
&\ \cdot\\
&H_M: \ \mu = X_{L_M} \ \text{(region belongs to Target M)}
\end{aligned}
\tag{3.1}
$$

The likelihood function under hypothesis H_i is defined as joint probability density function of all samples under the hypothesis H_i, $1 \leq i \leq M$, multiplied by a priori probability of hypothesis H_i being true, $P(H_i)$. Using the SAR image model previously described, the likelihood function can be written as

$$
L(X_{L_i}; X_1, X_2, \ldots, X_N) = P(H_i) \cdot \frac{\left(\prod_{j=1}^{N} X_j^{\alpha-1}\right) e^{-\alpha N\bar{X}/X_{L_i}}}{(\Gamma(\alpha))^N \left(\frac{X_{L_i}}{\alpha}\right)^{\alpha N}}
\tag{3.2}
$$

region. The test is carried out this way: If $\frac{\bar{X}}{S^2}$ is less than $\frac{\alpha}{2}$, the region is an edge region. Threshold of $\frac{\alpha}{2}$ has been chosen heuristically. To identify edge region, the center pixel of the window is assigned zero intensity, and no further testing described in previous section is needed. This test is a quick way to check whether the region defined by a processing window at a given position is homogeneous. However, it is not necessarily the optimal test. It may miss some of the edge regions. A better test (Likelihood Ratio Test) has been derived in [16]. Likelihood Ratio test would define the edge regions better at the expense of computational efficiency.

C. Performance Analysis of Maximum Likelihood Classification Algorithm

Section 3.A considered the problem of assigning center pixel of processing window in a SAR image to one of M targets. There is probability that the pixel is assigned to the wrong target. The purpose of this section is to derive an expression for this error probability. In [23], a performance analysis criteria were derived for two targets in the image and with Gaussian noise model assumed. The following analysis uses the multiplicative noise model and M targets in the image for derivation. Let

$Y_i = \frac{\bar{X}}{X_{Li}} + \ln X_{Li}$, $1 \leq i \leq M$, be our decision criterion. Let P_{e_1} be the probability of assigning the pixel to a target other than target 1, when hypothesis H_1 is true (that is, pixel belongs to Target 1).
Identically, P_{e_i}, $1 \leq i \leq M$ is the probability of assigning the pixel to target other than correct target with mean X_{Li}. P_{e_i} can also be expressed in terms of probability of correct decision:

$$P_{e_i} = 1 - P_{c_i} \tag{3.6}$$

P_{c_i} is the probability of correctly assigning the pixel to target i, when hypothesis H_i is true. Since all targets occur with equal probability, the average probability of misclassification can be calculated as:

$$P_e = \sum_{i=1}^{M} P_{e_i}/M \tag{3.7}$$

Thus the task of performance analysis boils down to finding P_{e_i}, $1 \leq i \leq M$.

Consider the following set of statistics:

$$Y_i = \frac{\bar{X}}{X_{L_i}} + \ln X_{L_i} \quad 1 \leq i \leq M \tag{3.8}$$

Let the true target means be arranged in ascending order: $X_{L1} < X_{L2} < \ldots < X_{LM}$. The probability of correctly assigning the pixel to Target 1, when hypothesis H_1 is true can be calculated as:

$$P_{c1} = P\big((Y_1 < Y_2,\ Y_1 < Y_3,\ \ldots,\ Y_1 < Y_M)\,|\,H_1\big) \tag{3.9}$$

where the comma in probability expression is defined as logical "and". Next, calculate the difference of random variables Y_i and Y_j where $1 \leq i \leq M$ and $1 \leq j \leq M$:

$$Y_i - Y_j = \frac{\bar{X}}{X_{Li}} + \ln X_{Li} - \frac{\bar{X}}{X_{Lj}} - \ln X_{Lj} = \bar{X}\,\frac{X_{Lj} - X_{Li}}{X_{Lj}X_{Li}} + \ln\frac{X_{Li}}{X_{Lj}} \tag{3.10}$$

The inequality $Y_i - Y_j < 0$ can thus be expressed as:

$$\bar{X} < \Big(\ln\frac{X_{Lj}}{X_{Li}}\Big)\,\frac{X_{Lj}X_{Li}}{X_{Lj}-X_{Li}} \tag{3.11}$$

Let $Z_{i,j}$ be equal to $\dfrac{X_{Lj}X_{Li}}{X_{Lj}-X_{Li}}\ln\dfrac{X_{Lj}}{X_{Li}}$. By inspection, one can conclude that $Z_{i,j} > 0$ and $Z_{j,i} = Z_{i,j}$. Rewriting the probability of correctly assigning pixel to target 1, we obtain:

$$P_{c1} = P\big((\bar{X} < Z_{1,2},\ \bar{X} < Z_{1,3},\ \ldots,\ \bar{X} < Z_{1,M})\,|\,H_1\big) \tag{3.12}$$

This joint probability can be expressed as the product of the following conditional probabilities:

$$\begin{aligned} P_{c1} = {} & P\big((\bar{X} < Z_{1,2})\,|\,H_1\big)\ P\big((\bar{X} < Z_{1,3}\,|\,\bar{X} < Z_{1,2})\,|\,H_1\big) \ldots \\ & P(\bar{X} < Z_{1,M}\,|\,\bar{X} < Z_{1,2},\ \ldots,\ \bar{X} < Z_{1,M})\,|\,H_1) \end{aligned} \tag{3.13}$$

But $X_{L1} < X_{L2} < \ldots < X_{LM}$, and therefore $Z_{1,2} < Z_{1,3} < \ldots, Z_{1,M-1} < Z_{1,M}$ so all conditional probabilities in the expression above are equal to unity i.e.,

$$P\left((\bar{X} < Z_{1,3} | \bar{X} < Z_{1,2}) | H_1\right) = 1$$

$\cdot$

$\cdot$

$\cdot$

$$P\left((\bar{X} < Z_{1,M} | \bar{X} < Z_{1,2}, \bar{X} < Z_{1,3}, \ldots, \bar{X} < Z_{1,M-1})\ H_1\right) = 1$$

and

$$P_{c1} = P(\bar{X} < Z_{1,2} | H_1) \tag{3.14}$$

By analogy the probability of correct classification for the largest a priori mean is:

$$P_{c_M} = P(\bar{X} > Z_{1,M} | H_M)\ P((\bar{X} > Z_{2,M} | \bar{X} > Z_{1,M}) | H_M)$$
$$P(\bar{X} > Z_{M-1,M} | \bar{X} > Z_{1,M}, \bar{X} > Z_{2,M}, \ldots, \bar{X} > Z_{M-2,M}) | H_M)$$

Again, all conditional probabilities are equal to unity, and

$$P_{CM} = P(\bar{X} > Z_{1,M} | H_M) \tag{3.15}$$

For any P_{ci}, $1 < i < M$, the expression becomes:

$$P_{ci} = P(\bar{X} > Z_{i,1}, \bar{X} > Z_{i,2}, \ldots, \bar{X} > Z_{i,i-1}, \bar{X} < Z_{i,i+1}, \ldots, \bar{X} < Z_{i,M} | H_i)$$

Or:

$$P_{ci} = P(\bar{X} > Z_{i,i-1} | H_i)\ P(\bar{X} < Z_{i,i+1} | \bar{X} > Z_{i,i-1}) | H_i)$$

It can be easily shown that the only targets that the pixel can be incorrectly assigned to when it belongs to target i, are targets with true means $X_{L_{i-1}}$ and $X_{L_{i+1}}$, directly below and above true mean X_{Li}. The probability of correct decision can be expressed as:

$$P_{c_i} = P(\bar{X} > Z_{i,i-1} | H_i)\ P(\bar{X} < Z_{i,i+1} | \bar{X} > Z_{i,i-1}) | H_i)$$
$$= P(Z_{i,i-1} < \bar{X} < Z_{i,i+1} | H_i). \tag{3.16}$$

In summary, probabilities of error P_{ei}, $P_{ei} = 1 - P_{ci}$, can be calculated as:

$$P_{e1} = P(\bar{X} > Z_{1,2} | H_1)$$

$\cdot$

$\cdot$

$\cdot$

$$P_{ei} = 1 - P(Z_{i,i-1} < \bar{X} < Z_{i,i+1} | H_i) \tag{3.17}$$

$\cdot$

$\cdot$

$\cdot$

$$P_{eM} = P(\bar{X} < Z_{1,M} | H_M)$$

Since the processing window size is large, one can apply central limit theorem to find the probability density function $\bar{X}$ under any hypothesis H_i $1 \leq i \leq M$. Expected value of $\bar{X}$ under hypothesis H_i is:

$$E[\bar{X} | H_i] = \frac{1}{N} \sum_{i=1}^{N} E[X_i] = \frac{1}{N} \sum_{i=1}^{N} = X_{Li} \tag{3.18}$$

And variance of $\bar{X}$ under the same hypothesis is:

$$\mathrm{Var}\ (\bar{X}|H_i) = \frac{1}{N^2} \sum_{i=1}^{N} \mathrm{Var}(X_i) = \frac{1}{N^2} \frac{N\alpha\ {X_{Li}}^2}{\alpha^2} = \frac{{X_{Li}}^2}{\alpha N} \tag{3.19}$$

Thus, according to the central limit theorem, it can be assumed that $\bar{X}$ has an approximate normal distribution under arbitrary hypothesis H_i, with mean of X_{Li}, and variance of ${X_{Li}}^2/\alpha N$. Let R_i be equal to:

$$R_i = \left(\frac{\bar{X} - X_{Li}}{X_{Li}}\right) \sqrt{\alpha N} \tag{3.20}$$

Under hypothesis H_i, R_i is approximately normal with zero mean and unit variance. Now we can express error probabilities in terms of R_i:

$$\begin{aligned}
P_{e_1} &= P\left(R_1 > \left(\frac{Z_{1,2} - X_{L1}}{X_{L1}}\right) \sqrt{\alpha N} \,\middle|\, H_1\right) \\
&\vdots \\
P_{e_i} &= P\left(\left(\frac{Z_{i,i-1} - X_{Li}}{X_{Li}}\right) \sqrt{\alpha N} < R_i < \left(\left(\frac{Z_{i,i+1} - X_{Li}}{X_{Li}}\right) \sqrt{\alpha N}\right) \,\middle|\, H_i\right) \\
&\vdots \\
P_{e_M} &= P\left(R_M < \left(\frac{Z_{1,M} - X_{LM}}{X_{LM}}\right)\sqrt{\alpha N} \,\middle|\, H_M\right)
\end{aligned} \tag{3.21}$$

The threshold $\left(\frac{Z_{i,j} - X_{Li}}{X_{Li}}\right) \sqrt{\alpha N}$ is equal to

$$\frac{\sqrt{\alpha N}}{T_{i,j}-1} (\ln T_{i,j} + 1 - T_{i,j})$$

where $T_{i,j} = \frac{X_{Li}}{X_{Lj}}$ is <u>the relative target contrast</u> of Target i with respect to Target j, $1 \leq I \leq M$, $1 \leq J \leq M$. $T_{i>j}$ can also be expressed in decibels. Now error probabilities can be expressed in terms of Q function, where the Q function is defined as:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-y^2/2} dy \tag{3.22}$$

The error probabilities can then be calculated as follows:

$$P_{e1} = Q\left(\left(\frac{\ln(T_{1,2}) + 1 - T_2}{T_{1,2} - 1}\right)\sqrt{\alpha N}\right)$$

$$\vdots$$

$$P_{ei} = 1 - Q\left(\left(\frac{\ln T_{i,i+1} + 1 - T_{i,i+1}}{T_{i,i+1} - 1}\right)\sqrt{\alpha N}\right) - Q\left(\left(\frac{\ln T_{i,i-1} + 1 - T_{i,i-1}}{T_{i,i-1} - 1}\right)\sqrt{\alpha N}\right) \quad (3.23)$$

$$\vdots$$

$$P_{eM} = 1 - Q\left(\left(\frac{\ln(T_{M,M-1}) + 1 - T_{M,M-1}}{T_{M,M-1} - 1}\right)\sqrt{\alpha N}\right)$$

D. Conclusions

Inspecting the expressions, derived in Section C one can make some important conclusions:

The maximum likelihood classifier can incorrectly assign the center pixel of processing window of size N only to targets with true means either above or below the mean of the target to which the region belongs. The error probability depends on two relative target contrasts $T_{i,i-1}$; $T_{i,i+1}$. The experiments to be discussed in a later section are consistent with these conclusions.

The performance of maximum likelihood classifier depends on the product of number of independent samples averaged by the imaging system (number of looks per pixel), α, and number of pixels in the processing window, N. Since the error does not depend on α and N individually, but rather on their product, there is a tradeoff between α and N. For instance, performance of maximum likelihood classifier on an image with 4 looks per pixel (α=4) and 25 pixels in the processing window (N=25) is equivalent to performance on the image with 10 looks per pixel and 10 pixels in the processing window. Thus we can call αN total number of looks or total number of samples.

The classification decision, derived in this section will yield the best error performance for a type of SAR image consistent with our statistical model. It has been shown in [9] that maximum likelihood classifier will minimize error probabilities. Therefore, this technique results in optimum performance for the given statistical model.

4.0 Target Mean Extraction Technique

The maximum likelihood classification algorithm, derived previously, assumes that true target means are known *a priori*. In many cases these means may not be available. This section deals with the problem of estimating target means. When the estimation process is complete, the resultant target means can be supplied to maximum likelihood classification algorithm. A test to compensate for imperfect extraction is also described here.

According to our statistical model, the pixel intensity of a homogeneous area is a gamma distributed random variable. Therefore the probability density function (pdf) of the whole image containing a number of targets can be characterized as pdf of a mixture of a number of gamma distributed random variables. The problem of estimating target means is equivalent to estimating the parameters of each mixing gamma pdf. Such estimation is possible if the mixture is identifiable (for definition of identifiability see [28]). There are methods to estimate parameters in identifiable mixtures [9]. However, the mixture of gamma variates is identifiable only if the random variable representing target means takes on only discrete values; otherwise the mixture is generally not identifiable [29]. Unfortunately, in our case the random variable representing target means has continuous pdf, and the mixture parameters cannot be estimated. Even if we assume that pixel intensities have approximately normal distribution, the mixture is still unidentifiable, because mixtures of normal pdf's are identifiable only if all normal variates used in the mixture have equal variances [28]. In our case the variances and the means for each area will be unequal. Therefore a different approach towards estimation of target means is needed. One of the approaches considered here is based on selecting the area that is most likely to be homogeneous and estimating target mean from such area.

A. Automatic Extraction of Target Means

The purpose of this section is to describe an automatic (unsupervised) target mean extraction procedure. The basic idea of the procedure is the following:

(a) The homogeneous areas of the image are identified, e.g. the ones that belong to the same target.

(b) These areas are combined into groups on the basis of likelihood of belonging to the same target. Therefore each target is identified by a respective group.

(c) All pixel intensities within each group are averaged to obtain estimates for target means.

In summary, pixels from homogeneous areas defined by the algorithm serve as a basis for estimating target means. Therefore the test to identify such regions has to be very stringent. The probability of selecting an edge area (e.g. area containing multiple targets) as homogeneous must be small. Conversely, the probability of missing (rejecting) a homogeneous region is not important. For example, if the target contains 500 local areas and only one of 500 is selected, this is still enough to provide a good estimate for the target mean. However, if a non-homogeneous area is selected as homogeneous, the estimate based on such area is wrong. Therefore, trying to minimize the probability of selecting edge areas as homogeneous is an important consideration.

The complete flow chart of the procedure is shown in Figure 1. The following is the description of flow chart block by block.

(1) The first step is to apply a sliding processing window (typical size 13x13) to the image. Then, for a given window position, the mean and variance of pixel intensities within the window is calculated.

(2) The variance test is a quick way to check whether the neighborhood (defined by the processing window) is homogeneous. It is basically a moment matching technique that tells how closely the observed variance of the neighborhood conforms to maximum likelihood estimate of that variance. The maximum likelihood estimate of the variance is predicted from the mean and is equal to:

$$\hat{\sigma}^2 = \frac{\bar{X}}{\alpha} \qquad (4.1)$$

α is a number of looks per pixel and is always known for a particular SAR imaging system [16]. If $|S^2 - \hat{\sigma}^2| < P$ where P is a threshold, the region may be homogeneous and chi square test is invoked to impose tighter constraints. If this inequality fails, the region is not homogeneous and no further testing is needed. The sliding window is then moved to another neighborhood.

(2) The next test for homogeneity uses a chi-square goodness-of-fit test. This test is a more stringent check of homogeneity. It provides a qualitative measure of how closely the gray level distribution measured from the neighborhood fits the gamma distribution, predicted by the model.

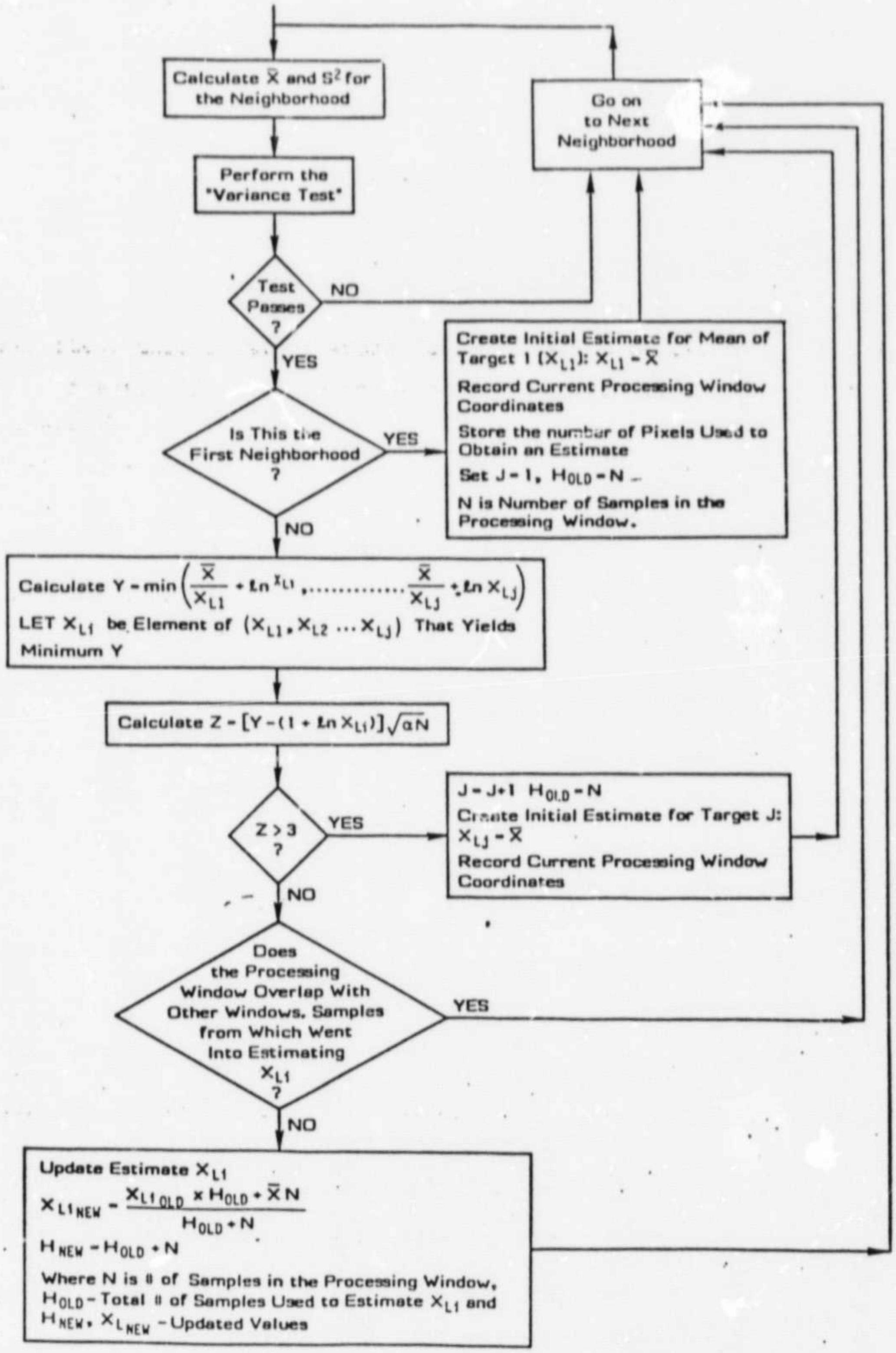


Figure 1. Flowchart of the target mean extraction algorithm.

Specifically, let X_1, X_2, ..., X_N be pixel intensities from the neighborhood. Again, according to the model, we can assume that random variables X_1, ..., X_N are independent and uncorrelated. A hypothesis test of the following form is then used:

$$
\begin{aligned}
H_o: f(X_i) &= \frac{X_i^{\alpha-1}\beta^{-\alpha}e^{-X_i/\beta}}{\Gamma(\alpha)} \quad 1 \leq i \leq N \\
H_i: f(X_i) &\neq \frac{X_i^{\alpha-1}\beta^{-\alpha}e^{-X_i/\beta}}{\Gamma(\alpha)} \quad 1 \leq i \leq N
\end{aligned}
\tag{4.2}
$$

where $f(X_i)$ is the probability density function of a given pixel X_i; α is number of looks per pixel and β is μ/α, as defined before. Since true mean μ is unknown, it can be replaced by its maximum likelihood estimate,

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i. \tag{4.3}$$

This is equivalent to the following hypothesis test:

H_o: neighborhood is homogeneous

H_1: neighborhood is not homogeneous

Dividing the range of pixel intensities into K intervals, the probability of random variable X_i being within given interval [a,b] is:

$$P(a \leq X_i \leq b) = {}_a\int^b \frac{X_i^{\alpha-1}\beta^{-\alpha}e^{-X_i/\beta}}{\Gamma(\alpha)} \, d_{X_i} = F(b) - F(a) \tag{4.4}$$

Where $F(y)$ is the distribution function of the pixel intensities. Thus expected number of samples from a neighborhood to fall within an interval [a,b] is $N(F(b) - F(a))$. The observed number of samples can be calculated from pixel intensity histogram for a given neighborhood. Define expected number of samples to fall within ith interval as $Nexp_i$, and actual number of samples as $Nobs_i$, respectively. Then define test statistic, χ, as follows:

$$\chi = \sum_{i=1}^{K} \frac{(Nobs_i - Nexp_i)^2}{Nexp_i} \tag{4.5}$$

where K is the number of intervals.

χ has an approximate chi-square distribution with (K-1) degrees of freedom. χ is defined as our measure of "degree of homogeneity". The greater χ is, the more non-homogeneous the neighborhood is likely to be. Thus we define homogeneous neighborhoods by the following test. We accept H_o if $\chi < \chi_T$, otherwise we reject H_o. Threshold χ_T is determined by the significance level of the test, and the typical values for χ_T are between 1 and 7. If we accept the hypothesis that the neighborhood is not homogeneous, we go on the next neighborhood.

(4) Having established that the neighborhood is homogeneous, and if the neighborhood is the first homogeneous neighborhood in the image, an initial estimate is created for the true mean of the first target. This estimate is obtained by taking the average of all pixels in the neighborhood. Also, the number of pixels that were averaged to obtain this initial estimate is recorded, along with processing window coordinates.

(5) If the homogeneous neighborhood is not the first one in the image, then there already exists some table of initial estimates. There are two possible outcomes: (a) The neighborhood belongs to a target, for which initial mean estimate has already been created; or (b) The neighborhood belongs to a target for which no initial estimate has been created. To check which of the two possible outcomes is true, the following statistic is calculated:

$$Y_i = \frac{\bar{X}}{X_{Li}} + \ell n\, X_{Li} = \min\left(\frac{\bar{X}}{X_{L_1}} + \ell n\, X_{L_1}, \ldots, \frac{\bar{X}}{X_{Lj}} + \ell n\, X_{Lj}\right)$$

where $\bar{X}$ is the local neighborhood mean, $X_{L1}, X_{L2}, \ldots, X_{Lj}$ are initial mean estimates for the first j targets, and $1 \leq i \leq j$. The statistic Y_i is nothing more than the maximum likelihood criterion, developed in section 3.

Now the two possible outcomes can be reformulated as (a) Neighborhood belongs to the target with mean X_{Li}; and (b) Neighborhood belongs to a new target with mean X_{Lj+1}, and initial estimate is : $X_{Lj+1} = \bar{X}$.

Under the hypothesis that the neighborhood belongs to target with mean X_{L_i}, Y_i is approximately normally distributed with mean $1 + \ln X_{L_i}$ and standard deviation of $1/\alpha N$. If the observation Y_i is within three standard deviations of its predicted mean $1 + \ell n\, X_{L_i}$, outcome (a) is likely. Otherwise, outcome (b) is likely. The measure of three standard deviations has been chosen heuristically.

(6) If Y_i is not within 3 standard deviations $\left(3/\sqrt{\alpha N}\right)$ of its mean, a new initial target estimate $\bar{X}_{Lj+1} = \bar{X}$ is created. Number of pixels in the neighborhood is also recorded, along with processing window coordinates, that define the current neighborhood. Then the processing window is moved to the next neighborhood.

(7) If Y_i is within 3 standard deviations of its mean, then the neighborhood is most likely to belong to the target with initial mean estimate of X_{Li}; $1 \leq i \leq j$. If that's the case, the target mean estimate can be updated by taking a weighted average of already existing estimate X_{L_i} and current neighborhood average $\bar{X}$.

It is important to assure that this weighted averaging required to obtain an estimate for a target mean is done over non-overlapping neighborhoods, e.g. the neighborhoods that don't have common pixels. Since each neighborhood is defined by processing window of fixed size at a given position, and window coordinates have been recorded, it's easy to check whether the current neighborhood overlaps with all other neighborhoods which were used to obtain an estimate for a given target mean X_{L_i}. If the current neighborhood does overlap, its gray level average cannot be used to update given target mean estimate X_{L_i}.

(8) If the neighborhood does not overlap with all others that belong to the same target, and pixels from which were used to obtain an estimate for target mean, then the initial estimate X_{Li} is updated by weighted averaging:

$$X_{Li\ new} = \frac{X_{Li\ old} \cdot H_{old} + \bar{X}N}{H_{old} + N} \tag{4.6}$$

$$H_{new} = H_{old} + N \tag{4.7}$$

where H_{old} is number of pixels that were used to obtain initial estimate, N is the number of pixels in the neighborhood. $X_{L_i\ new}$ and H_{new} are initial estimates after updating for a target mean, and number of pixels used to obtain that estimate, respectively. The current window coordinates are also recorded.

This procedure is a recursive procedure, the result of which is a vector of target mean estimates. Its performance will be evaluated in section 5.

B. A Test to Compensate for Imperfect Target Mean Extraction

The procedure described in the previous section leads to target mean estimates to be used as a input to maximum likelihood classifier. But these estimates may be imperfect. For instance, some targets may be missed, and others may be estimated incorrectly. This will increase the probability of classification error. However, there is a test to partially compensate for imperfect target mean estimates at the second (classification) stage.

As before, let Y_i be our decision criterion:

$$Y_i = \frac{\bar{X}}{X_{L_i}} + \ln X_{L_i}$$

Suppose that the neighborhood gets assigned to arbitrary target i with mean X_{L_i}. This test is similar to merging criterion described in the previous section. Under the hypothesis H_i being true, Y_i is approximately normal with mean of $1 + \ln X_{L_i}$ and variance of $1/\alpha N$. If the value of statistic Y_i is far enough from its expected value $1 + \ln X_{L_2}$ ("far enough" is a least three standard deviations: $3/\sqrt{\alpha N}$), then the center pixel of the neighborhood defined by processing window is assigned zero intensity. This test implies that even though the neighborhood belongs to a given target on the basis of maximum likelihood, its decision criterion's value is far apart from the expected value of that decision criterion. This may occur because the neighborhood belongs to a target, the true mean of which is missing, or has been estimated incorrectly. Therefore we don't really know which target the neighborhood belongs to and thus assign zero intensity to its center pixel to show that target couldn't be determined. Similar testing is done in the area of digital communications. When at the receiver it is not possible to determine whether the received bit of information was zero or one because of high ambiguity, the output bit is assigned "don't care" condition. Practice of assigning "don't care" condition to cases when it is not possible to determine correct status is often called "erasures", and is described in the communication literature [26]. The test developed here will be referred to as "erasure test" from here on.

5.0 Performance Evaluation

A. Verification of Performance of the Maximum Likelihood Classifier

A computer program based on the maximum likelihood classification algorithm described in Section 3 was written and tested on SAR image simulations and real SEASAT-A SAR image [27]. Simulated images were of size 250 x 250 pixels and contained 14 targets. This section discusses simulation results. The advantage of using simulated images to test the performance of the maximum likelihood classifier is that the location of each target and target means are known a priori. The description of simulation procedure is available in [21]. The basic idea of SAR image simulation is that a noiseless image (where all pixel values are equal to values of their respective target means) is multiplied by a random variable, which incorporates desired speckle, e.g. number of looks per pixel, α. As a result, the noisy image arises, with the noise statistics fitting the multiplicative noise model described previously. Such simulation is a good representation of a real SAR image. The noiseless image that serves as an input to simulation is called power map. The power map in Figure 2 was used for this experiment (the numbers in squares represent true target mean):

140	120	100	80	60
40	30	20	10	5
100	160	240	220	180
140	120	100	80	60
40	30	20	10	5

Figure 2. Power map for SAR image simulation

This image contains 14 targets 50x50 pixels each. The target means were selected such that a wide range of relative target contrasts would be present (from 0.38 dB to 3 dB). This is done because the probability of classification error depends on relative target contrasts (see Section 3.C). Also some of the same targets are separated and located in the different parts of the image to illustrate the fact that targets with the same mean can be disjoint. Five simulations were made, based on this power map. Images with α = 2, 4, 6, 8, and 10 were created. (See Figure 3). Then they were processed with 9x9 sliding window by the maximum likelihood classification algorithm.

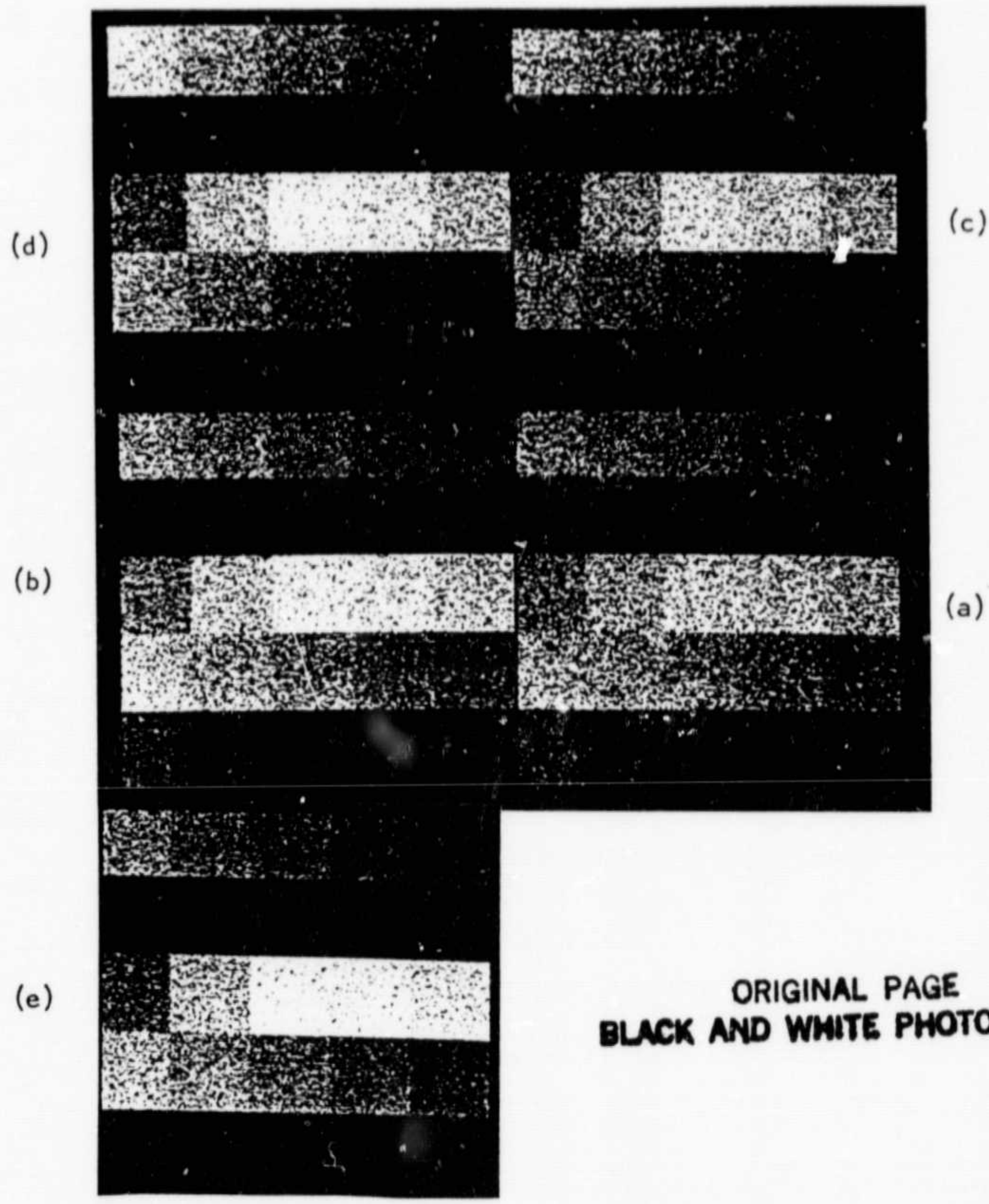


Figure 2. SAR Image Simulations: (a) $\alpha = 2$ (b) $\alpha = 4$ (c) $\alpha = 6$ (d) $\alpha = 8$ (e) $\alpha = 10$

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

(d) (c) (b) (a) (e)

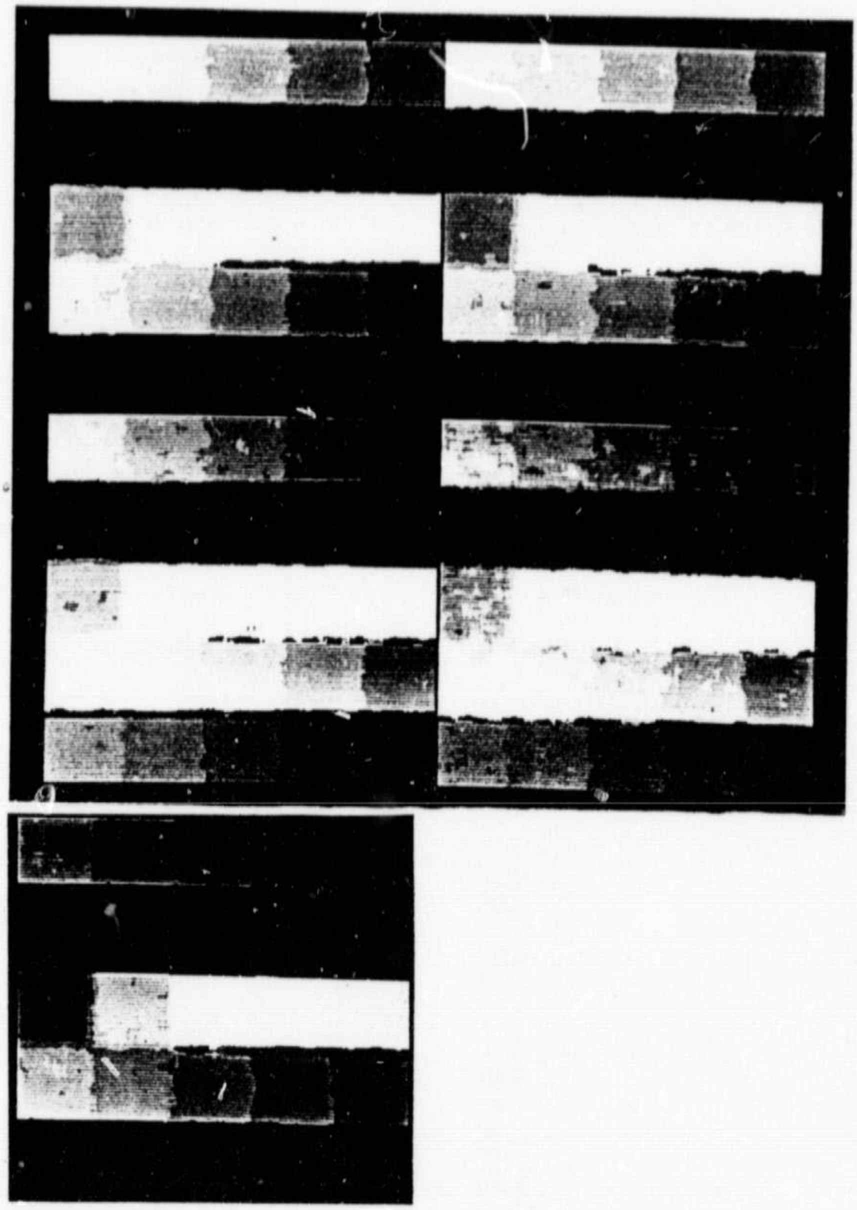
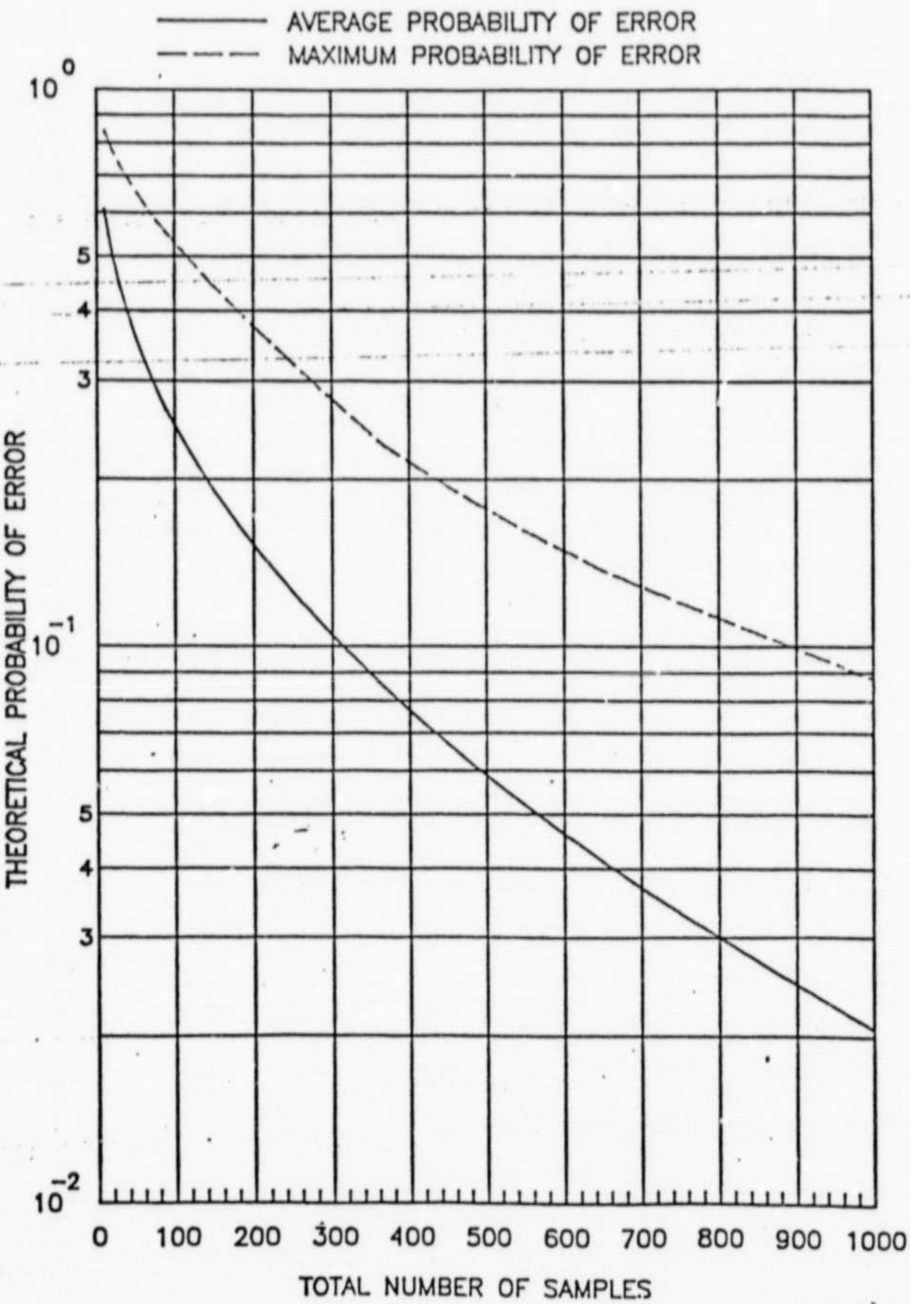


Figure 3. Results of Maximum Likelihood Classification Processing (a) α = 2 (b) α = 4 (c) α = 6 (d) α = 8 (e) α = 10

The results of processing are available in Figure 4. From inspecting the results, one can see that classification algorithm's performance improves as number of looks per pixel goes up, for a fixed window size. The black (zero intensity) points represent the edge regions selected by the test, described in Section 3.B. This edge region exclusion test defined most (but not all) local areas containing edges. It also defined as edge areas some areas that are homogeneous; but that was not a major problem, because such areas only constitute a small proportion of the image. The classification is more adversely affected by missing edge areas than defining false edges. For any given target, the misclassified pixels were assigned to a target with mean either above or below the mean of the given target, as expected. Also, the highest number of misclassifications occurred for a target with the lowest relative target contrasts $T_{i,i-1}$ or $T_{i,i+1}$, as defined in Section 3.C. All of these observations agree closely with the theory.

Figure 5 shows the plot of average probability of error $P_e = \frac{1}{M} \sum_{i=1}^{M} P_{ei}$ and maximum probability of error versus total number of samples, αN, for the simulations described above. These curves enable the operator to choose the right processing window size $\sqrt{N} \times \sqrt{N}$ to achieve the desired error performance for a given number of looks per pixel, α.



M=14 SAR IMAGE SIMULATION

Figure 5. Predicted Probability of Classification Error for Simulations From Figure 2.

B. Evaluation of Target Mean Extraction Technique

The target mean extraction algorithm was applied to the simulations described previously. Since the true means of all targets in these simulations are known a priori, it is possible to compare these means with the means, extracted by the algorithm for different numbers of looks per pixel. The threshold on a chi-square test, χ_T, was chosen to be 1.0, and the range of intensities was divided into 10 intervals, each having equal expected counts. The processing window was chosen to be 13x13. Table 1 compares results of the extraction procedure described above with known true means:

True Mean	Extracted Means				
	$\alpha = 2$	$\alpha = 4$	$\alpha = 6$	$\alpha = 8$	$\alpha = 10$
5.0	8.11	4.93	4.95	5.20	5.04
10.0	9.90	18.27	10.11	-	10.17
20.0	22.27	20.85	26.82	27.71	-
30.0	30.20	29.88	30.87	30.69	28.75
40.0	38.20	40.01	39.59	40.73	41.22
60.0	63.21	60.22	-	57.71	57.69
80.0	-	80.97	84.12	80.10	81.45
100.0	100.34	98.75	101.04	-	102.71
120.0	127.36	115.50	118.39	123.54	118.42
140.0	-	138.87	138.71	135.76	140.37
160.0	-	156.09	154.08	-	159.47
180.0	170.43	-	185.84	177.28	-
220.0	212.82	-	-	270.57	-
240.0	-	-	-	247.17	246.89

Table 1

One can observe that extraction technique has been able to estimate most target means correctly. However, there are some imperfections, like a few missing target mean estimates, and a few incorrect estimates. Effects of these imperfections can be partially compensated for by the test described in Section 4.B. The results of applying maximum likelihood classifier with erasure test to the simulations utilizing target means extracted by previously described algorithm are given in Figure 6. The dark areas represent the

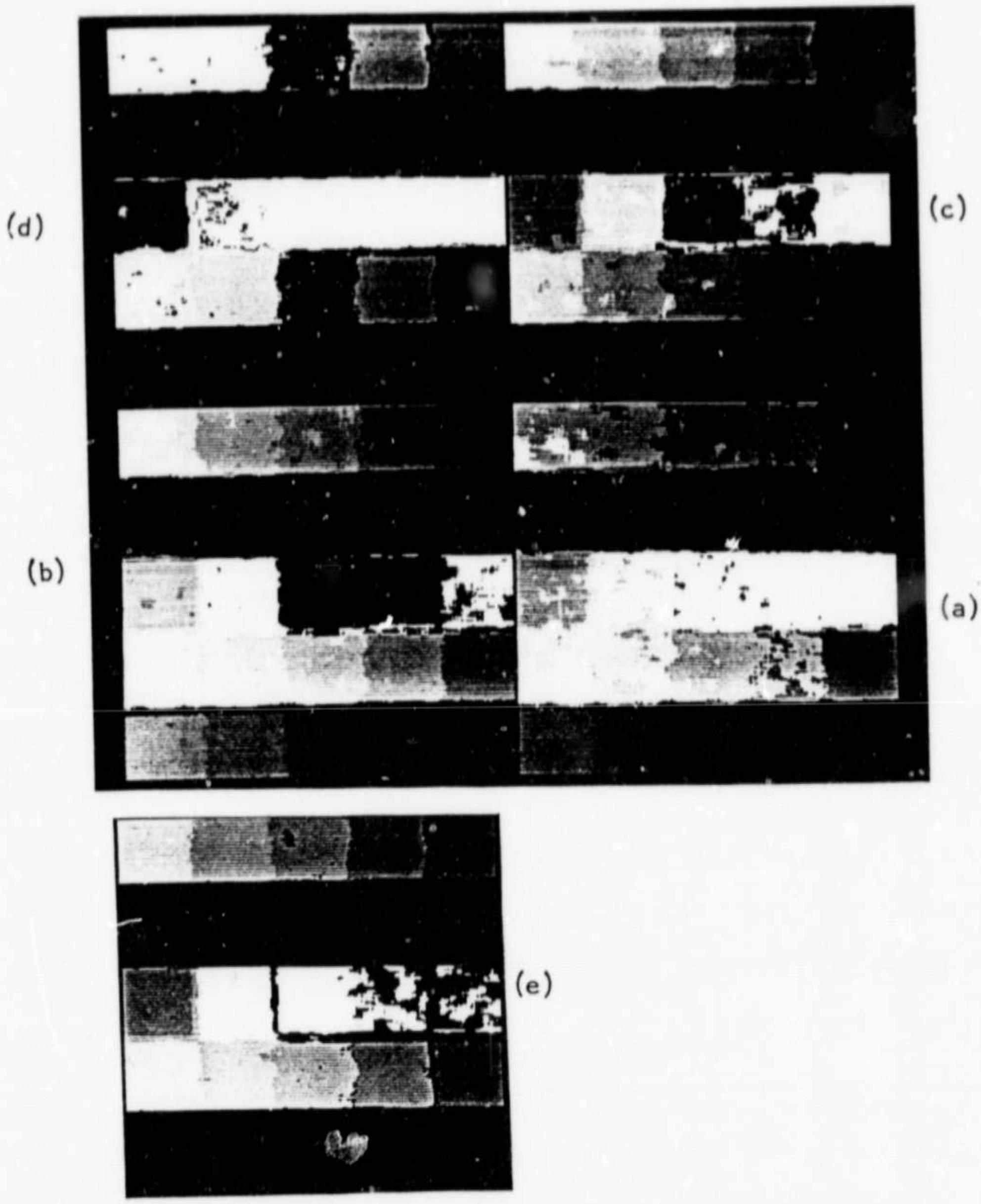


Figure 8. Results of Processing Simulations with Target Means Estimated by Extraction Algorithm ("Erasure test" was used to partially compensate for imperfect extraction) (a) $\alpha = 2$ (b) $\alpha = 4$ (c) $\alpha = 6$ (d) $\alpha = 8$ (e) $\alpha = 10$

regions, where classifier failed to make a decision. One can illustrate the effectiveness of erasure test by the following example:

In the case of $\alpha = 2$ simulation, the extraction routine missed a true target mean of 5 and extracted a false target mean of 8. The true target mean of 10 was found correctly and its estimate was 9.9. In the original image, targets with means of 5 and 10 are 50x50 squares next to one another in the bottom right-hand corner. Since true target mean of 5 was missing, the whole area which was supposed to belong to that target was assigned zero intensity. If no erasure testing was done, this whole area would have been assigned the value of false target mean of 8. The next square has a true target mean of 10, and most of its pixels were classified likewise. Only very few pixels were assigned the value of false label 8, most of them at the edge of areas with true target means of 5 and 10. Thus, a number of classifications to the false label was significantly reduced.

Of course, if mean of 8 was not extracted, there wouldn't have been any misclassifications at all. Therefore, it always degrades the performance more to extract a false target mean then to miss a true one. Yet the test, described in 4.B provides some degree of protection in both cases.

C. Application of Complete Algorithm to Real SEASAT Image

The SEASAT image [27] used in this experiment was 512x512 image of agricultural scene. This image is shown in Figure 7. The number of looks per pixel, α, was approximately 3. The image was processed by maximum likelihood classifier with 9x9 processing window size with edge test and compensation test included. Target means were extracted manually, and automatically. Resulting images are given in Figure 8 and 9, respectively. Table 2 lists target means, extracted in both cases. Note that neither of the extraction techniques is perfect.

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

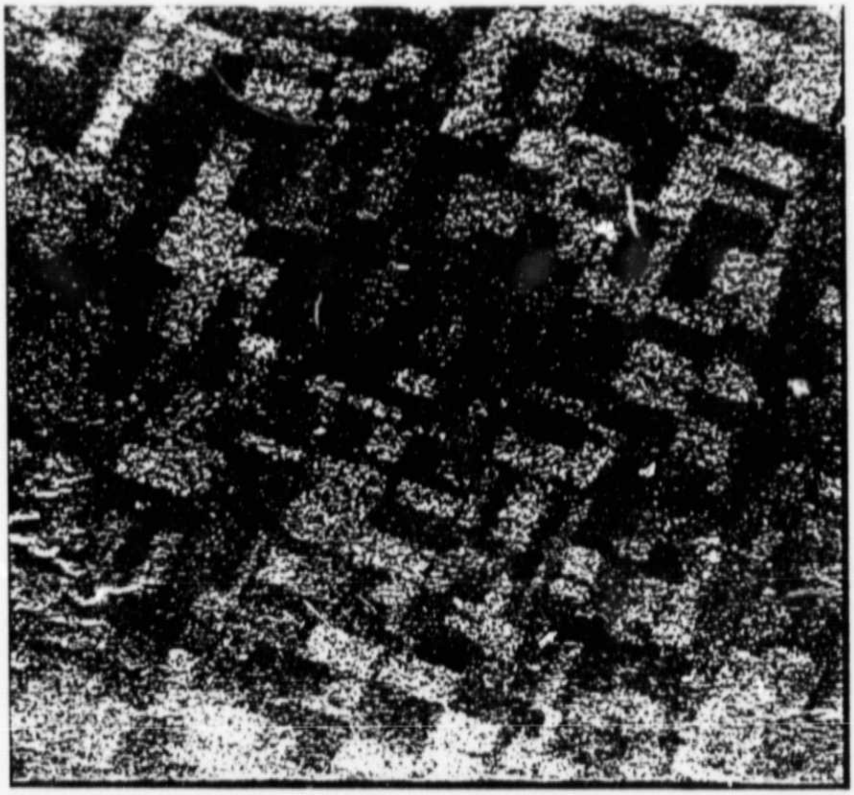


Figure 9. SEASAT-A-SAR Image

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

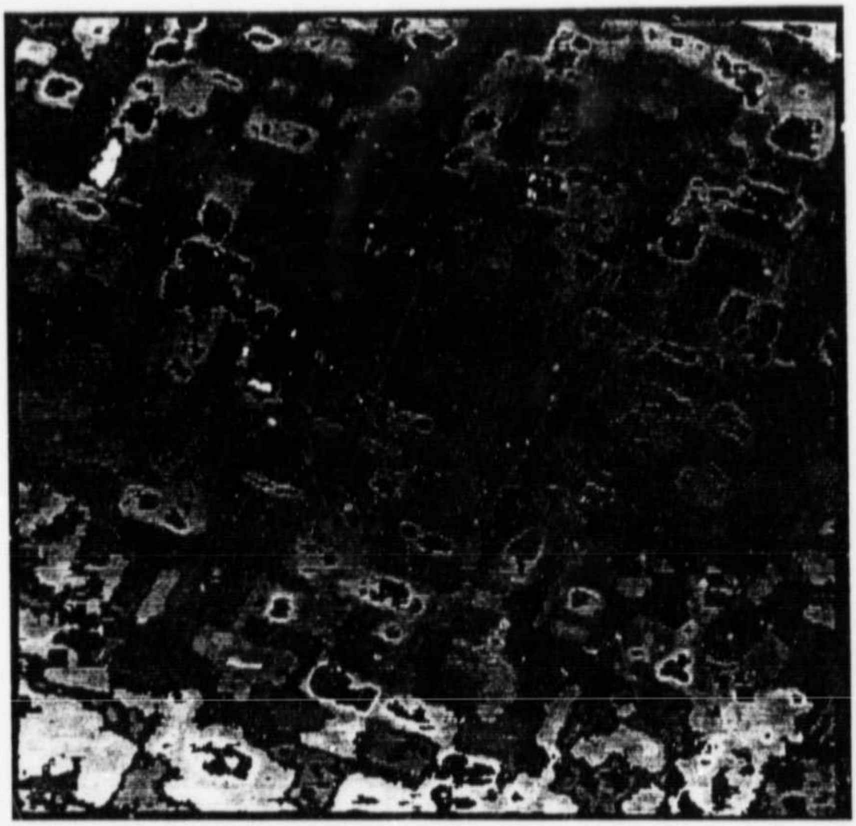


Figure 10. Result of Processing SEASAT-A-SAR Image (Target means extracted manually)

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

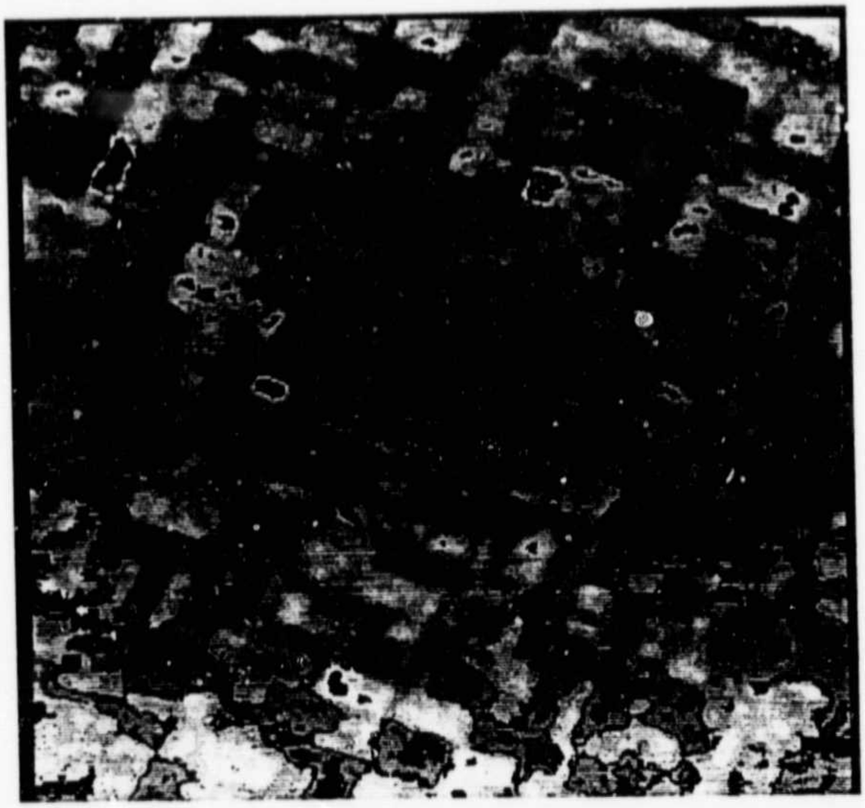


Figure 11. Result of Processing SEASAT-A-SAR Image (Target means estimated by extraction algorithm)

Real SEASAT Image Target Means

Estimated Manually	Estimated Automatically
15.3	18.05
25.0	27.50
41.9	40.96
48.9	51.28
59.8	62.69
107.8	72.57

Table 2

The performance in both cases is quite reasonable, especially compared with results of processing the same image using other techniques such as different types of gradient edge detectors [16]. A major problem is high misclassification rate at boundaries of targets (edges). This occurs because the statistical model doesn't apply to edge regions, and the mean^2/variance test described in Section 3.B fails to select all the edge regions. This problem would be reduced if more sophisticated edge test was used, such as likelihood ratio test, available in [16].

6.0 Conclusions

To summarize a maximum likelihood classification algorithm was developed for SAR images, and its theoretical performance was evaluated. Also, automatic extraction algorithm was developed to estimate target mean levels, and a test to partially compensate for imperfect extraction was introduced. The algorithms were tested, using simulations and SEASAT-A SAR imagery.

The classification scheme developed here is the best classification possible based on the given statistical model. This statement can be made because the maximum likelihood approach minimized the probability of classification error [9]. Another advantage of maximum likelihood classification algorithm is a high computational efficiency. It takes only slightly longer to complete classification then to complete equal weighted filtering of an image.

There are two major problems. First, is high misclassification rate at edges. This problem is created by the fact that the edge test, described in Section 3.B does not detect all edge regions. The problem of high misclassification rate at edges can be overcome by applying a better edge detector, for instance, maximum likelihood edge detector, developed in [16]. Another problem is the imperfect extraction of target means. Partially, this problem is overcome by applying compensation test, described in Section 4.B. The extraction algorithm is by no means optimal, although it does estimate most target means well (see Section 5.B). Further research is needed to improve extraction techniques. Finally, classification error can be reduced by applying a simple post-processing algorithm. This algorithm would select pixels or small groups of pixels that have been assigned to a target other than the majority of surrounding neighbors. These "isolated" pixels would then be re-assigned the same target level of as its neighbors.

7.0 References

[1] K.R. Castleman, Digital Image Processing, Prentice Hall, 1979.

[2] A. Rosenfeld, A. Kak, Digital Picture Processing, Second Edition Academic Press, 1982.

[3] R.C. Gonzalez, P. Wintz, Digital Image Processing, Addison-Wesley, 1977.

[4] I.E. Abdou, W.K. Pratt, Quantitative Design and Evaluation of Enhancement/Thresholding Edge Dete tors. Proceedings of IEEE, Vol. 67, No. 5, May 1979.

[5] G.B. Shaw, Local and Regional Edge Detectors: Some Comparisons, Computer Graphics and Image Processing, No. 5, 1979.

[6] S.W. Zucker, Region growing: Childhood and Adolescence, Computer Graphics and Image Processing No. 5, 1976.

[7] R.M. Haralick, K.S. Shanmugan, I. Dinstein, "Textural Features for Image Classification," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-3, November 1973.

[8] R.M. Haralick, "Statistical and Structural Approaches to Texture," Proceedings of the IEEE, Vol. 67, No. 5, 1979.

[9] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley, 1973.

[10] J.J. Kovaly, Synthetic Aperture Radar, Artech House, 1976.

[11] K. Tomiyasu "Tutorial Review of Synthetic Aperture Radar (SAR) with Applications to Imaging of the Ocean Surface", Proceedings of the IEEE, Vol. 66, No. 5, May 1978

[12] P.M. Shankar, H.M. Gupta, "Image Detection in the Presence of Speckle", Proceedings of IEEE, Vol. 67, No. 2, February 1978.

[13] J.W. Goodman, Some Fundamental Properties of Speckle, J. Opt. Soc. Amer. 66, No. 11, 1976.

[14] L.J. Porcello, et al., Speckle Reduction in Synthetic Aperture Radars, J. Opt. Soc. Amer. 66, No. 11, 1976.

[15] J.S. Lee, Speckle Analysis and Smoothing of Synthetic Aperture Radar Images, Computer Graphics and Image Processing, No. 17, 1981.

[16] V.S. Frost, et al., Topics in Radar Image Analysis, Remote Sensing Lab Technical Report 453-9, University of Kansas Center for Research, Inc., Lawrence, KS 66045.

[17] A.D. Whalen, Detection of Signals in Noise, Academic Press, 1971.

[18] J.L. Devore, Probability and Statistics for Engineering and the Sciences, Brooks/Cole Publishing Co., 1981.

[19] K.S. Shanmugan, Digital and Analog Communication Systems, John Wiley, 1979.

[20] V.S. Frost, et al., A Model for Radar Images, and Its Application to Adaptive Digital Filtering of Multiplicative Noise, IEEE Transactions on Pattern Analysis and Machine Intelligence, March, 1982.

[21] J.C. Holtzman, V.S. Frost, J.A. Stiles, V.H. Kaupp, "Radar Image Simulation", IEEE Trans. Geosci. Election, vol. GE-16, pp. 296-303, Oct. 1978.

[22] A.M. Breipohl, Probabilistic Systems Analysis, John Wiley, 1971.

[23] D.J. Shazeer, Performance Measures for Statistical Segmentation, Applications of Digital Image Processing IV (1982), SPIE Vol. 359.

[24] A. Papoulis, Probability, Random Variables, and Stochastic Processes, McGraw-Hill, 1965.

[25] R.V. Hogg, A.T. Craig, Introduction to Mathematical Statistics, Macmillan Publishing Co., 1978.

[26] J.G. Proakis, Digital Communications, McGraw-Hill, 1983.

[27] Jordan, R.L., The SEASAT-A Synthetic Aperture Radar System, IEEE Journal of Oceanic Engineering, OE-5, 2 (Apr. 1980) 154-165.

[28] H. Teicher, On the Mixture of Distributions, Annals of Mathematical Statistics, vol. 31, p. 55, 1960.

[29] H. Teicher, Identifiability of Finite Mixtures, Annals of Mathematical Statistics, vol. 34, p. 1265, 1963.

APPENDIX C

A Data Compression Technique for Synthetic Aperture Radar Images

(Paper for IEEE Transactions on Aerospace Electronics Systems)

A DATA COMPRESSION TECHNIQUE FOR SYNTHETIC APERTURE RADAR IMAGES

V. S. Frost

G. J. Minden

Manuscript Received __________; revised __________.
This work was supported by NASA under Grant NAGN-381. The authors are with the Telecommunications and Information Sciences Laboratory, University of Kansas Center for Research, Inc., Lawrence, Kansas 66045.

A DATA COMPRESSION TECHNIQUE FOR
SYNTHETIC APERTURE RADAR IMAGES

ABSTRACT

A data compression technique is developed for Synthetic Aperture Radar (SAR) imagery. The technique is based on a SAR image model and is designed to preserve the local statistics in the image by an adaptive variable rate modification of block truncation coding (BTC). A data rate of approximately 1.6 bits/pixel is achieved with the technique while maintaining the image quality and cultural (point like) targets. The algorithm requires no large data storage and is computationally simple.

1. Introduction

The purpose of this paper is to describe an efficient image data compression technique which has been specifically designed for Synthetic Aperture Radar (SAR) images. SAR has become an important class of imaging sensor for both civilian and military applications. As with other imaging systems there is a need to transmit and store SAR images, and thus, there has been considerable interest in efficient coding algorithms for SAR [1,2,3].

The aim of data compression is to minimize the data rate while maintaining the information contained in the signal using as simple an algorithm as possible. Thus a desirable property of an image encoding algorithm is fidelity, i.e. the reconstructed (received) image should preserve of all of the "important" features of the sensed image. For example, in some radar applications cultural features which appear in SAR images as small bright features are important and should be faithfully reproduced. In the geologic analysis of SAR images texture [4] is important and thus should be preserved. Image fidelity (quality) is a difficult quantity to measure [5] because it is application dependent. Image data compression algorithms are also compared on the basis of their compressing capability, i.e. the number of bits per image sample in the coded image. Implementation complexity is also an important consideration in evaluating data compression algorithms [6]. The technique described below preserves important image features (e.g. cultural features) at about 1.6 bits/sample and is simple to implement.

Standard compression techniques fall into two broad categories--predictive and transform coding. Predictive coding is performed in the spatial domain and attempts to remove the local redundancies in the image. Transform coding is performed by an energy preserving transformation of the image into another image so that the maximum information is placed into a minimum number of transform components [6]. Many different transforms, e.g. Fourier, Cosine, Karhune-Loeve, have been used. Transform coding tends to be more complex than predictive coding. It will be shown in Section 2 that predictive coding is not possible on SAR images, and further, it will be argued that transform coding is not a viable alternative because of the low correlation observed in SAR images even though it has been tried [1].

The technique developed here is a modification of the Block Truncation coding (BTC) algorithm developed in [7]. BTC is suitable for SAR images because it preserves the local statistics of the image. In SAR these statistics are important. In BTC the image is divided into small blocks (e.g. 4x4) of picture elements (pixels) and for each block a one bit quartizer is applied such that the block can be reconstructed with the moments (e.g. mean and variance) preserved. Clearly, in addition to the bit mask (quantized block), supplementary information is needed. In the BTC the supplementary information is the sample mean, $\bar{x}$, and standard deviation, s_x. For example, a BTC system using 4x4 blocks of picture elements (pixels) (with 8 bits/pixel) and 8 bits to code $\bar{x}$ and s_x results in a 4 to 1 compression or 2 bits/pixel. Reducing the number of mean and standard deviation code bits as well as further coding of the bit mask can result in further compression [7,8].

Direct application of the BTC algorithm [7] (2 bits/pixel) to SAR images produced reconstructed images that were of suitable quality visually and preserved cultural features. These results will be presented in Section 6. A further reduction in bit rate was achieved by observing that the local mean and variance are proportional in SAR images of homogenous areas, and thus, it is required to transmit only the mean. This modification results in a 5.3 to 1 compression or 1.5 bits/pixel (using the above example).

This modification did produce reconstructed images of acceptable quality, however, significant contrast was lost for cultural features. This weakness was overcome by developing an adaptive BTC algorithm. The adaptive BTC algorithm sends only the mean if the local area (block) fits the standard radar model. For those blocks where the model does not fit, both the mean and standard deviation are transmitted.

Using this adaptive approach, a 5 to 1 compression or about 1.6 bits/pixel was achieved with the quality of the original 2 bits/pixel BTC algorithm. Now a variable number of bits per block is required for the adaptive BTC technique. However, this modification to the BTC algorithm does not significantly increase its complexity.

A statistical model for SAR images will be reviewed in Section 2. The original BTC technique will be discussed in Section 3. The modifications to the BTC algorithm for SAR images will be described in the following section. The BTC, modified BTC (mean only) and adaptive BTC were implemented and tested using SEASAT-A SAR imagery. These results are presented in Section 6. The SAR image data compression described here is simple, produces reconstructed images of adequate quality for many applications and tends to preserve cultural futures.

2. A Statistical Model for Radar Images

2.1 Point Statistics

An imaging radar illuminates areas of the terrain within its field of view and records the value of the power returned from nonoverlapping resolution cells on the ground. A resolution cell is typically made up of a large number of scatterers, and under some mild assumptions we can model the signal received by the radar (before detection) as a narrowband Gaussian random process. Then, with a square-law detector, the value of the received power P from a resolution cell has the probability density function [9, 10, 11]

$$f_p(p) = \mu^{-1} \exp(-p/\mu) \qquad \text{for } p > 0 \tag{1}$$

where $E\{P\} = \mu$.

In most imaging radars several independent measurements of the reflected power for each resolution cell are obtained and are averaged to form the image intensity value $Y(t_1,t_2)$ for the resolution cell with a spatial location (t_1,t_2). The probability density function (pdf) of Y is the gamma distribution [9, 10] of the form

$$f_y(y) = \frac{y^{N-1} (\mu/N)^{-N} \exp(-yN/\mu)}{\Gamma(N)} \qquad y > 0 \tag{2}$$

where N is the number of independent measurements (or "looks"),

$$Y = \frac{1}{N} \sum_{i=1}^{N} P_i \text{ and } \Gamma(N) = (N-1)!.$$

The mean value μ of the power reflected from a resolution cell is proportional to the radar reflectivity X of the resolution cell, and we can assume that $\mu = X$ without any loss of generality. Since the radar reflectivity changes from resolution cell to resolution cell, we can model the reflectivity as a random variable X (or a random process $X(t_1,t_2)$) and write (2) as a conditional pdf of the form

$$f_{Y|X}(y|x) = \frac{y^{N-1}\ x^{-N} \exp(-yN/x)}{\Gamma(N)\ N^{-N}} . \tag{3}$$

We assume a Swerling type II [10] target model to describe the statistical characteristics of the echo on a per-pixel basis and then let the mean reflectivity X vary to model the SAR image of a large heterogeneous scene. With an appropriate change of variable we obtain the relationship between X and Y:

$$Y(t_1,t_2) = \frac{X(t_1,t_2)\ Z(t_1,t_2)}{2N} \tag{4}$$

where Z has a standard chi-square distribution with 2N degrees of freedom [10], and X and Z are statistically independent.

Note that $Z(t_1,t_2)$ represents the speckle noise in SAR images. Here we have explicitly shown that X and Z are functions of position, however, for notational convenience the spatial dependence will be dropped. It can be easily shown that for a given X

$$E[Y/X] = \mu \tag{5}$$

$$\sigma^2_{Y/\bar{X}} = \frac{\mu^2}{N}. \tag{6}$$

For our purposes, we can see that if we are considering an "homogeneous" target (i.e., $E[X] = \mu$), then we can predict the variance given the mean. This observation is the basis for the modified BTC algorithm.

2.2 Autocorrelation Properties of SAR Images

The feasibility of using either predictive or transform coding for SAR image can be discussed in terms of the image correlations properties. On a local level, i.e. inside a homogenous area, the model (equation 4) indicates that adjacent pixels will be uncorrelated. This is quite different compared to images collected with noncoherent sensors. For noncoherent sensors, e.g. LANDSAT or Aerial photographs, pixels in homogenous areas are highly correlated. The low correlation of adjacent pixels has been observed previously [12] and eliminates predictive coding from consideration.

However, transform coding can can operate over larger regions and and thus the regional correlation properties of SAR images needs to be considered. Figure 1 presents a typical autocorrelation* in the row and column directions of a SAR image of terrain (see Figure 4 for the image). Note that this autocorrelation functions decays very rapidly. The presence of some correlation indicates that transform coding is possible, however the rapid decay implies that the size of the transform window must be large, thus greatly increasing the memory and computational requirements of the compression algorithm. A sophisticated transform coding algorithm for SAR images has been reported [1]. The technique described in [1] uses row/column deletation (resampling) and transform coding; however, the algorithm is computationally intensive and requires substantial amounts of memory.

Reexamining the SAR image model (equation (4)) we notice that the mean and variance of each local area (within a larger homogeneous region) are redundant, and thus, a coding algorithm which preserves these image attributes would be suitable for SAR. In the next section, the BTC algorithm which does preserve these features is discussed, and it is modified to fit the above SAR image models.

3. Review of Block Truncation Coding

Let $\bar{y}$ and S_y denote the sample mean and standard deviation of a block (e.g. 4x4) of pixels in a SAR image. That is

$$\bar{y} = \frac{1}{m} \sum_{i=1}^{m} y_i \tag{7}$$

and

$$S_y^2 = \overline{(y^2)} - (\bar{y})^2 \tag{8}$$

with

$$\overline{(y^2)} = \frac{1}{m} \sum_{i=1}^{m} y_i^2 \tag{9}$$

where

m = number of pixels in the block

and

y_i = pixel intensity

*The image contained 512 x 512 pixels and the autocorrelation function was obtained using FFT techniques.

In BTC a one bit quantizer is used for each pixel in the block with the quantizing threshold set at $\bar{y}$. That is, if $y_i \geq \bar{y}$ then that pixel location is coded with a 1, otherwise with a 0. A bit mask is thus formed. This bit mask along with $\bar{y}$ and S_y are transmitted/stored.

At the receiver, a level A is assigned to a point if that pixel location within the block contained a 0 and a level B if a 1 was contained. The levels A and B are selected to preserve the moments of the block. These levels can be simply found as [7, 13]

$$A = \bar{y} - S_y \sqrt{\frac{q}{m-q}} \tag{10}$$

and

$$B = \bar{y} + S_y \sqrt{\frac{m-q}{q}} \tag{11}$$

where

q = number of one's in the received block.

For a 4x4 block and using 8 bits to code $\bar{y}$ and S_y results in 4 to 1 compression ratio or 2 bits/pixel.

4. Modified Block Truncation Coding

Based on the SAR image model, we can predict the standard deviation for a block based on the sample mean. Let the predicted standard deviation be

$$\sigma_p = \frac{\bar{y}}{\sqrt{N}} \tag{12}$$

where

N = number of looks for the SAR.

The BTC technique described in Section 3 is then applied at the source. Now only the sample mean and the bit mask are transmitted. At the receiver, the levels A & B are reconstructed using

$$A = \bar{y} \left(1 - \sqrt{\frac{q}{N(m-q)}}\right) \tag{13}$$

and

$$B = \bar{y} \left(1 - \sqrt{\frac{m-q}{Nq}}\right) \tag{14}$$

For a 4x4 block and using 8 bits to code the sample mean a compression ratio of 5.3 to 1 or a data rate of 1.5 bits/pixel is obtained.

It was found that this technique produced reconstructed images of homogeneous areas almost identical to the 2 bit/pixel technique. However, contrast was lost on cultural features relative to the initial BTC method. The reason for this is obvious. Regions containing cultural features do not fit the SAR model of equation (4). To overcome this weakness an adaptive variable bit rate BTC was developed.

5. Adaptive Variable Bit Rate Block Truncation Coding

The goal of the adaptive BTC technique is to use the modified BTC for those image blocks where it is appropriate, i.e. where the model fits, and to use the original BTC algorithm otherwise. A simple test based on the predicted and sample variances was developed to indicate if the data from an image block fits the SAR model. Specifically, if $k \cdot S_y^2 > \sigma_p^2$ (where k is a constant) then the model does not fit the data. That is, if the observed (sample) variance is "too much" greater than the predicted variance then it would be expected that the model does not provide an adequate description for the data. The proportionality constant, k, must be selected such that the probability of rejecting the model when the model is valid, P_F, is small, i.e.,

$$P_F = P(k\ S_y^2 > \sigma_p^2 \mid \text{model is valid}) \tag{15}$$

From the theory of confidence intervals [14] we know that we can find a β such that

$$P\left(\frac{m\ S_y^2}{\sigma_p^2} > \beta\right) = P\left(\sigma_p^2 < \frac{m\ S_y^2}{\beta}\right) = P_F \tag{16}$$

The above equation indicates that the probability that $m\ S_y^2/\beta$ (here $k = \frac{m}{\beta}$) is greater than the unknown parameter σ_p^2 is P_F when the SAR model fits the data. The predicted variance σ_p^2 is calculated using equation (12). Therefore, the SAR model is not appropriate if

$$\frac{\bar{y}^2}{N} < \frac{m}{\beta} S_y^2 \tag{17}$$

or

$$\frac{\bar{y}^2}{S_y^2} < \frac{mN}{\beta} \tag{18}$$

where β is selected to force P_F to be small, e.g. 10^{-3}. Note that $\bar{y}^2/S_y{}^2$ is an estimate for the number of looks used in the SAR processing based on the m pixels in the block. This ratio will be referred to as the local number of looks.

Assuming that the pixel intensities are Gaussian distributed (which is true for large N) the random variable $m\ S_y{}^2/\sigma_p{}^2$ has a χ^2 distribution with m-1 degrees of freedom. Thus, P_F can be estimated. For example, let m = 16 (i.e. 4x4 blocks) and N=4 then for a $P_F = .005$, $\beta = 32.8$, $k \simeq .5$ and $mN/\beta \simeq 2$. In this case we would expect that the sample standard deviations would be needlessly transmitted only for 1 in every 200 blocks.

The adaptive BTC algorithm for each block is implemented as shown in Figure 2. At the transmitter, the sample mean and standard deviation are calculated and the bit mask is formed as specified in the original BTC algorithm. An estimate for the number of looks, $\bar{y}^2/S_y{}^2$, is calculated next and compared to a threshold mN/β. If the local number of looks is less than the threshold, then a flag is set on and a data block is sent which contains the bit mask, sample mean, sample standard deviation, and the flag bit indicating the presence of the standard deviation. If the local number of looks exceeds the threshold then it is highly probable that the SAR model is valid and thus the standard deviation present flag is set off and a block is sent which contains only the bit mask, sample mean, and the flag. At the receiver the flag is tested if it is on the original BTC reconstruction algorithm is applied, i.e. equations (10 and 11), otherwise the modified BTC algorithm (equations 13 and 14) are used to reconstruct the image.

The adaptive BTC algorithm described above automatically adds 1/16 bit/pixel overhead, the standard deviation present flag. It was found that for SEASAT-A SAR images that the quality of the original BTC technique was maintained using the adaptive approach at approximately 1.6 bits/pixel.

The BTC algorithm produces reconstructed images which have a blocky appearance [13] when displayed with magnification. This characteristic also evident to when the BTC algorithm is applied to SAR images. However, the image reconstruction algorithm in the adaptive BTC technique can be modified to remove this blocky appearance.

When the SAR image model, e.g. (2), is valid and only the mean is transmitted we know that a pixels marked with a 1(0) can be modelled by a conditional probability density function, i.e., the p.d.f. given in equation 2 conditioned on the event that the sample is above (below) the mean. Synthetic sampling (using pseudo random numbers) can be used in the reconstruction to map pixels marked with a 1(0) into a gray level based on the appropriate conditional p.d.f. As will be shown in the following section using pseudo random numbers in the reconstruction algorithm does effectively remove the blockness. Unfortunately, generation of pseudo random numbers is computationally intensive, thus this modification increases the computational complexity of the algorithm. In some applications this refinement will not be required.

In the following sections the four compression algorithms, original, modified and adaptive BTC and adaptive BTC with pseudo random reconstruction will be compared.

6. Results

The compression algorithms, BTC, modified BTC, and adaptive BTC and adaptive BTC with pseudo random reconstruction have been applied to SEASAT-A SAR imagery. The purpose of this section is to discuss their performance. As mentioned previously, image quality is difficult to quantify. Here the performance evaluation is based on two criteria: 1) the faithful reproduction of cultural features, and 2) the general appearance of the reconstructed image relative to the original.

The SEASAT-A SAR imagery used here had a resolution of 25 x 25 m with $N \approx 4$. Each pixel intensity was represented by 8 bits (0-255 grey levels). (For more details about the sensor see [15].) For all the compression algorithms described here, the sample mean was coded using 8 bits/block. Also, a 4 x 4 pixel block was used in all cases. In the original BTC algorithm the standard deviation was also coded using 8 bits/block resulting in a 2 bits/pixel data rate.

The modified BTC algorithm resulted in a 1.5 bit/pixel data rate. In the adaptive BTC algorithm, the standard deviation was coded using 7 bits/block, then allowing for the one overhead bit results in a maximum of 32 bits/block. Thus, in the adaptive BTC technique each block is coded into 25 or 32 bits depending on the statistics of the pixel intensities of the block. A threshold of 2.0 was used in all cases. The data rate of the adaptive BTC algorithm is variable. However, in most cases it was about 1.6 bits/pixel.

The response of the first three algorithms to a "point" like target is shown one-dimensionally in Figure 3. Figure 3a represents an intensity profile of 100 pixels from a SEASAT-SAR image. There is one bright feature in the center of this profile. The target-to-background contrast in Figure 3 is about 9dB. Figure 3b is the reconstructed profile using the original BTC technique. The target is still quite evident, the average background level has remained the same as expected and the target level has been reduced. The target-to-background contrast is about 7.5 dB, a loss of 1.5 dB. The result of the modified BTC algorithm is shown in Figure 3c. In this case, the target-to-background loss is about 3.4 dB. This loss is not considered acceptable. The adaptive BTC algorithm, Figure 3d, restored the profile to that given by the original BTC at a small cost in data rate from 1.5 bits/pixel to 1.58 bits/pixel. There is little difference between the original and adaptive BTC results (Figures 3b and 3d), this observation is true for all the results presented here.

A scene containing a variety of terrain features is shown in Figure 4. This scene is composed of 512 x 512 pixels. The three compressed images favorably compare to the original in terms of reproducing the terrain features. However, there is some difference in scale of the texture in the homogeneous regions caused by the block nature of the block coding technique. These differences are more easily seen in the agricultural scene shown in Figure 5. The texture patterns in the reconstructed images appear as speckle patterns. This might be attributed to the compression algorithm's two level quantization of the pixels in each block. So the algorithm is mapping all the up fades to one value and the down fades to another.

The upper left corner of the agricultural scene is shown magnified in Figure 6. At this scale the "blockyness" property [13] of the algorithm is clearly illustrated. However, the shape of most cultural features is preserved. Specifically, the features identified as 1, 2, and 3 on the original SEASAT-A image, Figure 6a, are preserved in shape in all three (6b, c and d) reconstructed images. Note the loss of contrast for the "point" like target (feature #2) between the original BTC (Figure 6b) and the modified BTC (Figure 5c) reconstructed images. The result of the adaptive BTC algorithm using pseudo-random reconstruction is shown in Figure 6e. As expected this refinement reduced theblocky appearance of the reconstructed image. The properties of the BTC algorithm are also evident in Figure 7. This scene contains a water body, a dam, and a power transmission line (the row of bright points near the bottom of the scene). Again, the shape of these features is preserved and the modified BTC algorithm shows a loss of contrast for the point targets.

7. Conclusions

A data compression technique has been developed for SAR images. The method was tested on SEASAT-A SAR data and found to produce images with a suitable quality for a variety of applications. The algorithm developed here is an extension of the BTC technique developed in [7]. The specific statistical properties of SAR data were used to improve the data compression ratio. A compression ratio of 5 to 1 (data rate of 1.6 bits/pixel) was obtained. Further minor reductions in data rate might be possible by reducing the number of bits used to represent the sample mean and standard deviation as suggested in [18]. The benefit of such a reduction would have to be considered based on the application of the sensor. Also, further study is needed to evaluate the effect of changing the number of looks of the SAR on the quality of the reconstructed images.

The technique presented here requires no large data storage as opposed to transform coding methods [1], and is computationally simple so that a single chip implementation is possible [7]. As the SAR image formation moves closer to a real time operation and is thus performed on the sensor, data compression techniques as the one presented here will provide the system designer with additional trade-offs for transmitting and storing the data.

Acknowledgment

The author would like to thank Dr. K. S. Shanmugan for his helpful critism of this work. Also the help of Ellie Watson and Dave Boberg for processing the images is recognized.

References

[1] B. G. Kashef and K. K. Tam, "Synetic Aperture Radar Image Bandwidth Compression," in Proceedings of the SPIE, Vol. 432, Aug. 1983, pp. 45-53.

[2] R. G. Lipes and S. A. Butman, "Bandwidth Compression of Synethic Aperture Radar Imagery by Quantization of Raw Radar Data," in Proceedings of the SPIE, Vol. 119, 1977, pp. 107-114.

[3] C. Wu, "Considerations on Data Compression of Synethic Aperture Radar Images," in Proceedings of the SPIE, Vol. 87, 1976, pp. 134-140.

[4] K. S. Shanmugan, et al., "Textural Features for Radar Image Analysis," IEEE Trans. on Geoscience and Remote Sensing, Vol. GE-19, No. 3, July 1981, pp. 153-156.

[5] Moore, R. K., "Tradeoff between picture element dimensions and noncoherent averaging in side-looking airborne radar," IEEE Transactions on Aerospace and Electronics Systems, AES-15, Vol. 5, September 1979, pp. 697-708.

[6] A. K. Jain, "Image Data Compression: A Review," IEEE Proc., Vol. 69, No. 3, March 1981, pp. 349-384.

[7] E. J. Delp and O. R. Mitchell, "Image Compression Using Block Truncation Coding," IEEE Trans. on Communication, Vol. COM-27, No. 9, Sept. 1979, pp. 1335-1342.

[8] G. R. Arce and N. C. Gallaghen, "BTC Image Coding Using Median Filter Roots," IEEE Trans. on Communications, Vol. Com-31, No. 6, June 1984, pp. 784-793.

[9] Porcello, L. J., Massey, N. G., Innes, R. B., and Marks, J. M., "Spreckle reduction in synthetic aperture radars," Journal Optical Society of America, Vol. 66, No. 11, Nov. 1976, pp. 1305-1311.

[10] Meyer, D. P. and Mayer, H. A., Radar Target Detection, New York: Academic Press, 1973, pp. 36-65.

[11] Mitchell, R. L., Models of extended targets and their coherent radar images, Proceedings of the IEEE, Vol. 62, No. 6, June 1974, pp. 754-758.

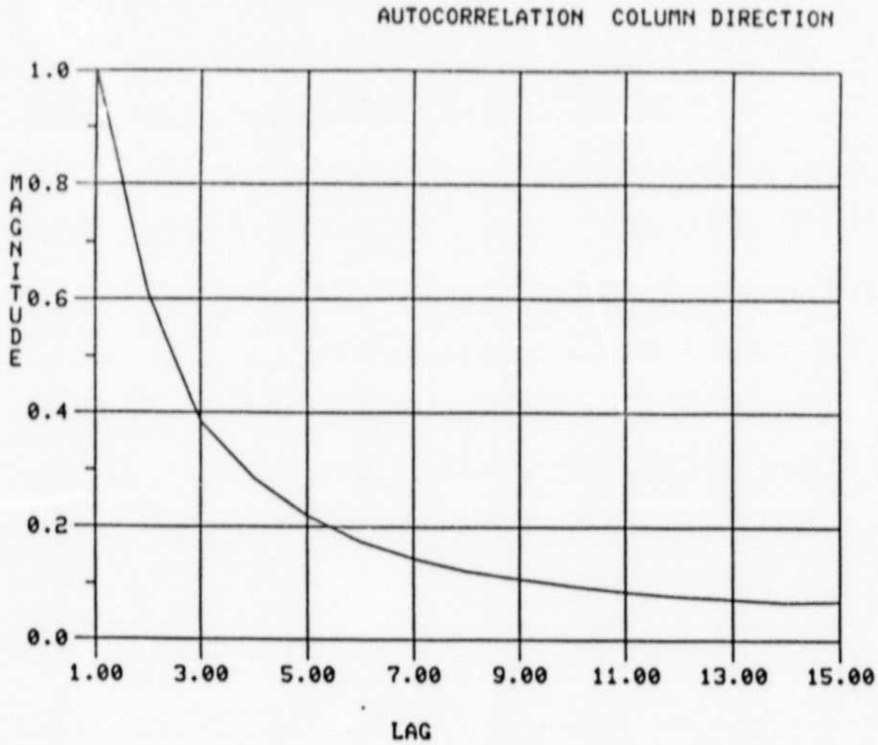
[12] Lee, J. S., Speckle analysis and smoothing of synthetic aperture radar images, Computer Graphics and Image Processing, Vol. 1, Dec. 1981, pp. 17-32.

[13] A. Rosenfeld andA. Kak, Digital Picture Processing, Academic Press, 1982.

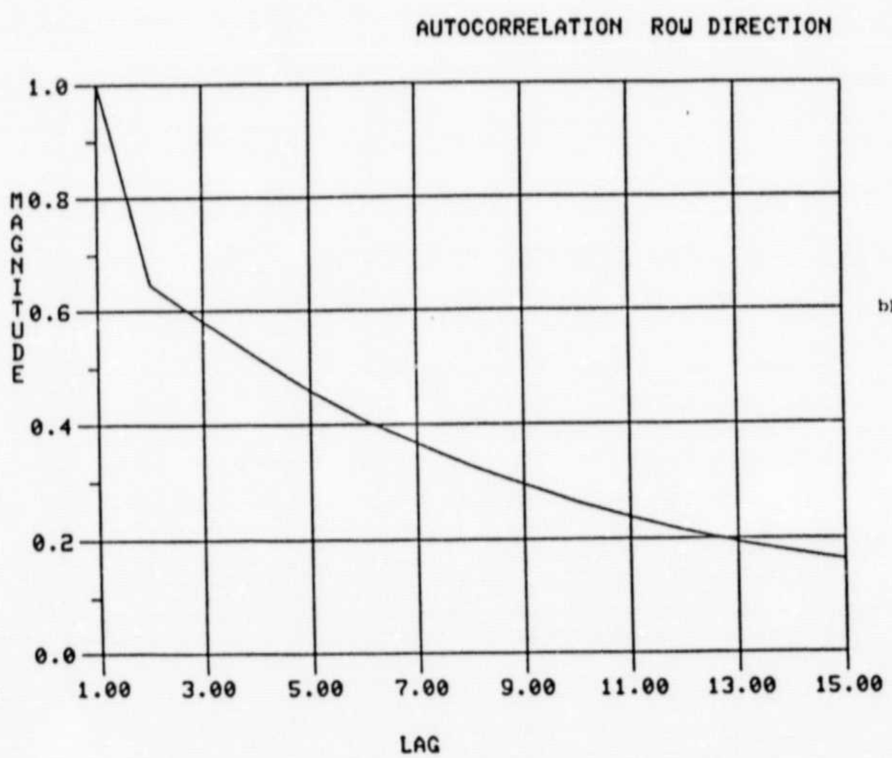
[14] R. V. Hogg and A. T. Craig, Introduction to Mathematical Statistics, MacMillan Co., London, 1970.

[15] Jordan, R. L., The SEASAT-A synthetic aperture radar system, IEEE Journal of Oceanic Engineering, Vol. OE-2, No. 2, April 1980, pp. 154-165.

List of Figures



a)



b)

Figure 1. SAR Image Autocorrelation Function of Terrain
(Scene shown in Figure 4)

```
SendAdaptiveBTC( InputImage, Threshold )
   DO for each row of blocks
      DO for each column of blocks
         Compute BlockMean and BlockStandardDeviation
         Send( BlockMean )
         IF ( (BlockMean / BlockStandardDeviation) < Threshold )
            THEN Send( 1 )
                 Send( BlockStandardDeviation )
            ELSE Send( 0 )
         DO for each row of block
            DO for each column of block
               IF ( Pixel > BlockMean )
                  THEN Send( 1 )
                  ELSE Send( 0 )
            OD
         OD
      OD
   OD
```

```
ReceiveAdaptiveBTC
   DO for each row of blocks
      DO for each column of blocks
         Receive( BlockMean )
         Receive( BlockSDFlag )
         IF ( BlockSDFlag = 1 )
            THEN Receive( BlockStandardDeviation )
            ELSE BlockStandardDeviation := BlockMean / sqrt( N )
         A := BlockMean - BlockStandardDeviation * sqrt( q / m - q )
         B := BlockMean - BlockStandardDeviation * sqrt( q / m - q )
         DO for each row of pixels
            DO for each column of pixels
               Receive( PixelFlag )
               IF ( PixelFlag = 0 )
                  THEN Pixel := A
                  ELSE Pixel := B
            OD
         OD
      OD
   OD
```

Figure 2a. Adaptive BTC Algorithm

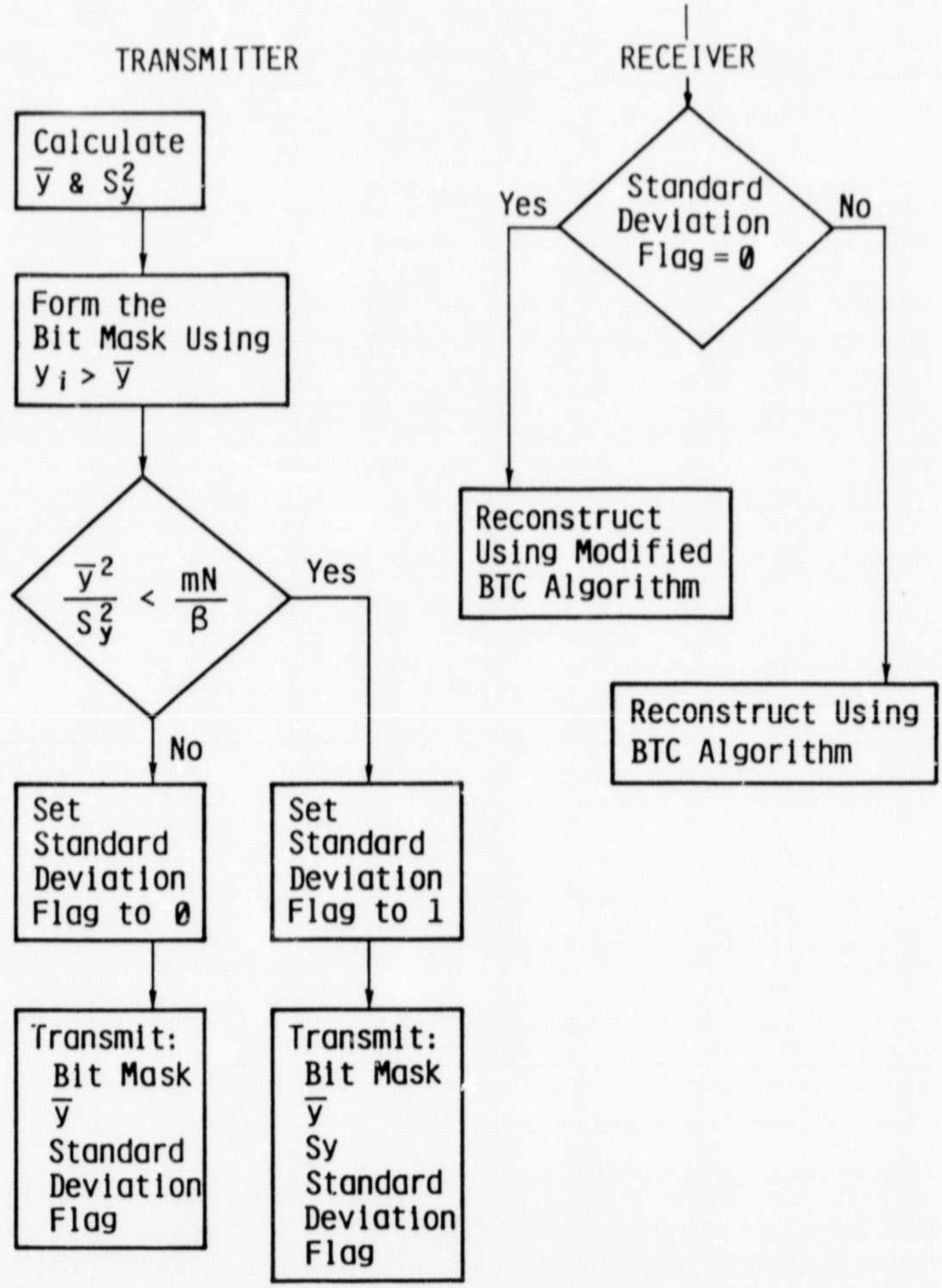


Figure 2b. Adaptive BTC Flow Diagram

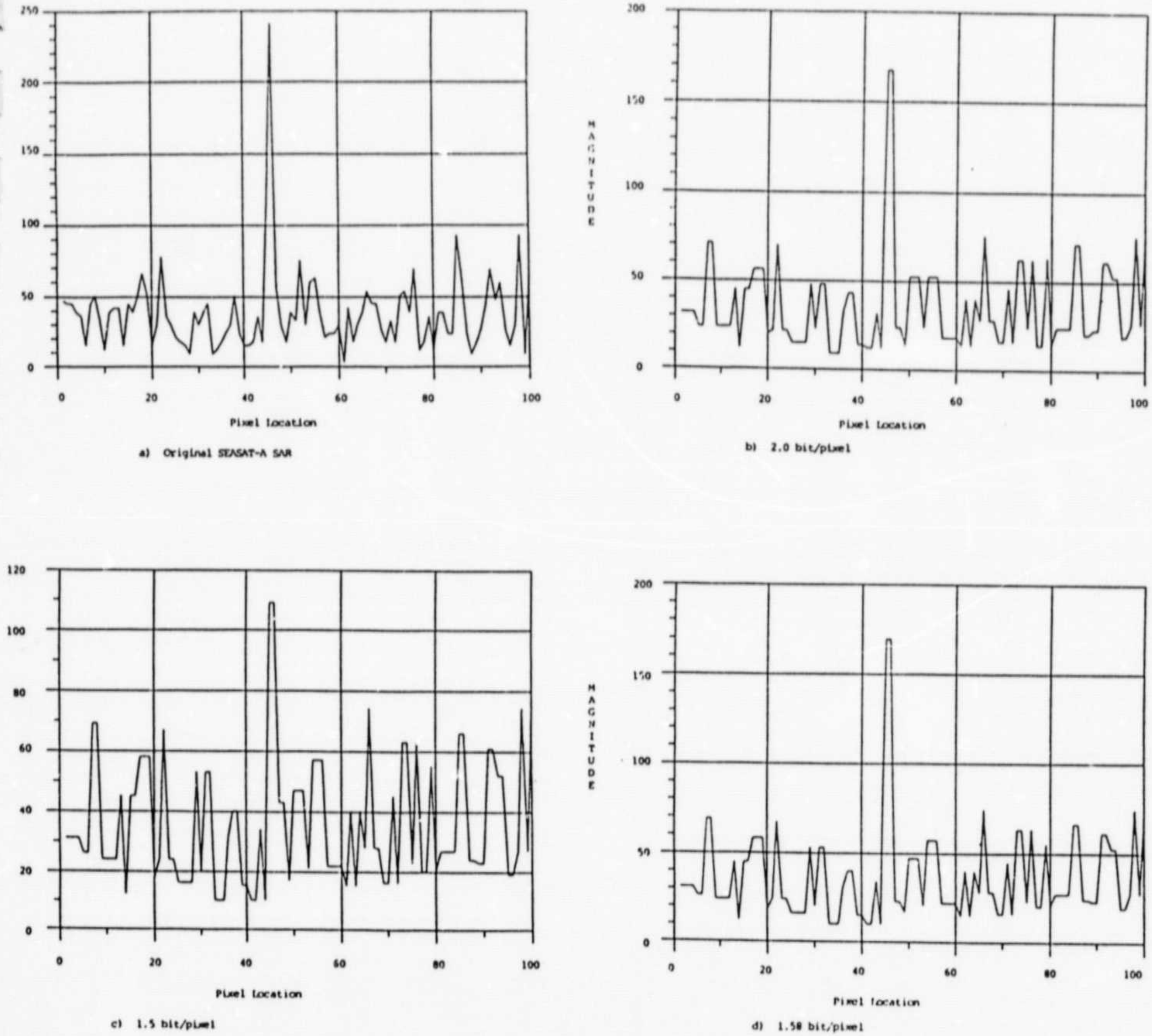
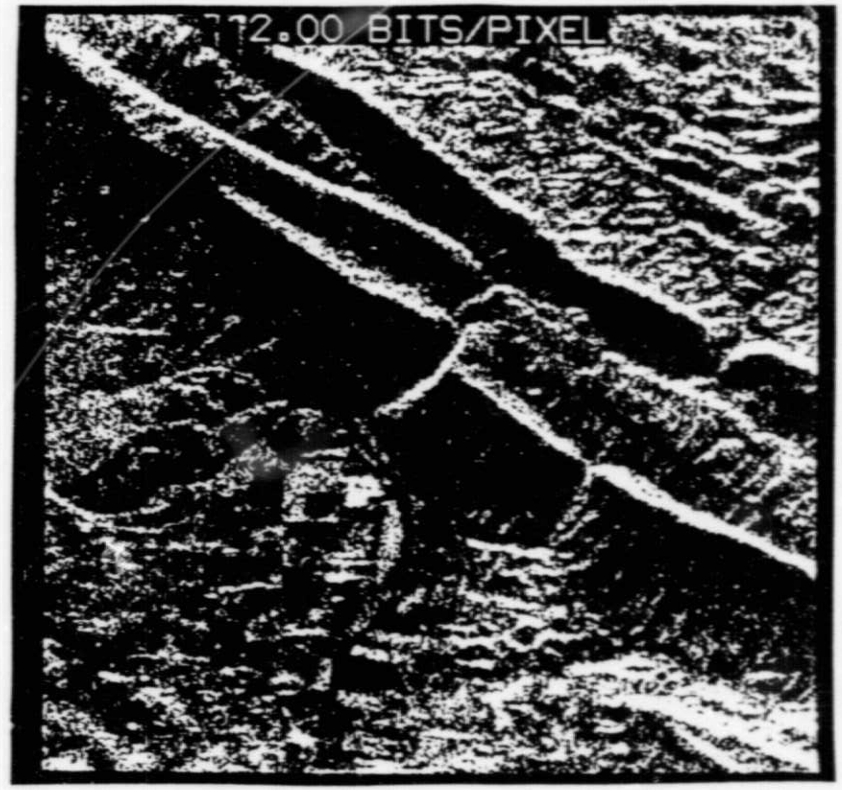


Figure 3. Cultural Feature Response of the Compression Algorithms
a) original data; b) BTC; c) modified BTC; d) adaptive BTC

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

a.

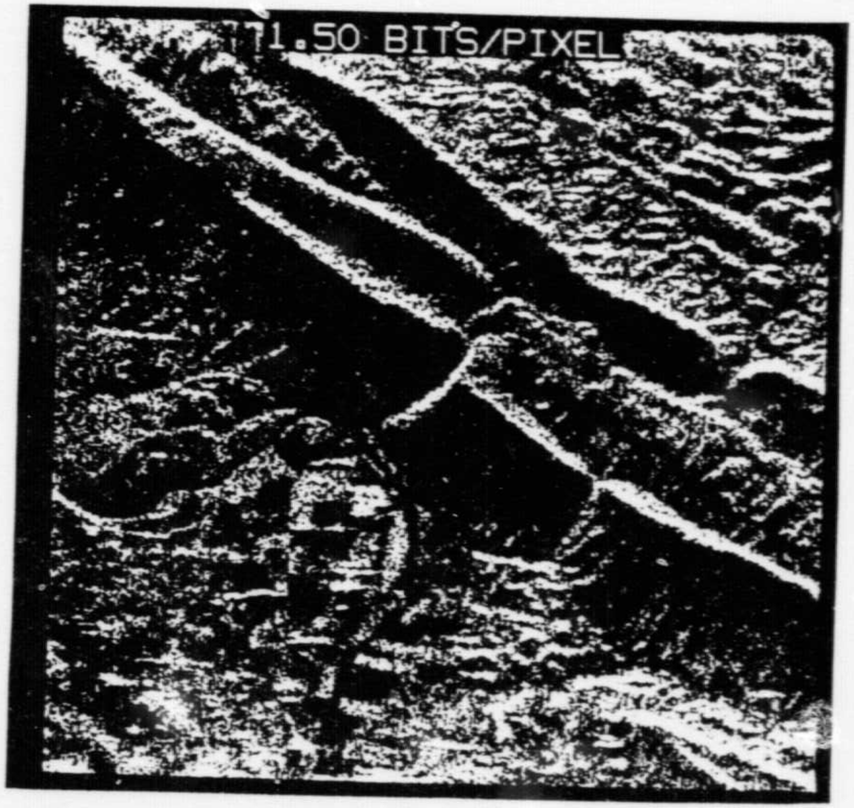


b.

Figure 4.

Result of Coding for a Scene Containing Elevation

a) original, b) BTC, c) modified BTC, d) adaptive BTC



c.

d.

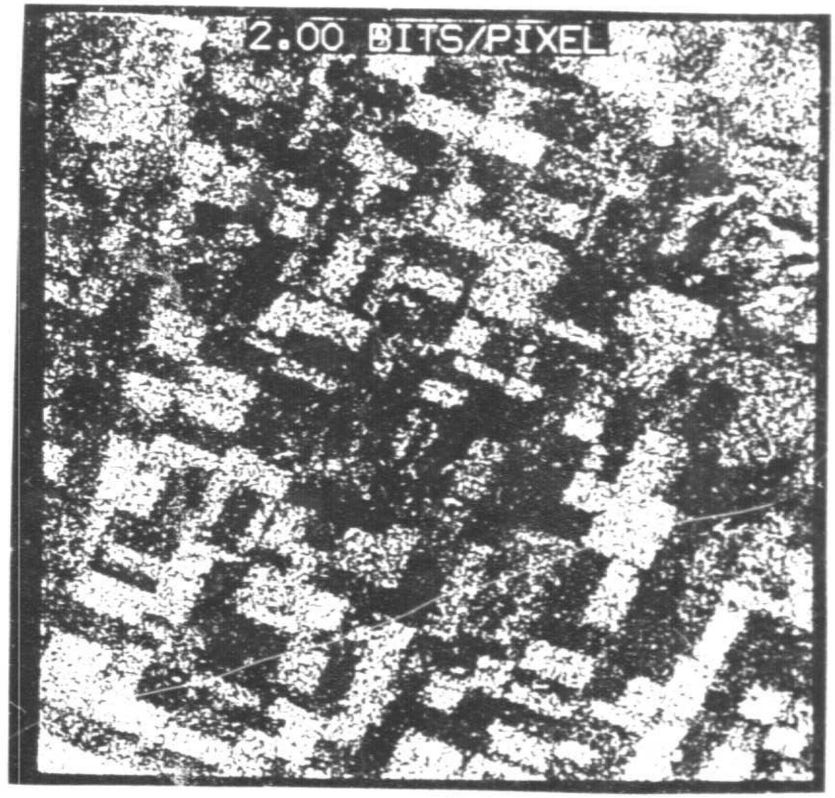
Figure 4.

Result of Coding for a Scene Containing Elevation

a) original, b) BTC, c) modified BTC, d) adaptive BTC

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

a.



b.

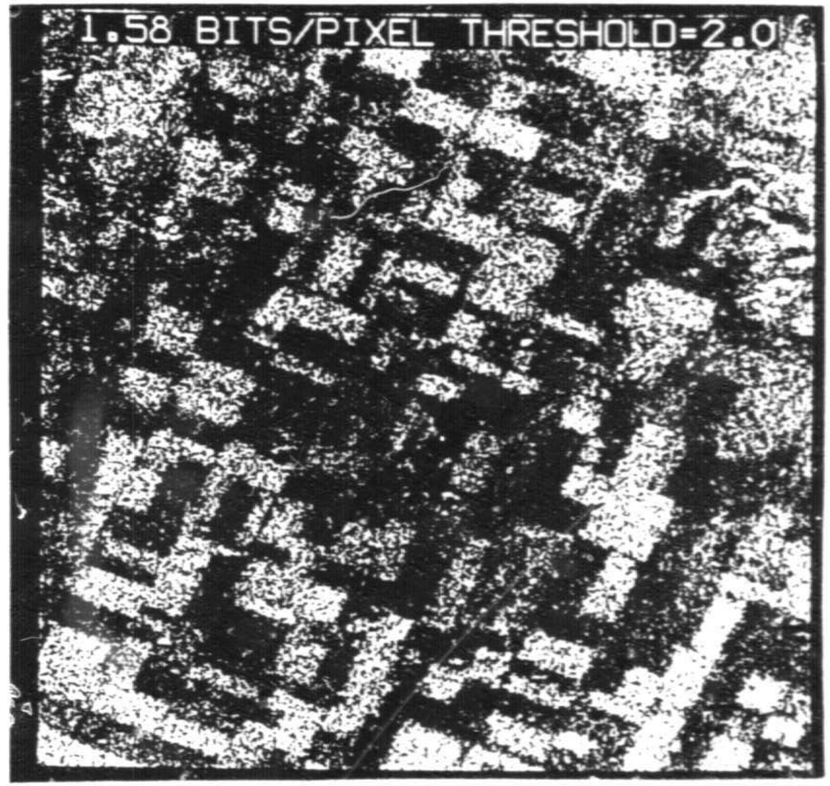
Figure 5.

Results of Coding for an Agricultural Scene

a) original, b) BTC, c) modified BTC, d) adaptive BTC

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

c.

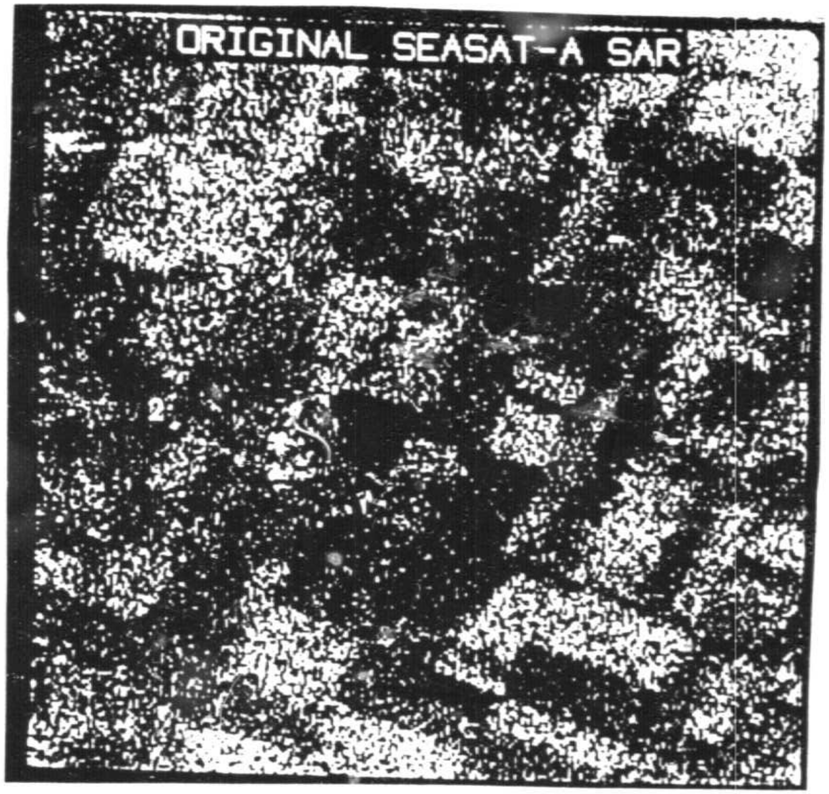


d.

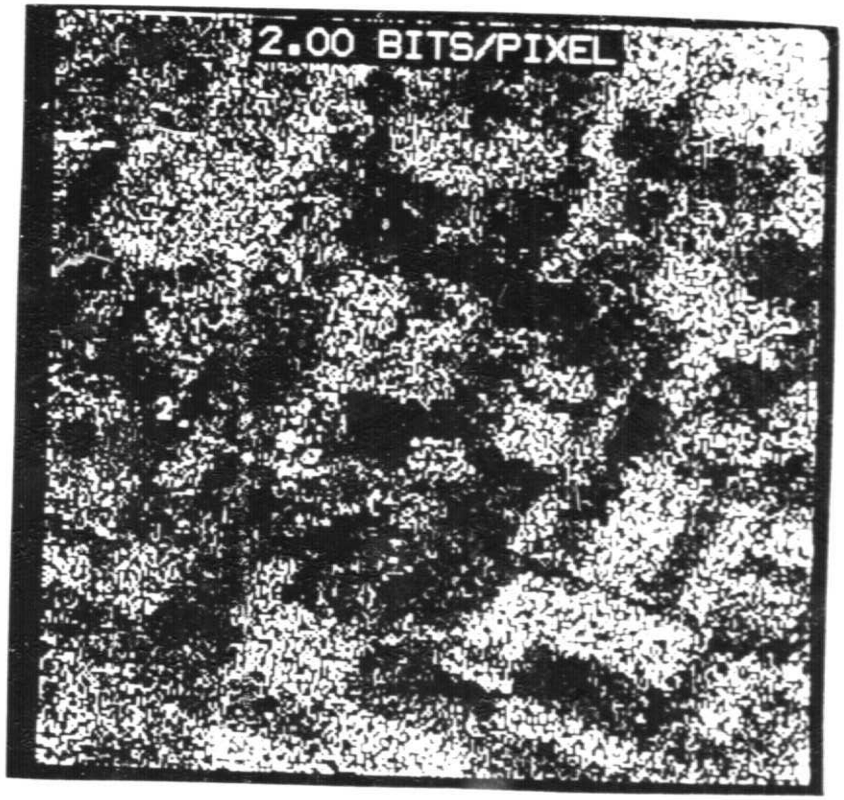
Figure 5.

Results of Coding for an Agricultural Scene

a) original, b) BTC, c) modified BTC, d) adaptive BTC



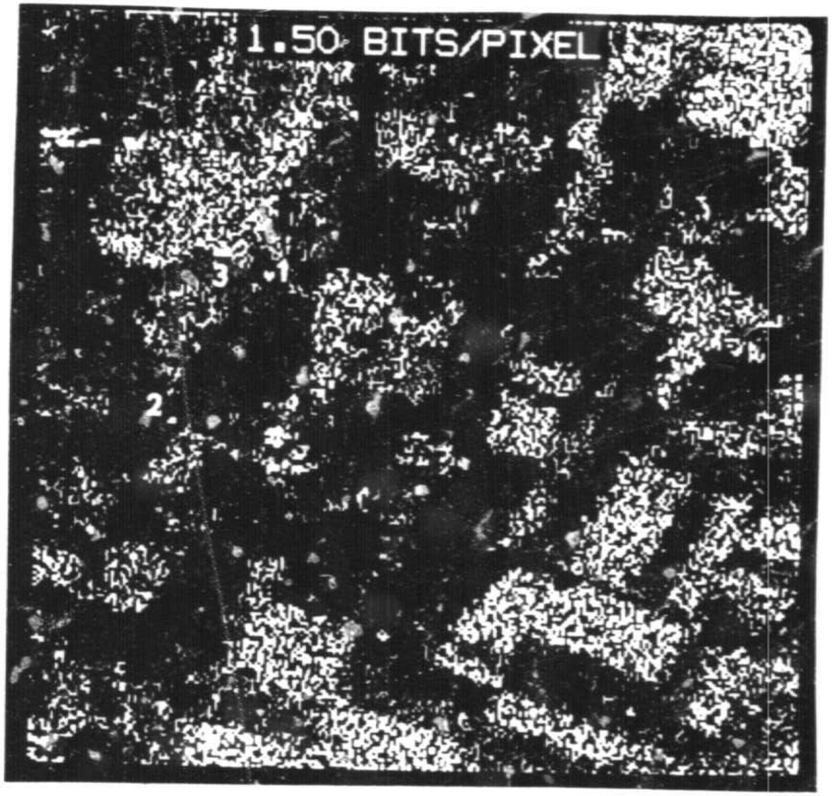
a.



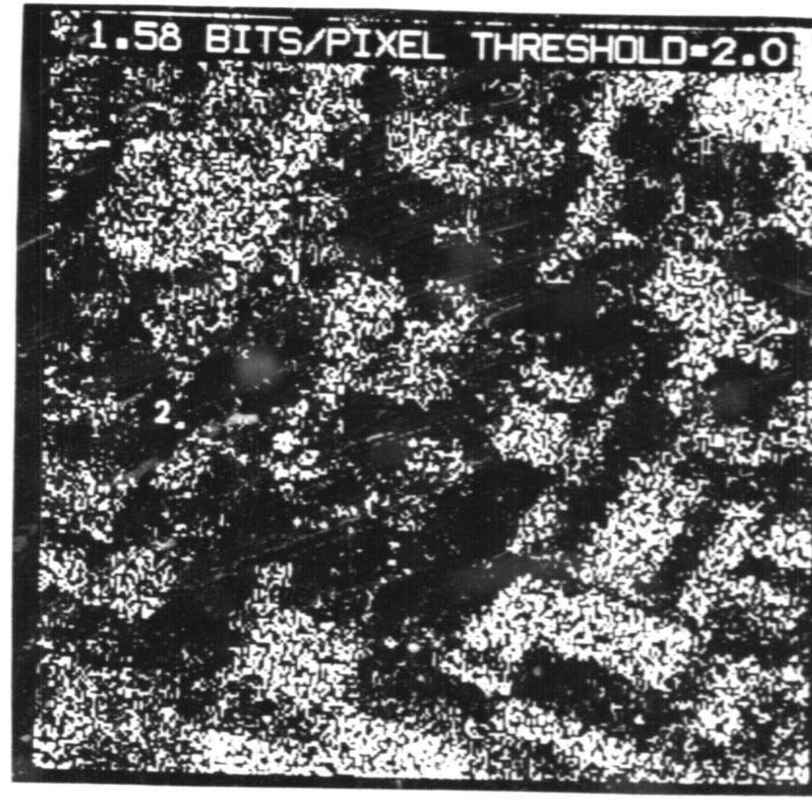
b.

Figure 6.

Results of Coding for an Agricultural Scene: Magnified a) original, b) BTC, c) modified BTC, d) adaptive BTC and e) adaptive BTC with pseudo-random reconstruction



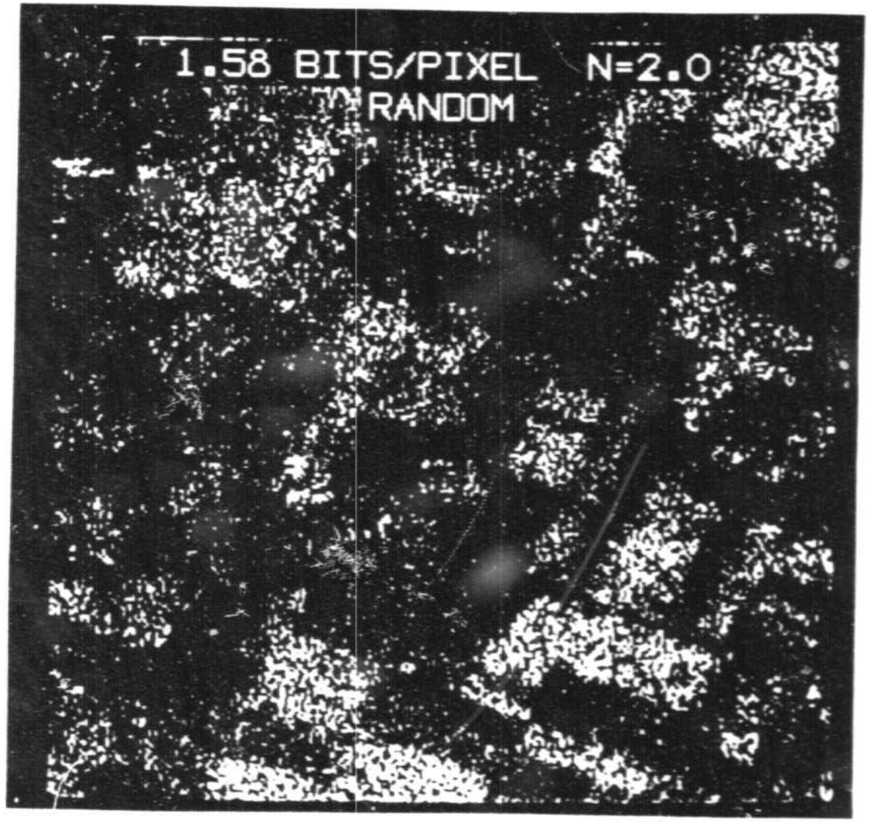
c.



d.

Figure 6.

Results of Coding for an Agricultural Scene: Magnified a) original, b) BTC, c) modified BTC, d) adaptive BTC and e) adaptive BTC with pseudo-random reconstruction



e.

Figure 6.

Results of Coding for an Agricultural Scene: Magnified a) original, b) BTC, c) modified BTC, d) adaptive BTC and e) adaptive BTC with pseudo-random reconstruction

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

a.

b.

Figure 7.

Results of Coding for General Scene

a) original, b) BTC, c) modified BTC, d) adaptive BTC

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

c.

d.

Figure 7.

Results of Coding for General Scene

a) original, b) BTC, c) modified BTC, d) adaptive BTC

APPENDIX D

An Optimal Frequency Domain Textural Edge Detection Filter

An Optimal Frequency Domain Textural Edge Detection Filter

Victor S. Frost
Assistant Professor

K. Sam Shanmugan
Professor

J. Keith Townsend
Research Assistant

Telecommunications and Information Sciences Laboratory
University of Kansas
Space Technology Center
Nichols Hall
2291 Irving Hill Road
Lawrence, Kansas 66045
(913) 864-4832

This work was supported by NASA under Contracts NADA 9-16664 and NAGW-381.

ABSTRACT

An optimal frequency domain textural edge detection filter is developed and its performance evaluated. For the given one-dimensional texture model, and filter bandwidth, the filter maximizes the amount of output image energy placed within a given resolution interval centered on the textural edge. Filter derivation is based on relating textural edge detection to tonal edge detection via the complex lowpass equivalent representation of narrowband bandpass signals and systems. The filter is specified in terms of translated-in-frequency prolate spheriodal wave functions. Performance is evaluated using the asymptotic approximation version of the filter. This evaluation demonstrates satisfactory filter performance for ideal and non-ideal textures. In addition, the filter can be adjusted to detect textural edges in noisy images at the expense of edge resolution.

I. INTRODUCTION AND OVERVIEW

Edge detection is an important first step in extracting information from an image. Many edge detection schemes have been employed to enhance the boundaries between regions of different average gray tone. These tonal edge detectors are inadequate when regions in an image are characterized by similar average gray tone, but different textural features.

A textural edge detection filter is presented in this paper which is optimal in the sense that, for the given model, a maximum amount of output image energy is placed within a given resolution interval width and a given filter bandwidth. The resolution interval is centered on the textural edge in the input image. The filter is derived in the frequency domain, and is easily implemented on a digital computer using Fast Fourier Transform (FFT) techniques.

The optimum textural edge detection filter is developed by treating the textural edge as a bandpass extension of a tonal edge. Hence, the optimum tonal edge detector derived by Shanmugan, Dickey and Green [1] (correspondence by Lunscher [2]), is related to the textural edge detection case via the complex lowpass equivalent representation of signals and systems. It should be pointed out that the development is carried out in one-dimension. However, symmetries required for extension to two-dimensions are retained.

Section II presents a brief review of the optimum tonal edge detector. The textural model used in the development of the optimum textural edge detector is then introduced in Section III. The mathematical form of the optimum textural edge detection filter and some one-dimensional examples are presented in Section IV. Concluding remarks are given in Section V.

II. REVIEW OF THE OPTIMUM TONAL EDGE DETECTOR

Ths purpose of this section is to briefly review the optimum tonal edge detector derived by Shanmugan, et al., [1]. For a given filter bandwidth, the optimum tonal edge detector places a maximum amount of output image energy within a given resolution interval length in the vicinity of tonal edges. The tonal edge detector is insensitive to textural edges where the average gray levels of the different textural regions are equal.

The derivation of the optimum tonal edge detector is based on representing the filter output (for a step edge input) in terms of prolate spheriodal wave functions (for the derivation, see [1], [2]). The exact one-dimensional form of the filter transfer function is given in Shanmugan, et al., [1] as

$$H_{STEP,E}(\omega) = \begin{cases} B_1 \omega\ \psi_1(c,\ \omega I/2\Omega), & |\omega| < \Omega \\ 0 & \text{elsewhere} \end{cases} \tag{1}$$

where $c = \frac{\Omega I}{2}$ and ψ_1 is the first order prolate spheriodal wave function. (The subscript STEP,E in Equation (1) denotes the Exact form of the STEP edge detector). For any given values of spatial bandwidth, Ω, and resolution interval length, I, the transfer function in Equation (1) places the maximum amount of energy in I. The filter is difficult to implement in this form, because the values of ψ_1 are tabulated. Application of approximations by Slepian and Streifer [1], yield the asymptotic approximation of the filter, which is in closed form, hence easy to implement. The resulting expression is

$$H_{STEP,E}(\omega) \cong H_{STEP}(\omega) = K_1\ \omega^2\ \exp(-\frac{c\omega^2}{2\Omega^2}) \tag{2}$$

Combining the constants that appear in the argument of the exponent, and dropping the gain factor, K_1, yields

$$H_{STEP}(\omega) = \omega(\omega e^{-K\omega^2}) = \omega^2\ e^{-K\omega^2} \tag{3}$$

It should be noted that the parameters I and Ω can no longer be independently specified.

Choice of K sets the bandwidth of the filter, and also the resolution interval length. As K increases, resolution interval size increases, and filter bandwidth decreases. Note that even though the asymptotic approximation to the optimum transfer function is not strictly bandlimited, $H_{STEP}(\omega)$ is effectively zero for spatial frequencies above a certain value, depending on the choice of K. The asymptotic approximation will be used in the remainder of the development.

III. TEXTURAL MODEL

One inherent difficulty with textural processing is the fact that no single "best" model exists for characterizing texture in images. The model used here in the development of the optimum textural edge detector capitalizes on the relationship between texture and spatial frequency by representing each texture as a sinusoid of different spatial frequency (i.e., fine textures contain greater concentrations of energy at higher spatial frequencies than coarser textures do) [3], [4], [5], [6], [7], [8], [9].

In general, a class of one-dimensional images with n textures can be defined as

$$q(x) = A(x) \cos(\omega_i x + \Theta(x)) \qquad i = 1, 2, \cdots, n \tag{4}$$

where

$$A(x) = a(1 + \alpha(x)) \qquad |a(x)| < 1 \tag{5a}$$

and

$$\Theta(x) = b \int_{-\infty}^{x} \beta(\lambda)\, d\lambda \tag{5b}$$

The functions $\alpha(x)$ and $\beta(x)$ are random processes, ω_1 represents the ith texture, a and b are constants, and x is the spatial variable. Note that q(x) is allowed to be negative. This can be viewed as subtracting off the mean level from an image, thus allowing negative brightness or gray level. In this model, $\alpha(x)$ represents average gray level, and $\beta(x)$ represents the variation of spatial frequency within a texture. In other words, the envelope of q(x) can be thought of as the average gray level variation, while the underlying texture is represented by each different ω_i, where the random change of texture for a given ω_i is controlled by $\beta(x)$. Note that if time were the independent variable, q(x) would be a double sideband plus large carrier modulated waveform, with simultaneous frequency modulation.

An ideal texture is represented in this model by a sinusoid with constant spatial frequency and constant amplitude. Hence, a transition between two ideal textures can be represented by a pure sinusoid at one spatial frequency followed by a pure sinusoid at another spatial frequency. For the ideal two texture case let

$A(x) = 1$

$O(x) = 0$

$-\infty < x < \infty$ (infinite size)

Thus, an image with two ideal textures and a textural edge at $x = 0$ is represented mathematically as

$$f(x) = \cos(\omega_i x), \qquad -\infty < x < \infty \tag{6}$$

where

$i = 1$ for $x < 0$ and

$i = 2$ for $x \geq 0$.

The optimum textural edge detector is derived using the ideal, two texture image, $f(x)$.

IV. OPTIMUM TEXTURAL EDGE DETECTOR RESULTS AND PERFORMANCE

This section presents the mathematical form of the optimum textural edge detection filter and discusses the performance of the filter for several different classes of input images. The derivation is only briefly sketched here, the details are given in Townsend [10].

For a two texture input image with one texture represented by a sinusoid with frequency ω_1, and the other texture represented by a sinusoid with frequency ω_2, the transfer function of the optimum tonal edge detector is given by

$$H_{OPT}(\omega) = H_1(\omega) + H_2(\omega) \tag{7}$$

where

$$H_1(\omega) = H_{STEP}(\omega - \omega_1) + H_{STEP}(\omega + \omega_1) \tag{8a}$$

$$H_2(\omega) = H_{STEP}(\omega - \omega_2) + H_{STEP}(\omega + \omega_2) \tag{8b}$$

and

$$H_{STEP}(\omega) = \omega^2 e^{-K\omega^2} \tag{3}$$

It is clear from Equations (7), (8), and (3), that the optimum textural edge detector is the sum of the responses of two bandpass "sub" filters, $H_1(\omega)$ and $H_2(\omega)$. Each "sub" filter is a translated-in-frequency version of the optimum tonal edge detector, $H_{STEP}(\omega)$, discussed in Section II. Note that $H_{STEP}(\omega)$ is translated to each of the two textural frequencies.

The optimum textural edge detector is derived by recognizing that the two-ideal-texture input image, $f(x)$, given in Section III can be expressed as the sum of two truncated sinusoids, one at frequency ω_1, defined for $-\infty < x < 0$ and the other at frequency ω_2, defined for $0 \leq x < +\infty$. But each of these two truncated sinusoids are bandpass at frequencies ω_1 and ω_2 respectively. Each truncated sinusoid has a step function for its complex lowpass equivalent [11]. Because $H_{STEP}(\omega)$ is optimized for detecting step type edges, a bandpass version of $H_{STEP}(\omega)$ centered on frequency ω_1 is optimum for detecting the discontinuity (modulated step function), in the truncated sinusoid at frequency ω_1 [10]. Similarly, a bandpass version of $H_{STEP}(\omega)$ translated in frequency to ω_2 is optimum for detecting the discontinuit[illegible] the truncated sinusoid at frequency ω_2. The sum of the outputs of these two [illegible]andpass filters produces the optimized output. A block diagram of the filter structure for the two texture case is shown in Figure 1.

A qualitative discussion is presented here to gain insight into how the filter works. Figure 2 presents an example of the optimum textural edge detector in the frequency domain. Note from the figure that the response at ω_1 and ω_2 (the spatial frequencies representing the two ideal textures) is zero. Hence, $H_{OPT}(\omega)$ does not respond to any input which has spectral energy only at these two frequencies. Therefore, the response to an input representing either pure texture (in steady state) is zero. The textural edge is characterized by a transition from one texture to the other. The Fourier transform of this boundary contains spectral energy at frequencies other than ω_1 and ω_2. In particular, there is energy in the passband portions of $H_{OPT}(\omega)$, therefore filter response near the textural edge is non-zero resulting in a large amount of output image energy in the vicinity of the textural edge.

The Fourier transform of the entire input image is given by

$$F(\omega) = F_1(\omega) + F_2(\omega) \tag{9}$$

where $F_1(\omega)$ amd $F_2(\omega)$ are the Fourier transforms of the truncated textures represent[illegible] by sinusoids at ω_1 and ω_2 respectively. Multiplication of $F(\omega)$ with $H_{OPT}(\omega)$ yields the transform of the output, $G(\omega)$, i.e.,

$$G(\omega) = F(\omega)\ H_{OPT}(\omega) \tag{10}$$

but this is equivalent to

$$\begin{aligned} G(\omega) &= [F_1(\omega) + F_2(\omega)]\ [H_1(\omega) + H_2(\omega)] \\ &= F_1(\omega)\ H_1(\omega) + F_1(\omega)\ H_2(\omega) + F_2(\omega)\ H_1(\omega) + F_2(\omega)\ H_2(\omega) \end{aligned} \tag{11}$$

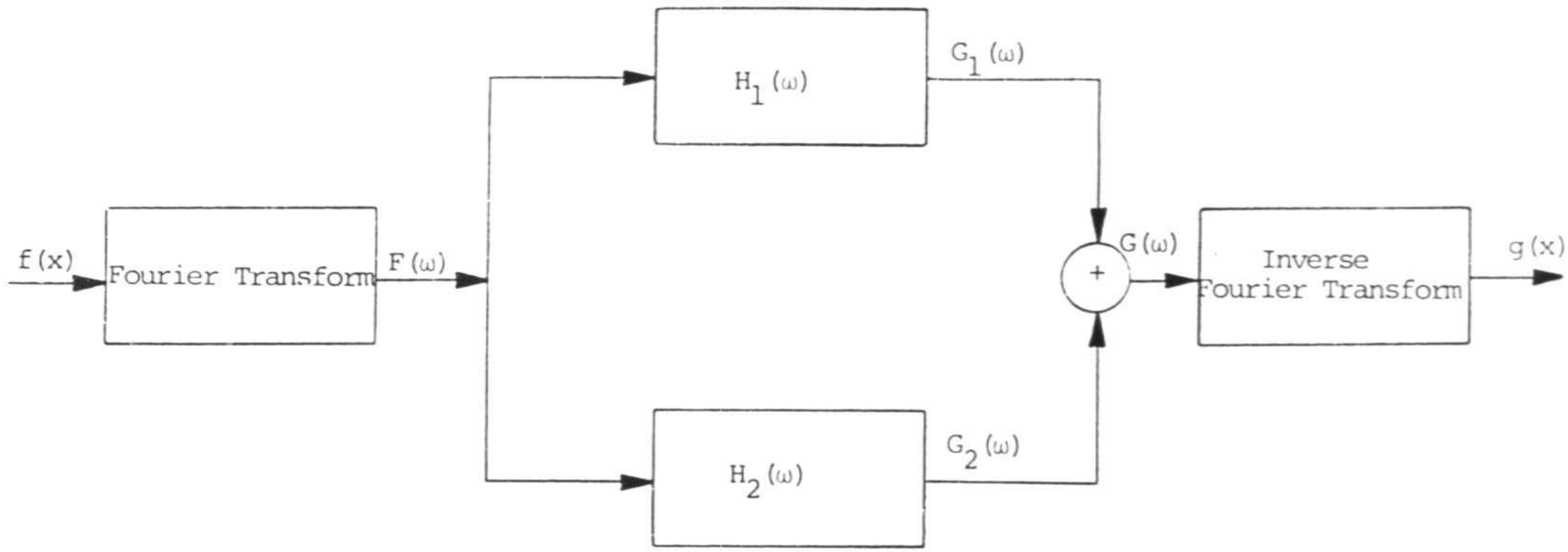


Figure 1   Block diagram of the optimum textural edge detection filter for two textures.

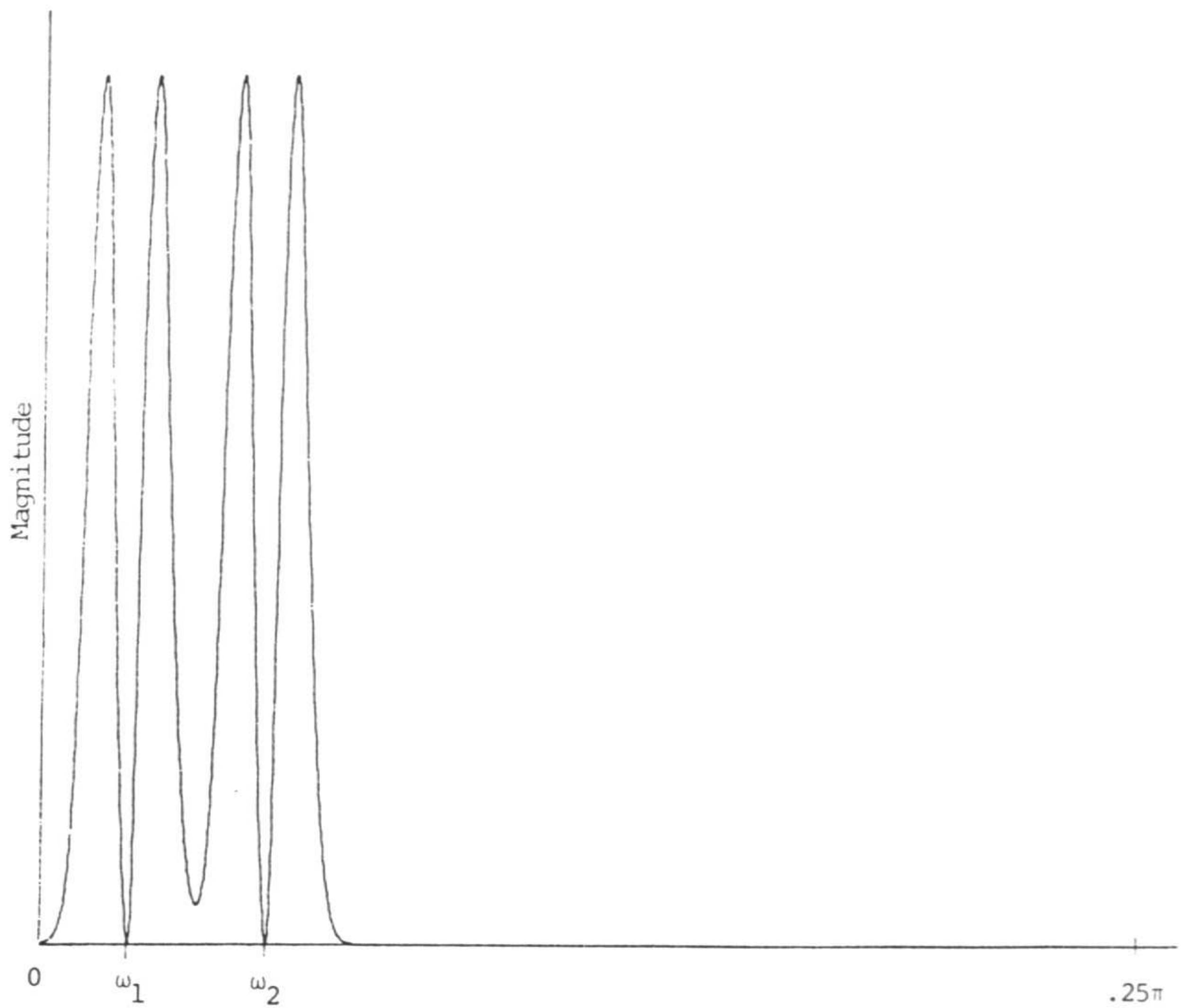


Figure 2 Single sided transfer function of the optimum textural edge detection filter. The bandwidths of $H_1(\omega)$ and $H_2(\omega)$ are narrow enough that response at ω_1 and ω_2 is zero.

but

$$F_1(\omega)\ H_2(\omega) \cong 0 \tag{12}$$

and

$$F_2(\omega)\ H_1(\omega) \cong 0 \tag{13}$$

Substitution of Equations (12) and (13) into Equation (11) yields

$$G(\omega) = F_1(\omega)\ H_1(\omega) + F_2(\omega)\ H_2(\omega)$$

$$= G_1(\omega) + G_2(\omega) \tag{14}$$

Hence,

$$g(x) = g_1(x) + g_2(x) \tag{15}$$

Equations (12) and (13) are true because of the spectral separation between the two sets of bandpass inputs and systems. In non-ideal texture cases, there can be considerable spectral overlap between the Fourier transforms of the textures. The spectral overlap can cause non-zero response of a system, $H_1(\omega)$, for example, to a textxure not centered at ω_1, $F_2(\omega)$ for example. This could also occur if the bandpass bandwidth of $H_1(\omega)$ is wide enough to pass a significant amount of energy due to $F_2(\omega)$.

Choosing the exponential parameter, K, such that the bandpass bandwidths of $H_1(\omega)$ and $H_2(\omega)$ are wider than the spatial frequency separation between ω_1 and ω_2 results in non-zero response to the two textures. There is improved resolution at the expense of an increase in the "background" level in the output image, thus decreasing edge visibility. The "background" refers to the out-of-resolution-interval gray level. Edge visibility describes the difference in gray level between the in-resolution-interval and out-of-resolution-interval (background) portions of the output image. The spatial frequency separation of the textures affects the performance of the filter, i.e., the greater the separation, the better the performance.

It was shown in Shanmugan, et al., [1] that the optimum tonal edge detector could be used to enhance tonal edges in images corrupted by additive white Gaussian noise. The same theory applies to the optimum textural edge detector. The exponential parameter, K, can be chosen to decrease the bandwidth of the "sub" filters to decrease the effects of the noise. The price paid for this is an increase in the resolution interval length [10]. The benefits of increased edge visibility may more than offset the decrease in resolution.

Figure 3 shows the result of implementing the filter on a digital computer. Displayed are the input and output images (one-dimensional) of the optimum textural edge detection filter for an input with two ideal textures (one textural edge). The textural edge is clearly marked in the output image.

The transfer function, $H_{OPT}(\omega)$, can be generalized to n textures by simply adding more translated-in-frequency versions of $H_{STEP}(\omega)$. Denote the generalized, n texture transfer function as $H_{OPT,n}(\omega)$, defined as

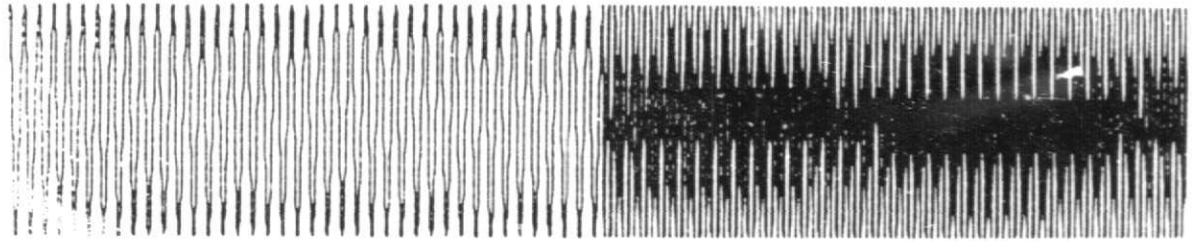
$$H_{OPT,n}(\omega) = \sum_{i=1}^{n} H_i(\omega) \tag{16}$$

where

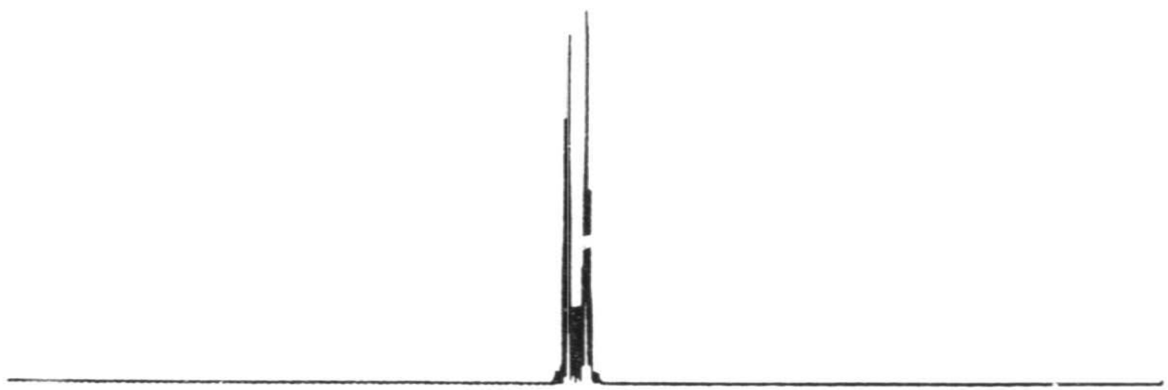
$$H_i(\omega) = H_{STEP}(\omega - \omega_i) + H_{STEP}(\omega + \omega_i) \tag{17}$$

and ω_i represents the frequency of the ith texture. Each of the n filters respond to transient energy where textural transitions occur but null out response to the ith texture in steady state. An example of a one-dimensional output image for an input image containing four ideal textures with three textural edges is shown in Figure 4. The normalized frequencies of the four different textures in the figure are $.04\pi$, $.06\pi$, $.08\pi$, and $.1\pi$, with each texture occurring once in the input image.

It should be pointed out that although each of the "sub" filters (i.e., $H_1(\omega)$, $H_2(\omega)$, $\cdots$) are narrowband bandpass about the respective textural frequencies, the overall system bandwidth and image bandwidth are about equal, as shown in Figure 5. The total textural edge detector bandwidth, BW, is written in terms of the tonal edge detector bandwidth as follows:



(a)



(b)

Figure 3 (a) Input image consisting of two ideal textures.
(b) Magnitude of the optimum textural edge detector response (in the spatial domain).

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

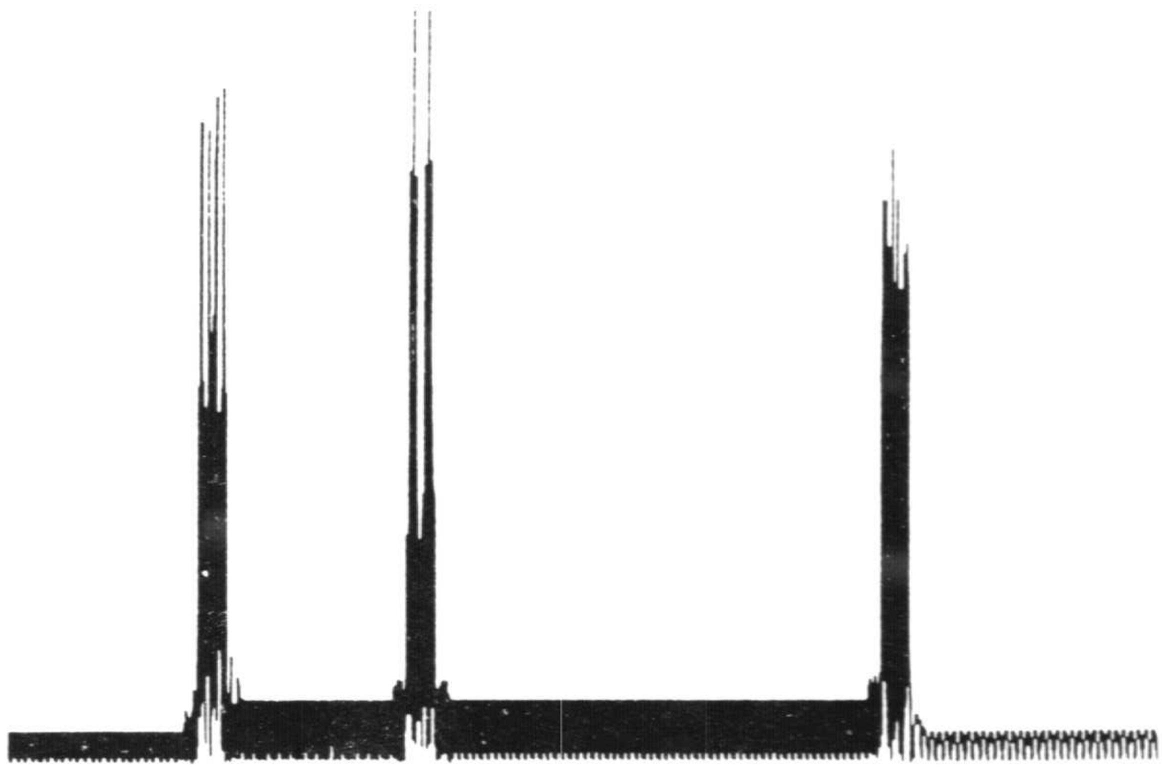
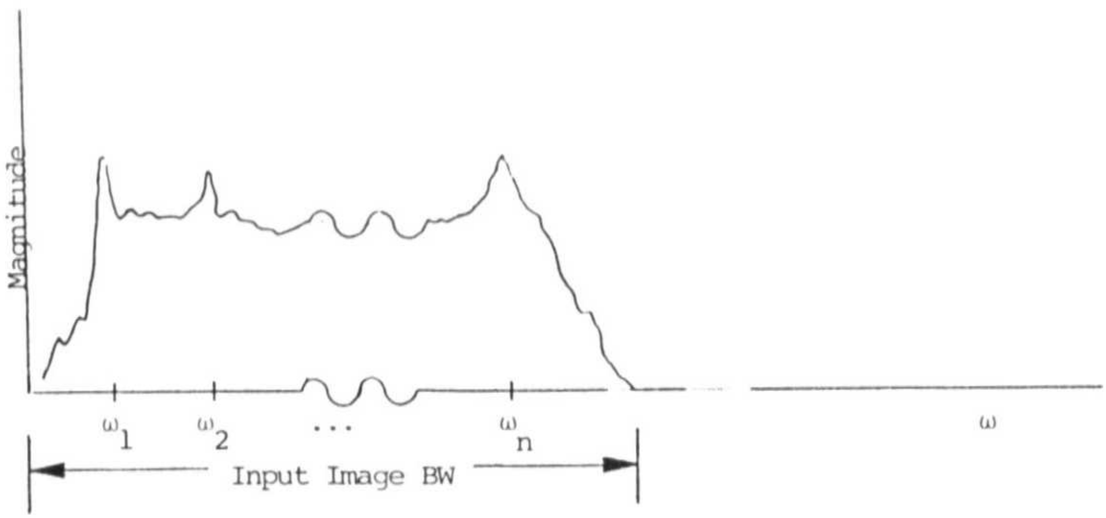
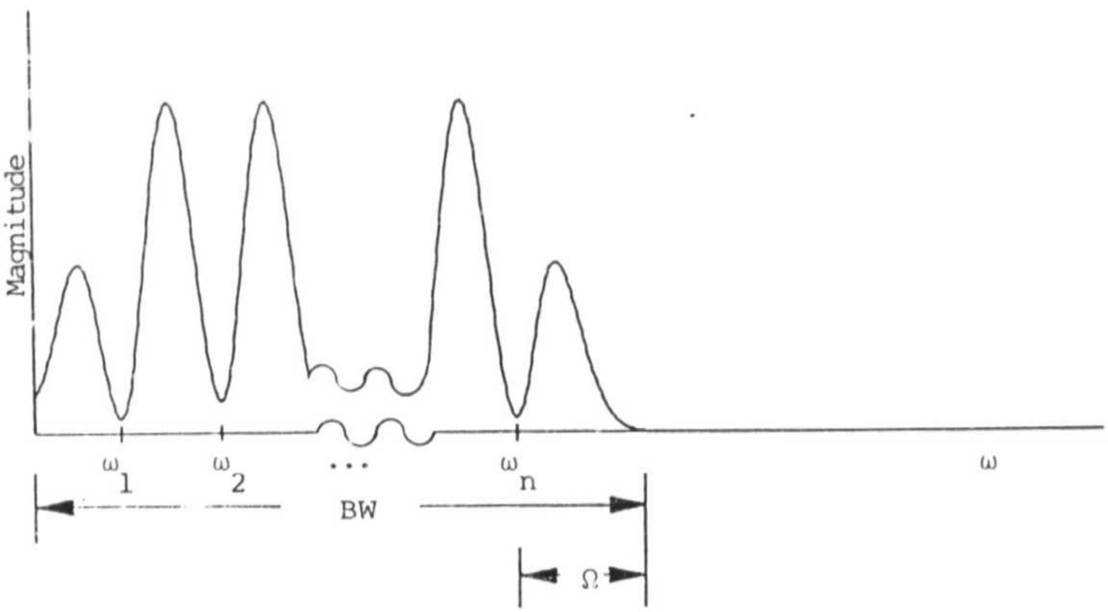


Figure 4 Magnitude of the response of the textural edge detection filter due to an input image with four ideal textures and three textural edges. The normalized spatial frequencies of the four textures are $.04\pi$, $.06\pi$, $.08\pi$, and $.1\pi$.



(a)



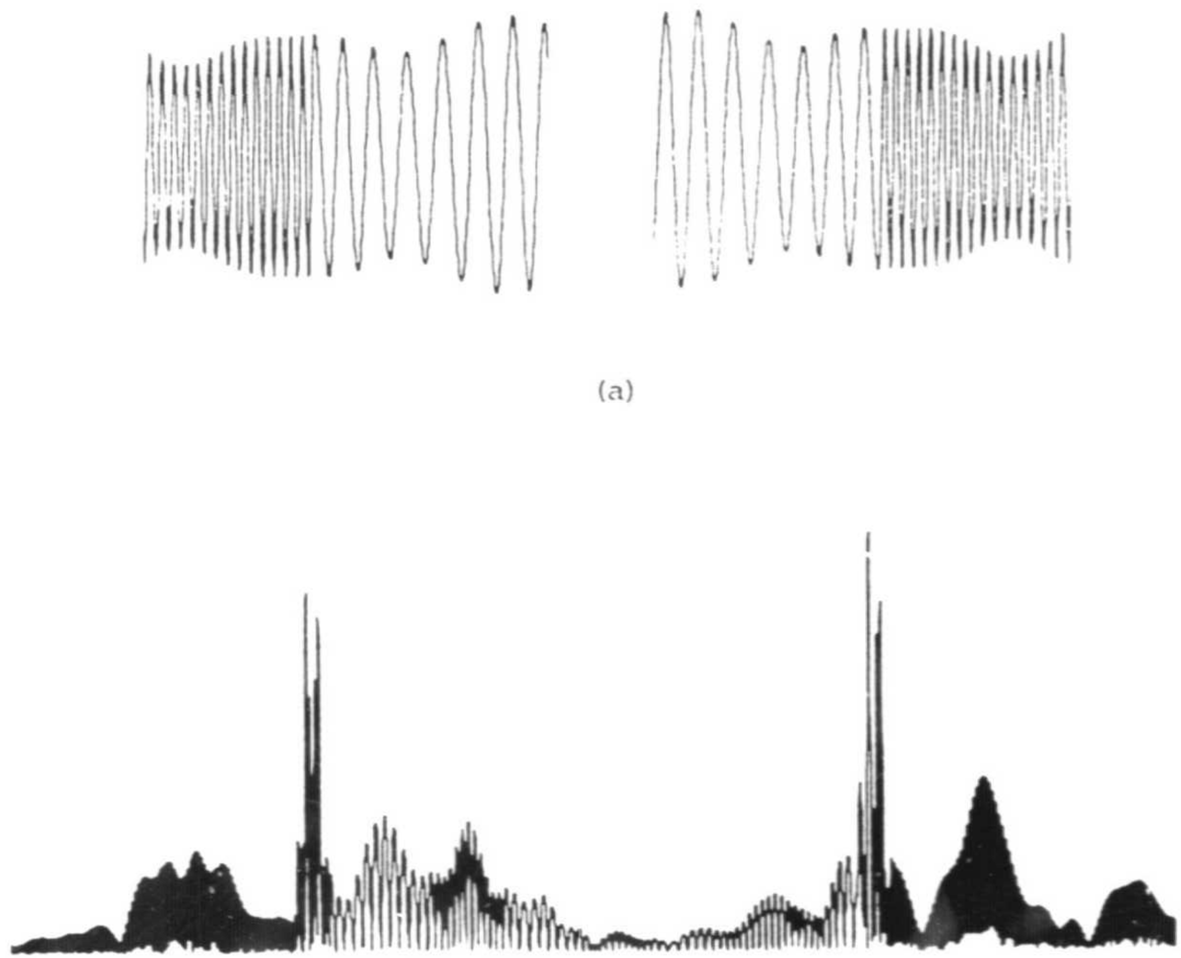
(b)

Figure 5 (a) Spectrum of an arbitrary input image.
(b) Spectrum of optimum textural edge detection filter with bandwidth shown in terms of ω_n and Ω.

$$BW = \omega_n + \delta \tag{11}$$

where ω_n represents the highest-frequency texture, and 2δ is the bandpass bandwidth of the filter centered on ω_n.

The most general case of the model used in this development is one in which each of the spatial frequencies representing the different textures in the image are allowed to rardomly deviate about some average frequency. This complication is introduced to allow for some of the irregularity of a real texture. A one-dimensional example in which both the amplitude and spatial frequency vary in proportion to independent random processes is shown in Figure 6. In this example, the average normalized spatial frequencies representing the two textures are $.04\pi$ and $.1\pi$ respectively. In terms of the general model presented in Section III, $\alpha(x)$ and $\beta(x)$ are independent Gaussian noise processes, with unit variance. The bandwidths of the amplitude noise and frequency noise processes are $.008\pi$ and $.006\pi$ respectively. Note that the filter adequately marks the two textural edges in the image, but also responds to regions within each texture where the spatial frequency changes. Decreasing the bandwidth of the noise modulating the frequency causes the spectral separation of the textures in the input image to increase. This results in improved performance of the filter at distinguishing textural edges from frequency deviations within a texture.



(a)

(b)

Figure 6 (a) Input image with both amplitude and frequency varying in proportion to a bandlimited Gaussian noise process (horizontal axis magnified two times around each textural edge).

(b) Magnitude of the optimum textural edge detector response due to (a).

V. CONCLUSION

A frequency domain textural edge detection filter has been developed which, for the given model and filter bandwidth, places a maximum amount of image energy within a specified resolution interval near the textural edge. The textural edge detector was derived by relating textural edge detection to tonal edge detection via complex lowpass equivalent representation. Hence, the optimum textural edge detector was found to be a sum of translated-in-frequency versions of the optimum tonal edge detector. This form allows the filter to be adapted to multitextural images. In addition, examples were presented which show the filter's insensitivity to tonal features in an image. The filter is adjustable; resolution can be traded for edge visibility in the case where the input image has been corrupted by noise.

The qualitative and complex nature of texture suggests that a totally general approach to modeling and classifying texture may never be found. It has been an objective in this investigation to develop a filter which optimizes a certain criteria relating to textural edge detection. But, as always, simplifications and assumptions were made indicating the need for further research. The model used in this development represented texture in terms of spatial frequency, and gray tone in terms of amplitude. One example of further research might be to base the development on a more complex model which incorporates a statistical description of texture. In addition, further work is needed in extension of the one-dimensional filter to two-dimensions.

This work has provided an approach to textural edge detection which can be implemented on digital hardware using the FFT. With the increased size and availability of digital computing facilities at a decreased cost, digital image processing methods will become more popular in the future.

REFERENCES

[1] K. S. Shanmugan, F. M. Dickey, J. A. Green, "An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-1, Jan. 1979.

[2] W. H. H. J. Lunscher, "The Asymptotic Optimal Frequency Domain Filter for Edge Detection," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-5, Nov. 1983.

[3] L. Kirvida, "Texture Measurements for the Automatic Classification of Imagery, IEEE Trans. Electromagnetic Compat., Vol. 18, pp. 38-42, Feb. 1976.

[4] L. Kirvida and G. Johnson, "Automatic Interpretation of ERTS Data for Forest Management," Symp. on Significant Results Obtained from the Earth Res. Technol. Satellite, NASA SP-327, March 1973.

[5] R. P. Kruger, W. B. Thompson, and A. F. Turner, "Computer Diagnosis of Pneumoconiosis," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-4, No. 1, pp. 40-49, January 1974.

[6] R. N. Sutton and E. L. Hall, "Texture Measures for Automatic Classification of Pulmonary Disease," IEEE Trans. Computers, Vol. C-21, pp. 667-676, July 1972.

[7] H. Maurer, "Texture Analysis with Fourier Series," Proc. Ninth Int. Symp. on Remote Sensing of Environment (Environmental Research Institute of Michigan, Ann Arbor, MI), pp. 1411-1420, April 1974.

[8] R. Bajcsy and L. Lieberman, "Computer Description of Real Outdoor Scenes," in Proc. Second Int. Joint Conf. on Pattern Recognition (Copenhagen, Denmark), pp. 174-179, August 1974.

[9] R. Bajcsy and L. Lieberman, "Texture Gradient as a Depth Cue," Comput. Graph. Image Processing, Vol. 5, No. 1, pp. 52-67, 1976.

[10] J. K. Townsend, "An Optimal Frequency Domain Textural Edge Detection Filter," Master's Thesis, University of Kansas, December 1984.

[11] J. G. Proakis, Digital Communications, McGraw-Hill, Inc., 1983.