THE DESIGN AND IMPLEMENTATION OF A HIGH PERFORMANCE VIDEO ENHANCEMENT INSTRUMENT

by

LAWRENCE BRUCE HEWITT
B.Sc., University of Calgary, 1982

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in the Department of
Electrical Engineering

ACCEPTED
ACULTY OF GRADUATE STUDIES
[redacted]
850816

We accept this thesis as conforming
to the required standard

[redacted]
Supervisor Dr. L.T. Bruton

[redacted]
Dr. J.C. Muzio

[redacted]
Dr. I. Barrodale

[redacted]
Dr. M. Shinbrot

[redacted]
Dr. A. Antoniou

[redacted]
Dr. G.C. Shoja

© LAWRENCE BRUCE HEWITT, 1985
UNIVERSITY OF VICTORIA

March 1985

All rights reserved. This thesis may not be reproduced in whole or in part, by mimeograph or other means, without the permission of the author.

Supervisor: Professor and Dean Leonard T. Bruton

ABSTRACT

A *high speed* instrument capable of digitizing, enhancing and displaying television images is presented. The instrument, called a Video Enhancement Instrument, uses two dimensional (2D) recursive Spatial Integrator filters for enhancing the images because of the reduced internal storage requirements and the reduced number of arithmetic operations when compared with other methods. The Instrument digitizes a (512 x 512) by 8 bit image in 1/30 second; enhances that image in approximately 4.0 seconds and instantaneously displays the enhanced image. The design technique can be extended, by parallelism, to further reduce the time taken to enhance an image.

In support of the design method, an experimental comparison is provided of the implementation of a 2D Spatial Integrator filter with the implementation of a corresponding 2D Direct Form filter.

Highly acceptable recursive 2D Spatial Integrator filters of order (3 x 3) may be implemented with single precision 16 bit integer multiplier coefficients. 2D Direct Form filters of order (3 x 3) require 20 bit multiplier coefficients.

It is shown by examination of row-recursive 2D filtering, that the $(n+1)$ th row of an image may be *processed concurrently* with the n th row, providing the recursion in the $(n+1)$ th row follows by one pixel, the recursion

in the n th row. Therefore, an algorithm which is suitable for *multi-processor* implementation is developed.

The Instrument consists of commercially available electronic circuitry for acquiring a single black and white image from a television camera and for displaying (512 x 512) by 8 bit images. The Instrument also contains a 2D Processor that has been specifically designed for 2D recursive filtering, a central controller to interface the user to the Instrument and to synchronize the 2D Processor to the Display Memory. The 2D Processor contains four high performance 16 bit TMS32010 microprocessors that are manufactured by Texas Instruments Inc. The Instrument provides twenty seven 2D filter transfer functions, a function for contrast stretching, and a function for histogram equalization.

Examiners: [redacted]

Dr. L.T. Bruton

[redacted]

Dr. J.C. Muzio

[redacted]

Dr. A. Barrodale

[redacted]

Dr. M. Shinbrot

[redacted]

Dr. A. Antoniou

[redacted]

Dr. G.C. Shoja

ACKNOWLEDGEMENTS

I would like to thank Dr. L.T. Bruton for his guidance, for recommending the project and for his helpful advice during the preparation of the manuscript. I would also like to thank Mr. N.R. Bartley for his assistance and suggestions.

Financial assistance received from the Natural Sciences and Engineering Research Council is gratefully acknowledged.

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

TABLE OF CONTENTS CONTINUED

TABLE OF CONTENTS CONTINUED

TABLE OF CONTENTS CONTINUED

TABLE OF CONTENTS CONTINUED

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES CONTINUED

LIST OF FIGURES CONTINUED

LIST OF FIGURES CONTINUED

1. A REVIEW OF LINEAR SHIFT INVARIANT 2D SIGNAL PROCESSING

1.1. INTRODUCTION

A signal is any medium which conveys information. Examples of signals include photographs, time varying voltages, and sequences of numbers. This thesis is primarily concerned with the enhancement of digitized TV images using two dimensional (2D) recursive filter algorithms. The concepts and filtering algorithms developed here can easily be extended to any 2D signal.

The objective of image enhancement is to make the image more informative and useful to the observer. Linear two dimensional (2D) filtering techniques are used to enhance images by amplifying or attenuating some of the spatial frequency components of the image with respect to others.

This chapter contains a brief introductory review of the concepts of linear shift invariant (LSI) 2D signal processing and the second chapter is concerned with the implementation of 2D filter algorithms. The Spatial Integrator method [5] is compared, in detail, with the widely used Direct Form method and shown to have important practical advantages in terms of the required arithmetical precision and the maximum processing speed. The comparison is pursued in chapter 3 for an illustrative and particularly useful

(3 x 3) order transfer function. Arithmetic, quantization effects and dynamic range are considered and detailed experimental simulations are reported. The results confirm the superiority of the Spatial Integrator technique. Previously, only sensitivity results were available for 2D Spatial Integrator methods [5].

A Video Enhancement Instrument is described in chapters 4 and 5. It is a user friendly instrument that employs the Spatial Integrator technique and multiple processors to enhance a (512 x 512) by 8-bit image.

1.2. TWO DIMENSIONAL SIGNAL PROCESSING

1.2.1. Two Dimensional Signals

A two dimensional (2D) discrete signal or 2D sequence is obtained from a 2D continuous signal $x_a(s,t)$ by sampling at a finite number of grid points. A discrete signal $x(m,n)$ sampled using a rectangular geometry is illustrated in Figure 1.1. The sample points or picture elements called *pixels* for $x(m,n)$ are obtained from

$$x(m,n) = x_a(s,t) \Big|_{\substack{s = mS \\ t = nT}} \tag{1.1}$$

where S, T are the sampling intervals along the spatial axes s, t respectively.

Practical 2D images such as a photograph or an X-ray image are spatially bounded; therefore $x(m,n)$ will only take on values for *finite* m and n. Outside this finite region the image is undefined.

The 2D unit impulse $\delta(m,n)$ is a special sequence defined as

$$\delta(m,n) = \begin{cases} 1 & m = n = 0 \\ 0 & otherwise \end{cases} \tag{1.2}$$

A second special sequence is the 2D unit step $u(m,n)$ given by

$$u(m,n) = \begin{cases} 1 & m \geq 0 \text{ and } n \geq 0 \\ 0 & otherwise \end{cases} \tag{1.3}$$

A third sequence of special interest is the complex exponential defined by

$$x(m,n) = exp(\, fm + gn \,) \tag{1.4}$$

where f and g are complex numbers. If f and g have unity magnitude this sequence reduces to the complex sinusoid

$$x(m,n) = exp(j\omega_1 m + j\omega_2 n) \tag{1.5}$$

where

$$\begin{aligned} f &= exp(j\omega_1) \\ g &= exp(j\omega_2) \end{aligned} \tag{1.6}$$

1.2.2. Fundamental Operations

Multi-dimensional signal processing is used for the extraction of useful information from a signal in the presence of extraneous information. This usually involves the use of an operator that maps an input signal $x(m,n)$ into an output signal $y(m,n)$ by some transformation $T[\,]$.

The transformation $T[\,]$ is *linear* if for arbitrary constants a and b and for

$$y_1(m,n) = T\left[x_1(m,n)\right];\ y_2(m,n) = T\left[x_2(m,n)\right] \tag{1.7}$$

the output can be given by

$$a\ y_1(m,n) + b\ y_2(m,n) = T\left[a\ x_1(m,n) + b\ x_2(m,n)\right] \tag{1.8}$$

A *shift invariant* system is one for which a shift in the input sequence implies a corresponding shift in the output sequence. A system will be shift invariant for a transformation $T[\,]$ if and only if

$$y(m-k,\ n-l) = T\left[x(m-k,\ n-l)\right] \tag{1.9}$$

for all integer values of k and l.

A system that is both linear and shift invariant is termed a *linear shift invariant* (LSI) system.

1.2.3. LSI Systems

LSI systems are an important class of discrete systems. This is because the behavior of these systems in many cases can be characterized without regard to a specific input. This makes LSI systems easy to analyze and relatively easy to design.

Consider a linear system with an input $x(m,n)$, which may be represented as a sum of weighted and shifted 2D impulses.

$$x(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k,l)\, \delta(m-k, n-l) \tag{1.10}$$

If this sequence $x(m,n)$ is applied to a linear system $L[\,]$ the output sequence will be given by

$$y(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k,l)\, L\left[\delta(m-k, n-l)\right] \tag{1.11}$$

because $x(k,l)$ is a constant for any (k,l). The value $L\left[\delta(m-k, n-l)\right]$ is the response of the linear system to a unit impulse located at (k,l). If the system is also shift invariant the response $L\left[\delta(m-k, n-l)\right]$ will be the same for any ordered pair (k,l). Therefore, the impulse response for the system is given by

$$h(m,n) = L\left[\delta(m,n)\right] \tag{1.12}$$

The impulse response $h(m,n)$ completely characterizes an LSI system.

The 2D output sequence $y(m,n)$ of Equation 1.11 can be written as the 2D convolution summation

$$y(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k,l)\, h(m-k,\, n-l) \tag{1.13}$$

1.2.4. 2D Frequency Response for LSI Systems

The complex sinusoidal output response $y(m,n)$ of an LSI system to a complex sinusoidal input $exp(j\omega_1 m + j\omega_2 n)$ is known as the system frequency response. The output response $y(m,n)$ is determined by convolving $exp(j\omega_1 m + j\omega_2 n)$ with the system impulse response $h(m,n)$.

$$y(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k,l)\, exp(j\omega_1(m-k) + j\omega_2(n-l)) \tag{1.14}$$

$$y(m,n) = exp(j\omega_1 m + j\omega_2 l)\, H(\omega_1,\omega_2) \tag{1.15}$$

where

$$H(\omega_1, \omega_2) = \left[\sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k,l)\, exp(-j\omega_1 k - j\omega_2 l) \right] \tag{1.16}$$

and $H(\omega_1, \omega_2)$ is known as the system frequency response. It is the 2D Fourier transform of the impulse response $h(k,l)$.

If the system magnitude frequency response $\left|H(\omega_1, \omega_2)\right|$ at a particular frequency ordered pair (ω_1,ω_2) is close to 1 then frequencies at (ω_1,ω_2) will be transmitted. A system for which $\left|H(\omega_1, \omega_2)\right|$ at (ω_1,ω_2) is close to zero will

reject these frequencies.

1.2.5. 2D Transfer Function

The 2D frequency response is the response of an LSI system to a complex sinusoidal input. If the input to an LSI system takes the form of a complex exponential

$$x(m,n) = z_1^m \; z_2^n \tag{1.17}$$

where z_1 and z_2 are complex numbers. The output $y(m,n)$ defined by the convolution summation will be

$$\begin{aligned} y(m,n) &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} z_1^{m-k} \; z_2^{n-l} \; h(k,l) \\ &= z_1^m \; z_2^n \; H(z_1, z_2) \end{aligned} \tag{1.18}$$

where

$$H(z_1, z_2) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k,l) \, z_1^{-k} \; z_2^{-l} \tag{1.19}$$

and is known as the transfer function. The transfer function of a 2D LSI system is the 2D Z transform of the impulse response.

1.2.6. 2D Difference Equations

A difference equation of a 2D LSI system defines a relationship between a 2D input signal $x(m,n)$ and a 2D output signal $y(m,n)$. A 2D LSI difference equation of finite order has the form

$$\sum_{k=-K_1}^{K_2} \sum_{l=-L_1}^{L_2} b_{kl}\, y(m-k, n-l) = \sum_{k=-K_3}^{K_4} \sum_{l=-L_3}^{L_4} a_{kl}\, x(m-k, n-l) \quad (1.20)$$

where K_1, K_2, K_3, K_4, L_1, L_2, L_3, L_4 are integers and b_{kl} and a_{kl} are constants.

An output $y(m,n)$ can be computed provided the input values $x(m-k, n-l)$ on the right hand side of Equation 1.20 are available and the other output values $y(m-k, n-l)$ have either been previously computed or have been specified as initial conditions.

A filter in which b_{kl} equals zero except for b_{00} which equals one in Equation 1.20 is termed a Finite Impulse Response (FIR) or nonrecursive filter. This is because a FIR filter has an impulse response with a finite number of samples. FIR filters are always stable.

An Infinite Impulse Response (IIR) or recursive filter is one in which the b_{kl} values of Equation 1.20 are not all equal to zero. Because the past output values $y(m-k, n-l)$ are used in the calculation of the current output $y(m,n)$, the impulse response of an IIR filter is spatially unbounded. 2D IIR filters may or may not be stable. All the 2D IIR filters used in this thesis were designed using the Ramamoorthy/Bruton method [3,4] which guarantees a stable 2D IIR filter.

The Transfer function $H(z_1, z_2)$ of a 2D LSI system can be computed by taking the 2D Z transform of the difference Equation (Equation 1.20) giv-

ing

$$H(z_1, z_2) \equiv \frac{Y(z_1, z_2)}{X(z_1, z_2)} = \frac{\sum_{k=-K_3}^{K_4} \sum_{l=-L_3}^{L_4} a_{kl}\ z_1^{-k}\ z_2^{-l}}{\sum_{k=-K_1}^{K_2} \sum_{l=-L_1}^{L_2} b_{kl}\ z_1^{-k}\ z_2^{-l}} \tag{1.21}$$

where

$$Y(z_1,z_2) \equiv Z\left[y(m,n)\right] \quad \text{and} \quad X(z_1,z_2) \equiv Z\left[x(m,n)\right]$$

1.2.7. 2D Discrete Fourier Transform

The 2D Discrete Fourier transform (DFT) is an exact Fourier transform for a spatially bounded 2D sequence and it is a Fourier series expansion for a 2D periodic sequence. This dual nature of the transform accounts for many of the properties it possesses.

A 2D periodic sequence $\tilde{x}(m,n)$ may be represented as a finite sum of harmonically related complex sinusoids

$$\tilde{x}(m,n) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \tilde{X}(k,l)\ exp\,(j\frac{2\pi}{M}mk\ +\ j\frac{2\pi}{N}nl) \tag{1.22}$$

where $\tilde{X}(k,l)$ are the Fourier series coefficients given by

$$\tilde{X}(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \tilde{x}(m,n)\ exp\,(-j\frac{2\pi}{M}mk\ -\ j\frac{2\pi}{N}nl) \tag{1.23}$$

These two relations (Equation 1.22 and Equation 1.23) define a transform from one periodic sequence to another periodic sequence.

A 2D finite duration sequence $x(m,n)$ can easily be expanded to a periodic sequence $\tilde{x}(m,n)$ by

$$\tilde{x}(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(m-kM, n-lN) \tag{1.24}$$

where M and N are the row and column size of the original sequence.

The DFT consists of samples of the Fourier transform. Therefore, in order for the complete recovery of the Fourier transform and the original sequence by the use of the inverse Fourier transform the input sequence $\tilde{x}(m,n)$ must be spatially limited. The 2D DFT $\tilde{X}(k,l)$ of the periodic extension $\tilde{x}(m,n)$ of $x(m,n)$ (Equation 1.24) computed from Equation 1.23 is the periodic extension of the sampled Fourier transform of the finite duration sequence $x(m,n)$.

The usefulness of the Fourier transform is that the Fourier transform of the convolution of two sequencies is the product of their Fourier transforms. A similar statement can be made about the DFT; the DFT of the *circular convolution* of two sequences is the product of their DFTs. It can be shown that if a 2D input sequence $x(m,n)$ has a spatially limited DFT of size $P \times Q$ and the impulse response of a 2D LSI system $h(m,n)$ has a spatially limited DFT of size $R \times S$, then the output sequence $y(m,n)$ can be computed using DFTs by

(1) Choosing the size of the DFT $M \times N$ such that

$$M \geq P + R \qquad (1.25)$$
$$N \geq Q + S$$

(2) The sequence $h(m,n)$ and $x(m,n)$ are augmented with zeros to fill in the undefined sample points.

(3) The $M \times N$ point DFTs of $h(m,n)$ and $x(m,n)$ are computed to form $H(k,l)$ and $X(k,l)$.

(4) The product $H(k,l)X(k,l)$ is computed.

(5) The $M \times N$ point inverse DFT of $H(k,l)X(k,l)$ is computed.

In simulating LSI systems, it is this result which is equivalent to computing the linear convolution that is desired.

1.2.8. Row - Column 2D FFT

A (M x N) point 2D DFT can be calculated by first computing the 1D DFT of each column of a 2D input $x(m,n)$ or the 2D impulse response $h(m,n)$, forming an intermediate array, followed by the 1D DFT of each row. If M and N are powers of 2 a 1D FFT can be used for each row or column. Thus, it can be shown that the total number of complex multiplications $T_{complex}$ using a Row - Column 2D FFT is

$$T_{complex} = MN \frac{\log_2(MN)}{2} \tag{1.26}$$

1.3. REALIZATION OF A 2D LSI SYSTEM

1.3.1. Introduction

A 2D LSI system enhances an input signal $X \equiv x(m,n)$ so that the desired output signal $Y \equiv y(m,n)$ is easier to observe. For example, a 2D LSI lowpass filter may be used to smooth a low illumination TV image because this type of degraded image contains high frequency spatial noise.

There is a lot of choice in the method used to implement a 2D LSI system. A designer may use a difference equation to directly implement a FIR or IIR filter, or the filter may be implemented using a 2D FFT. In this thesis, an Instrument was designed that implements IIR filters. The next section examines the different methods of implementation which formed the basis for choosing IIR filters.

1.3.2. FIR filters

One of the biggest advantages of a FIR filter is that an implementable 2D FIR filter can be designed to have a purely real frequency response. With a 2D LSI system, the input signal X is represented as a superposition of complex sinusoids. An LSI filter with a non-trivial frequency response will provide a complex amplification of some of the sinusoidal components of X

and a complex attenuation of others. The phase response of a filter will shift some frequency components of X with respect to the others. Thus for a filter with a non-linear phase response a line or an edge in a 2D image that is composed of aligned sinusoidal components will appear slightly blurred since some of the frequency components are shifted with respect to the other frequency components.

Another advantage of FIR filters is that they are always stable. An algorithm to design FIR filters is not constrained as to the values it may choose for the difference equation multiplier coefficients.

A disadvantage of FIR filters is that an acceptable frequency response may require a very high order filter, that is, there may be a large number of difference equation multiplications. In the 1D case, FIR filters of order 30 to 40 are not uncommon. In the 2D case a direct implementation of the difference equation of a (40 x 40) order filter would require 1681 multiplications for each output pixel. For a (512 x 512) pixel image the total number of real multiplications required for a (40 x 40) order Direct Form 2D FIR filter implementation is 419×10^6.

1.3.3. 2D FFT Implementation

A more efficient implementation of a high order FIR filter is to use a 2D FFT. The number of arithmetic operations required to compute an output image Y from an input image X using a 2D FFT is not as dependent on the

order of the filter as a spatial domain convolution summation. Assume that the DFT frequency response $H(k,l)$ has been computed and is stored in memory. To compute the desired output samples $y(m,n)$ two 2D FFTs must be performed, one forward and one inverse. If a Row - Column 2D FFT algorithm is used, the number of real multiplications required to compute the two 2D FFTs and to compute the intermediate array $H(k,l)\,X(k,l)$ is

$$2\,MN\,\log_2(MN) + 2\,MN \tag{1.27}$$

where M and N are powers of 2 and are the row and column sizes of the 2D FFT. For a (512 x 512) point 2D FFT implementation the total number of real multiplications will be 9.96×10^6. The size of the input image may have to be smaller than 512 pixels to prevent spatial aliasing from the circular convolution of the input image $X \equiv x(m,n)$ with the impulse response $h(m,n)$ of the filter.

Although a 2D FFT implementation of a filter is more efficient with respect to computation it requires sufficient storage to contain all N, M points of the input image X and all M, N points of the DFT frequency response of the filter $H(k,l)$. *If several types (highpass, lowpass) of filters with different cut off frequencies are required the amount of storage necessary becomes prohibitive.*

One of the inherent disadvantages of a 2D FFT implementation over a difference equation implementation is the spectral resolution of the 2D FFT. Consider the implementation of a lowpass filter with a very low cutoff frequency. In the 2D spatial domain, frequencies below the cutoff frequency will be sampled numerous times per cycle. To distinguish between frequencies slightly below the cutoff frequency and frequencies slightly above the cutoff frequency requires a highly selective filter. This implies the use of a high order difference equation.

In the 2D frequency domain, the spectral resolution that is the ω_1,ω_2 sampling frequency is fixed by the size of the 2D FFT. If the input image X is composed of low frequency components, the 2D FFT of this image will reflect these components by being composed of samples close to the origin in the ω_1,ω_2 frequency plane. Since the distance between adjacent frequency points (spectral resolution) is fixed by the size of the 2D FFT it would be impossible to design a highly selective frequency domain lowpass filter in this frequency range. To increase the spectral resolution a larger size 2D FFT could be used but this would also increase the number of computations.

1.3.4. 2D IIR Filters

The implementations in the spatial domain discussed above have been for FIR filters. 2D FIR filters are a subset of the broader class of IIR filters. The difference equation of a general finite order IIR filter is given in

Equation 1.20. The general transfer function is given in Equation 1.21. The design of FIR filter frequency responses is limited to the manipulation of the a_{kl} coefficients of Equation 1.21 since the denominator of a FIR filter frequency response is one. With IIR filters both the a_{kl} and b_{kl} coefficients may be optimized. As expected, the degree of freedom in being able to manipulate both the denominator and numerator of Equation 1.21 allows a *similar magnitude frequency response to be met with a much lower order filter than with a FIR filter.*

There are some unique problems in the implementation of IIR filters. The implementation of IIR filters is discussed in length in chapter 2.

Highly selective (3 x 3) order stable IIR filter have been designed using the Bruton/Ramamoorthy method. A direct implementation of these (3 x 3) order filters using the difference equation (Equation 1.20) requires 31 multiplications per pixel. For an image size of (512 x 512) pixels the required number of multiplications is 8.12646×10^6.

IIR filters do not require as much memory as either a FIR filter or a 2D FFT implementation. The required information for the difference equation (Equation 1.20) to compute a pixel $y(m,n)$ is the input and output pixels in the neighborhood of $y(m,n)$ and $x(m,n)$. Thus to compute an output pixel $y(m,n)$ only a small window of the entire image is required. It is shown in chapter 2 that a (M x N) input image X may be transformed into an output

image Y using a (3 x 3) order IIR filter with an internal storage requirement of (3 x M) words.

When using a 2D FFT implementation, the entire image X is usually resident in memory before the FFT transformations. A 2D *LSI system that implements IIR filters may start processing the input X while it is still being sampled. This presents a significant savings in time and memory over a 2D FFT implementation.*

The advantages of using 2D IIR filters to implement an LSI system are

(1) a reduced number of arithmetic operations when compared to FIR and 2D FFT implementations.

(2) a significant savings in the storage requirement for intermediate calculations

(3) a spatial domain implementation that does not have the spectral resolution problems inherent with FFT implementations.

A disadvantage of a IIR filter is that it is much more difficult to design. The manipulation of the b_{kl} coefficients in Equation 1.20 must be constrained to achieve stability. If for any frequency ordered pair z_1,z_2 where $\left|z_1,z_2\right| \geq 1$ the denominator of Equation 1.21 becomes zero the filter *may* be unstable.

Another disadvantage of IIR filters is that in general the phase response of a IIR filter will be non-linear. If the distortion in the output image Y due to the phase becomes too severe the image may need to be filtered a second time in the opposite directions. The resultant zero-phase transfer function $G(\omega_1,\omega_2)$ can be expressed in the frequency domain as [2]

$$G(\omega_1,\omega_2) = H(\omega_1,\omega_2)\, H^*(\omega_1,\omega_2) \tag{1.28}$$

The inverse DFT of $H^*(\omega_1,\omega_2)$ is given by

$$F^{-1}\left[H^*(\omega_1,\omega_2)\right] = h^*(-m\,,-n\,) \tag{1.29}$$

from which it is clear that the directions of recursion are reversed.

Despite the disadvantages of IIR filters, they provide an excellent method of implementing LSI systems. The computational, storage and data manipulation savings of a 2D IIR filter implementation over 2D FFT and 2D FIR filter implementations make possible the design of the 2D Video Enhancement Instrument described in chapter 4.

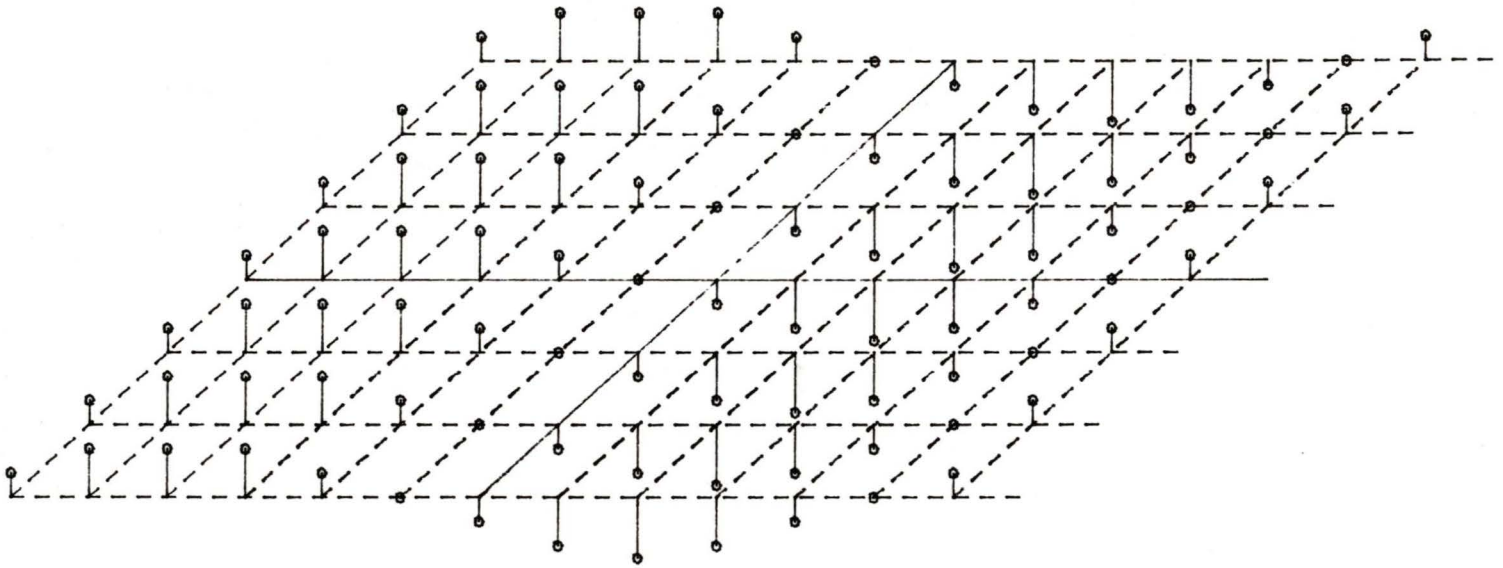


Figure 1.1 Graphical Representation of 2D Pixels

2. THE REALIZATION OF TWO DIMENSIONAL RECURSIVE FILTERS

2.1. INTRODUCTION

A digital filter can be implemented in two ways; the filter can be realized on a digital computer or it can be realized with dedicated hardware. Two dimensional (2D) filters implemented in software perform too slowly for many applications.

A hardware system can be designed to be portable, fast, and a fraction of the cost of even a minicomputer system. Hardware has the disadvantage that the filter structure, that is the algorithm, cannot be changed as easily as software. With both methods the user must design and incorporate acquisition and display interfaces. The trade-off between the two methods is the flexibility of software versus the computational speed of dedicated hardware.

The realization of 2D filters is described in this chapter. This description takes the form of a signal flow graph (SFG). A Direct Form 2D SFG is developed for row-recursive or column-recursive image processing. This SFG structure is extended to include Direct Form and Spatial Integrator realizations.

A brief introduction to a system for high-speed image processing is presented in section 2.6. High-speed is attained by using an algorithm that splits the 2D processing into a number of 1D processing tasks. Each of the 1D processing tasks consists of calculating the output pixels $y(m,n)$ of a single row or column of the 2D input image $X \equiv x(m,n)$. The 1D processing tasks can be performed in parallel to achieve a high-speed image processing system.

2.2. 2D RECURSIVE FILTERS

Quarter plane (QP) filters are a class of two dimensional infinite impulse response (IIR) filters that can be realized recursively. The difference equation for a 2D LSI QP recursive filter is

$$y(m,n) = \sum_{k=0}^{K} \sum_{l=0}^{L} a_{kl}\, x(m-k,n-l) - \sum_{k=0}^{K} \sum_{\substack{l=0 \\ (k,l)\neq(0,0)}}^{L} b_{kl}\, y(m-k,n-l) \quad (2.1)$$

where $y(m,n)$ is the current output pixel, $x(m,n)$ is the corresponding input pixel, and the pixels $x(m-k, n-l)$ are contained in a rectangular mask of input pixels of dimensions $K+1$, $L+1$ shown in Figure 2.1b. Similarly the pixels $y(m-k, n-l)$ are contained in a rectangular mask of output pixels of dimensions $K+1$, $L+1$ shown in Figure 2.1a. Equation 2.1 describes a filter that is causal in the positive m and positive n directions. This means that the current output value $y(m,n)$ is determined from the current input value $x(m,n)$ and mask values which lie below and to the left

of the current output $y(m,n)$ in Figure 2.1a and Figure 2.1b. Because the current output $y(m,n)$ is calculated from its mask values the direction of recursion, or the direction in which the filter output $y(m,n)$ is calculated, is fixed in one of the two directions shown in Figure 2.2. In the case of a row-recursive QP filter $y(m+1,n)$ will be the next pixel to be calculated, for a column-recursive QP filter $y(m,n+1)$ will be the next pixel to be calculated (Figure 2.2).

Other types of QP filters in which the direction of recursion is in the negative m or n direction are also realizable and shown in Figure 2.3. There are four types of QP recursive filters depending upon their mask orientations. These filters correspond to the four quadrants of the x-y plane.

2.3. A SIGNAL FLOW GRAPH FOR A DIRECT FORM 2D FILTER

Consider the filtering of an image $X \equiv x(m,n)$ having $M \times N$ pixels where

m is the column index; $m = 0, 1, 2, 3 \ldots M-1$

n is the row index; $n = 0, 1, 2, 3 \ldots N-1$

using a QP filter to produce an output image Y. For the 2D difference equation (Equation 2.1) a spatially bounded image may be processed row-by-row (incrementing the m index first) or column-by-column (incrementing the n

index first). In either case the 2D input image is presented to the filtering algorithm as a stream of vectors, where each vector is a row or column of the input image X as shown in Figure 2.4. For a row-recursive QP filtering algorithm there will be N row vectors each of M pixels in length. When processing an input vector U_n , information is required from the previous input image row vectors U_{n-1}, U_{n-2}, $\cdots$, U_{n-K} and output image row vectors V_{n-1}, V_{n-2}, $\cdots$, V_{n-K} to determine the pixel elements of output vector V_n . For example, consider a QP filter of order (2 x 2). The difference equation for this filter is given by

$$y(m,n) = \sum_{k=0}^{2}\sum_{l=0}^{2} a_{kl}\, x(m-k,n-l) - \sum_{k=0}^{2}\sum_{\substack{l=0 \\ (k,l)\neq(0,0)}}^{2} b_{kl}\, y(m-k,n-l) \quad (2.2)$$

The masks required to calculate the current output pixel $y(m,n)$ are shown in Figure 2.5.

When the current output $y(m,n)$ has been calculated using the difference equation (Equation 2.2), the next output pixel to be calculated in a row-recursive QP filtering algorithm will be $y(m+1,n)$. The new masks required to determine $y(m+1,n)$ are shown in Figure 2.6 in the solid frame.

A comparison of the masks for the previous pixel $y(m,n)$ and the masks for the pixel $y(m+1,n)$ is shown in Figure 2.6. The information required for the masks for calculating $y(m+1,n)$ can be obtained from the $y(m,n)$ masks with the exception of the following pixels; $x(m+1,n)$,

$x(m+1,n-1)$, $x(m+1,n-2)$, $y(m,n)$, $y(m+1,n-1)$, $y(m+1,n-2)$. Of course, $y(m,n)$ is the output from the previously calculated difference equation so it is readily available and $x(m+1,n)$ is available from the input image.

Placing this into vector notation, the required vector elements for the calculation of output vector element $V_n(m+1)$ are

Input vector elements

$$\begin{array}{lll} U_n(m+1) & U_n(m) & U_n(m-1) \\ U_{n-1}(m+1) & U_{n-1}(m) & U_{n-1}(m-1) \\ U_{n-2}(m+1) & U_{n-2}(m) & U_{n-2}(m-1) \end{array}$$

Output vector elements

$$\begin{array}{lll} & V_n(m) & V_n(m-1) \\ V_{n-1}(m+1) & V_{n-1}(m) & V_{n-1}(m-1) \\ V_{n-2}(m+1) & V_{n-2}(m) & V_{n-2}(m-1) \end{array}$$

Consider a 2D SFG implementation of the above algorithm which processes row vectors U_n as inputs and row vectors V_n as outputs. The special diagrammatic notation for such a 2D SFG is summarized in Figure 2.7 where it will be observed that the operator T_1 implies a unit row shift between two nodes of the graph and similarly T_2 implies a unit column shift. The 2D SFG in Figure 2.8 is proposed for the implementation of the (2 x 2) order 2D difference equation (Equation 2.2). The suggested algorithm requires three input row vectors U_n, U_{n-1}, U_{n-2} and two previously

calculated output row vectors V_{n-1}, V_{n-2} to calculate one output row vector V_n.

For a spatially bounded image that is processed using a row-recursive QP filtering algorithm, pixel $x(m,n+1)$ will be processed M iterations after pixel $x(m,n)$ as shown in Figure 2.9. This suggests that each element of input vector U_{n-1} in Figure 2.8 be obtained by shifting each element of input vector U_n , M pixels. Similarly, vectors U_{n-2}, V_{n-1}, V_{n-2} can be derived from vectors U_{n-1}, V_n, and V_{n-1} respectively by shifting each of the elements of these vectors, M pixels.

The 2D structure in Figure 2.8 processes a spatially bounded image $X \equiv \left\{ U_0\ U_1\ U_3\ \cdots\ U_{N-1} \right\}$ that is the dimensions of the image M and N are finite; therefore the mask elements at the start of each row vector U_n must be initialized. For example, when calculating the output vector element $V_n(0)$ all the mask elements except $U_{n-1}(0)$, $U_{n-2}(0)$, $V_{n-1(0)}$, $V_{n-2}(0)$ are off the edge of the image as shown in Figure 2.10 and should be set to an initial value since they would otherwise contain values derived from the calculation of $V_{n-1}(M-1)$. An initial value of zero is chosen.

Taking the above considerations into account a 2D spatially bounded SFG is proposed and is shown in Figure 2.11. The SFG is known as a *Direct Form* structure because there is a one to one correspondence between

the nodes of the SFG and the 2D difference equation (Equation 2.2).

The advantages of the proposed SFG structure are:

(1) The task of processing a 2D IIR filter can be considered as a series of 1D processing tasks since the input image X and output image Y are considered as a stream of 1D row vectors U_n and V_n. It is then easier to implement the 1D processing tasks in dedicated hardware or with computer software than it is to implement a 2D processing task.

(2) After the input image X has been translated into the input vector stream U_n, the same memory used for the input image can be used to store the output vector stream V_n. For a (512 x 512) by 8-bit image this is a saving of 256 Kbytes.

2.4. A REDUCED COLUMN SHIFT 2D DIRECT FORM STRUCTURE

A 1D canonic recursive filter is one in which the number of unit delays is equal to the order of the filter [7]. This section develops a similar reduced unit delay structure for 2D recursive filters. The proposed reduced unit delay 2D QP filter of order (K x L) (row order by column order) when processed

using a row-recursive filtering algorithm will have K(L+1) row delay elements and L column delay elements. The advantage of this structure is that a reduction in the number of shift operations means a reduction in the amount of internal storage.

Consider the Z-transform of the (2 x 2) order filter difference equation (Equation 2.2)

$$H_2(z_1,z_2) \equiv \frac{Z(y(m,n))}{Z(x(m,n))} = \frac{N(z_1,z_2)}{D(z_1,z_2)} \tag{2.3}$$

$$= \frac{\sum_{k=0}^{2} \sum_{l=0}^{2} a_{kl} z_1^{-k} z_2^{-l}}{1 + \sum_{k=0}^{2} \sum_{\substack{l=0 \\ (k,l) \neq (0,0)}}^{2} b_{kl} z_1^{-k} z_2^{-l}}$$

In Figure 2.12a the transfer function, $H_2(z_1,z_2)$ is broken down into two simpler cascaded transfer functions. In the reduced column delay structure the order in which the two simple transfer functions are realized is reversed as shown in Figure 2.12b. A realization of this structure is shown in Figure 2.13 using a signal flow graph representation. The T^1 operation of Figure 2.11 can be replaced by z_1^{-1} and the T^M operation by z_2^{-1} since these are the z-domain characteristics of these elements.

Examining nodes A and A' of Figure 2.13 one finds that both these nodes contain the same value. Similarly, the points B and B' are the same value since there is only a unit column delay between A-B and A'-B'.

Therefore, the unit column delays in path $A'-B'$, $B'-C'$ ... can be eliminated. This yields the 2D reduced column delay Direct Form structure of Figure 2.14.

2.5. SPATIAL INTEGRATOR STRUCTURE

The signal flow graph of Figure 2.13 leads to a practical method of implementing a Direct Form 2D filter in hardware or software. However, the Direct Form structure is highly sensitive to quantization errors of the multiplier coefficients [6]. Two dimensional discrete filters using Spatial Integrators have been investigated by Bruton and Strecker [5]. It is expected that these structures would allow for much shorter multiplier coefficient word lengths. In 1D filters a structure that has a reduction in multiplier coefficient sensitivity generally has an increase in dynamic range [6]. This is expected to apply to 2D filters as well. In this section, a SFG for the 2D Spatial Integrator structure is proposed. The method involves replacing the shift operators z_1^{-1} and z_2^{-1} by Spatial Integrators so that $\hat{z}_1^{-1} \equiv z_1^{-1}/(1-z_1^{-1})$ and $\hat{z}_2^{-1} \equiv z_2^{-1}/(1-z_2^{-1})$ [5]. From a given transfer function $H(z_1,z_2)$ the corresponding Spatial Integrator transfer function $\hat{H}(\hat{z}_1,\hat{z}_2)$ is easily determined by replacing $\hat{z}_i$ $(i=1,2)$ with z_i -1 and equating coefficients [5]. A (2 x 2) order Spatial Integrator transfer function $\hat{H}_2(\hat{z}_1,\hat{z}_2)$ is given by

$$\hat{H}_2(z_1,\hat{z}_2) = \frac{\sum_{k=0}^{2}\sum_{l=0}^{2} c_{kl}\,\frac{z_1^{-k}}{(1-z_1^{-1})^k}\,\frac{z_2^{-l}}{(1-z_2^{-1})^l}}{1+\sum_{k=0}^{2}\sum_{\substack{l=0\\(k,l)\neq(0,0)}}^{2} d_{kl}\,\frac{z_1^{-k}}{(1-z_1^{-1})^k}\,\frac{z_2^{-l}}{(1-z_2^{-1})^l}} \tag{2.4}$$

where c_{kl} and d_{kl} are the Spatial Integrator multiplier coefficients.

For a bounded image the output $r(m,n)$ of a row Spatial Integrator $z_1^{-1}/(1-z_1^{-1})$ is simply the sum of all the input quantities $u(i,n)$ of the present row and is given by

$$r(m,n) = \sum_{i=0}^{m-1} u(i,n) \tag{2.5}$$

Similarly, the output $s(m,n)$ of a column Spatial Integrator $z_2^{-1}/(1-z_2^{-1})$ is given by

$$s(m,n) = \sum_{j=0}^{n-1} u(m,j) \tag{2.6}$$

Since row summation and column summation are independent operations, the row and column spatial integrations may be performed in any order.

In the Z domain the term

$$\hat{z}_1^{-2},\hat{z}_2^{-2} \equiv \frac{z_1^{-2}}{(1-z_1^{-1})^{-2}}\,\frac{z_2^{-2}}{(1-z_2^{-1})^{-2}} \tag{2.7}$$

from Equation 2.4 may be calculated in one of the six ways shown in Figure

2.15. With this in mind the 2D Spatial Integrator structure shown in Figure 2.16 is proposed by replacing the shift operations and multiplier coefficients of Figure 2.14 by their corresponding Spatial Integrator operations. The mask element values P,Q,S,T,V and W of Figure 2.16 are initialized to zero at the start of each row in a similar fashion to the Direct Form structure.

The advantages of the Spatial Integrator structure are:

(1) Redundant column shifts are eliminated by reversing the order in which the numerator and denominator of the transfer function are realized which decreases the amount of necessary internal storage.

(2) The use of Spatial Integrators reduces the sensitivity of the structure to multiplier coefficient quantization The Spatial Integrator structure is chosen for implementation in the 2D Video Enhancement Instrument (chapter 4) because of these advantages.

2.6. INTRODUCTION TO A MULTIPLE PROCESSOR IMPLEMENTATION

Two dimensional filtering requires an incredible amount of mathematical processing. For instance, for the 2 by 2 order filter shown in Figure 2.13 there are 17 multiplications and 16 additions per pixel. If one is processing

an image 512 pixels by 512 pixels, the required number of mathematical operations is 8,650,752. Using a processor that can perform a mathematical operation every other instruction cycle (50 % overhead), an instruction rate of over 17.3 million instructions per second is necessary to process the image in less than one second.

A simple method of increasing the 2D processing throughput rate by dividing the algorithm into separate vectors which may be calculated concurrently is investigated in this section.

Consider the processing of a $M \times N$ image X into Y using the SFG in Figure 2.14 and starting at the lower left-hand corner as shown in Figure 2.17a. The SFG algorithm in Figure 2.14 computes the output pixel $y(0,0)$ on its first iteration, and the z_2^{-1} mask values A, D in Figure 2.14 are placed in a long shift vector of length M. For a row-recursive algorithm the processor then computes output pixel $y(1,0)$. Output pixel $y(0,1)$ is computed M iterations after output pixel $y(0,0)$. *However, output pixel* $y(0,1)$ *can be calculated concurrently with output pixel* $y(1,0)$. The information required to compute output pixel $y(0,1)$ is the following

a) Input pixel $x(0,1)$

b) Output of z_2^{-1} delay elements from the calculation of output pixel $y(0,0)$. (States D and G of Figure 2.14)

All the other states are initialized to zero because it is the start of a new row.

Thus in the second iteration of the algorithm of Figure 2.14 two output pixels can be calculated, namely $y(0,1)$ and $y(1,0)$ *using two hardware processors.* In the third iteration of the algorithm three processors may be used in parallel to *simultaneously* calculate the output for pixels $y(2,0)$, $y(1,1)$, $y(0,2)$. The image is effectively processed in a diagonal manner from the starting lower left-hand corner to the opposite corner (Figure 2.17c).

For a square image $M \times M$, if the total number of processors equals the row size M, with a processor associated with each row, by the time the first processor reaches the end of the first ($m = M-1$, $n = 0$) row, the last processor will be on the first pixel of the last ($m = 0$, $n = M-1$) row. If one iteration of the filtering algorithm calculates an output pixel in T seconds, the time it takes to process the whole image is equal to the time it takes a single processor to complete $(2M-1)$ pixels, so that the total time for computing the complete image T_{total} is given by

$$T_{total} = (2M-1) \times T \ seconds \tag{2.8}$$

This may be compared with implementing the same SFG with one processor in which case the total execution time T_{one} is given by

$$T_{one} = M^2 \ T \ seconds \tag{2.9}$$

If several images are resident in memory, the first processor could start on the next image while the other $M-1$ processors are working on the last image. This would give an overall pipelined throughput of $1/(MT)$ images per second.

For example, for a (512 x 512) image using a (2 x 2) order Direct Form structure (Figure 2.14), 512 processors which can perform an add or multiply in 400 nano-seconds (2.5 MHz instruction rate), could process the whole image in 27.0 milli-seconds (video rate is 33.33 milli-seconds). For comparison, one processor would take 6.92 seconds to process the complete image. These calculations assumes that only 1/2 of the machine instructions are mathematical (50 % overhead).

2.7. SUMMARY

Quarter plane infinite impulse response filters are of interest since they can be easily realized in hardware or software. A 2D signal flow graph for efficient software or dedicated hardware filter designs is presented in Figure 2.13.

By reversing the order in which the numerator and denominator of the filter transfer function are realized, redundant shifts are eliminated. This structure is given in Figure 2.14.

The principle behind the algorithm developed for Figure 2.14 is extended to include the Spatial Integrator filter structure.

A method of increasing the throughput of a 2D filtering system through parallel processing is presented. Examination of row-recursive QP filtering shows that the $(n+1)$ th row of the image may be processed concurrently with the n th row, providing the $(n+1)$ th row lags the n th row by one pixel. For a highly parallel processing system square images may be processed at

$$T \times (2M-1)$$

seconds per image where M is the number of pixels per row or column T is the time it takes to process 1 pixel.

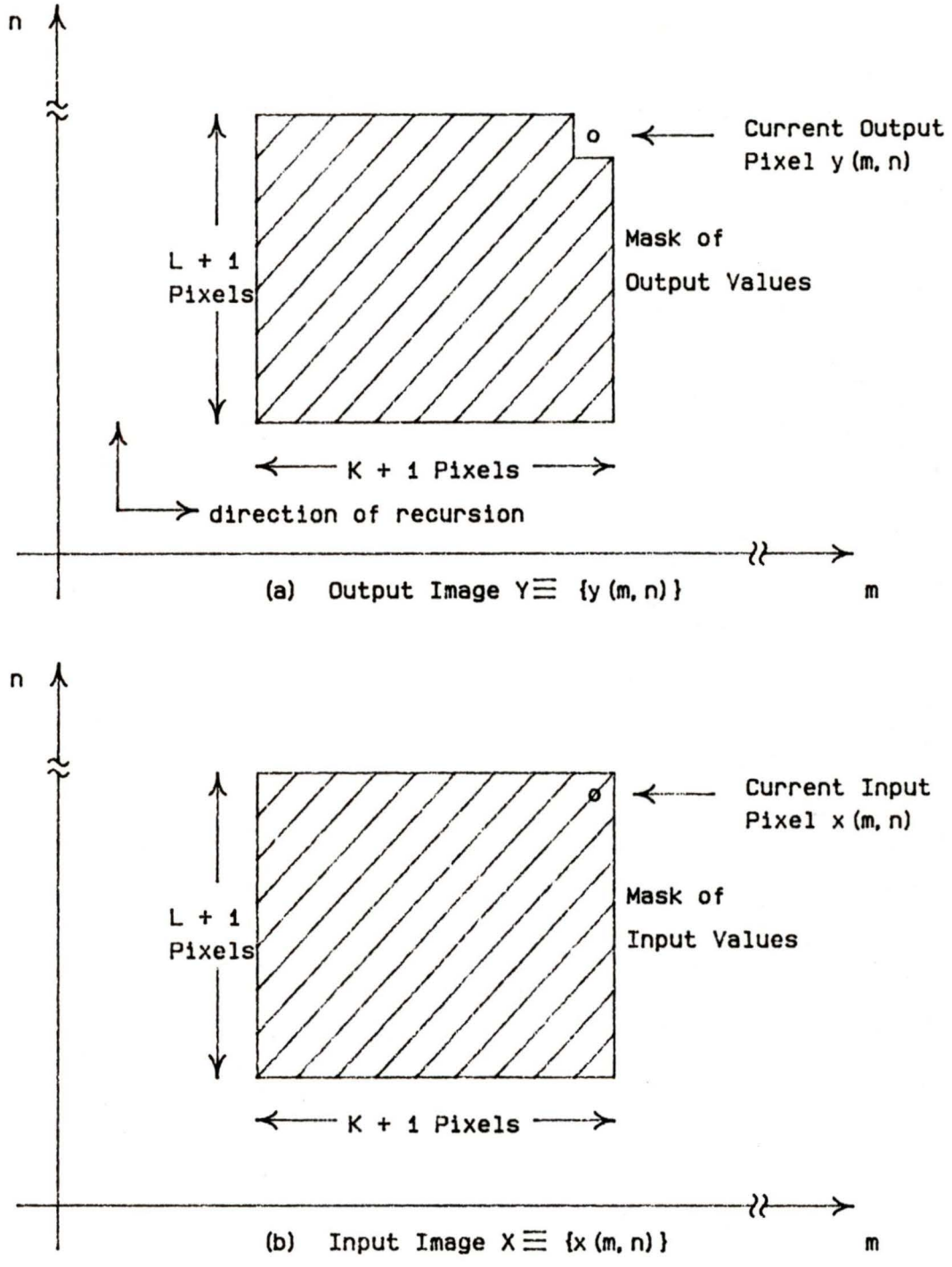
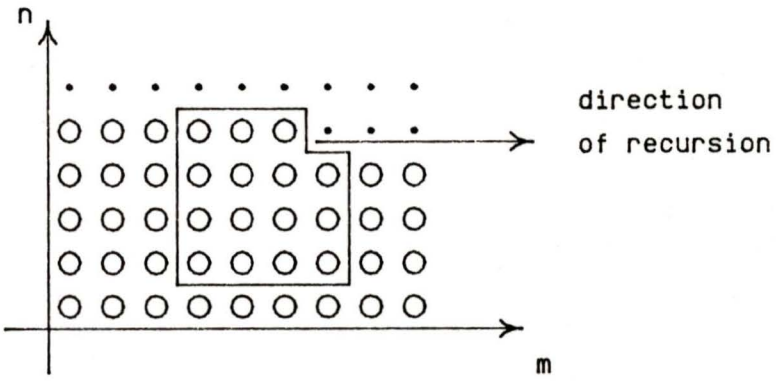
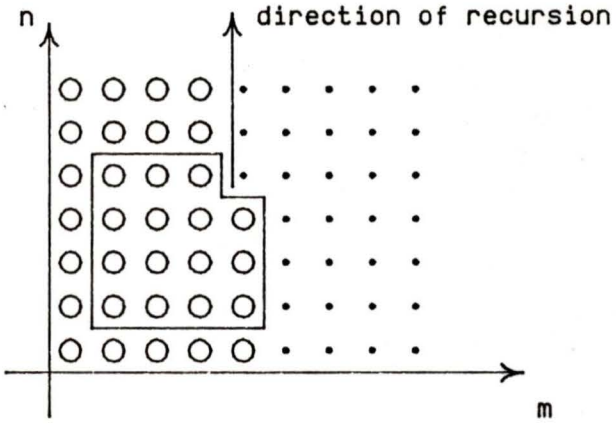


Figure 2.1 Quarter Plane Filter Masks



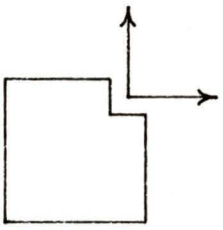
a) Direction of Recursion for Row-Recursive Processing of Quarter Plane Filters



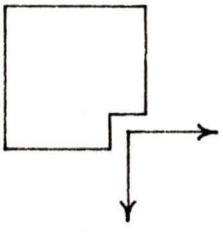
b) Direction of Recursion for Column-Recursive Processing of Quarter Plane Filters

• Represent output values to be calculated
O Represent computed output values

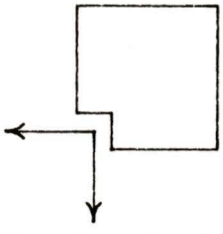
Figure 2.2 Direction of Recursion for Quarter Plane Filters



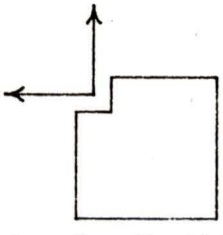
a) Positive Row Positive Column Mask



b) Positive Row Negative Column Mask



c) Negative Row Negative Column Mask



d) Negative Row Positive Column Mask

Figure 2.3 Output Filter Masks and the Resulting Directions of Recursion

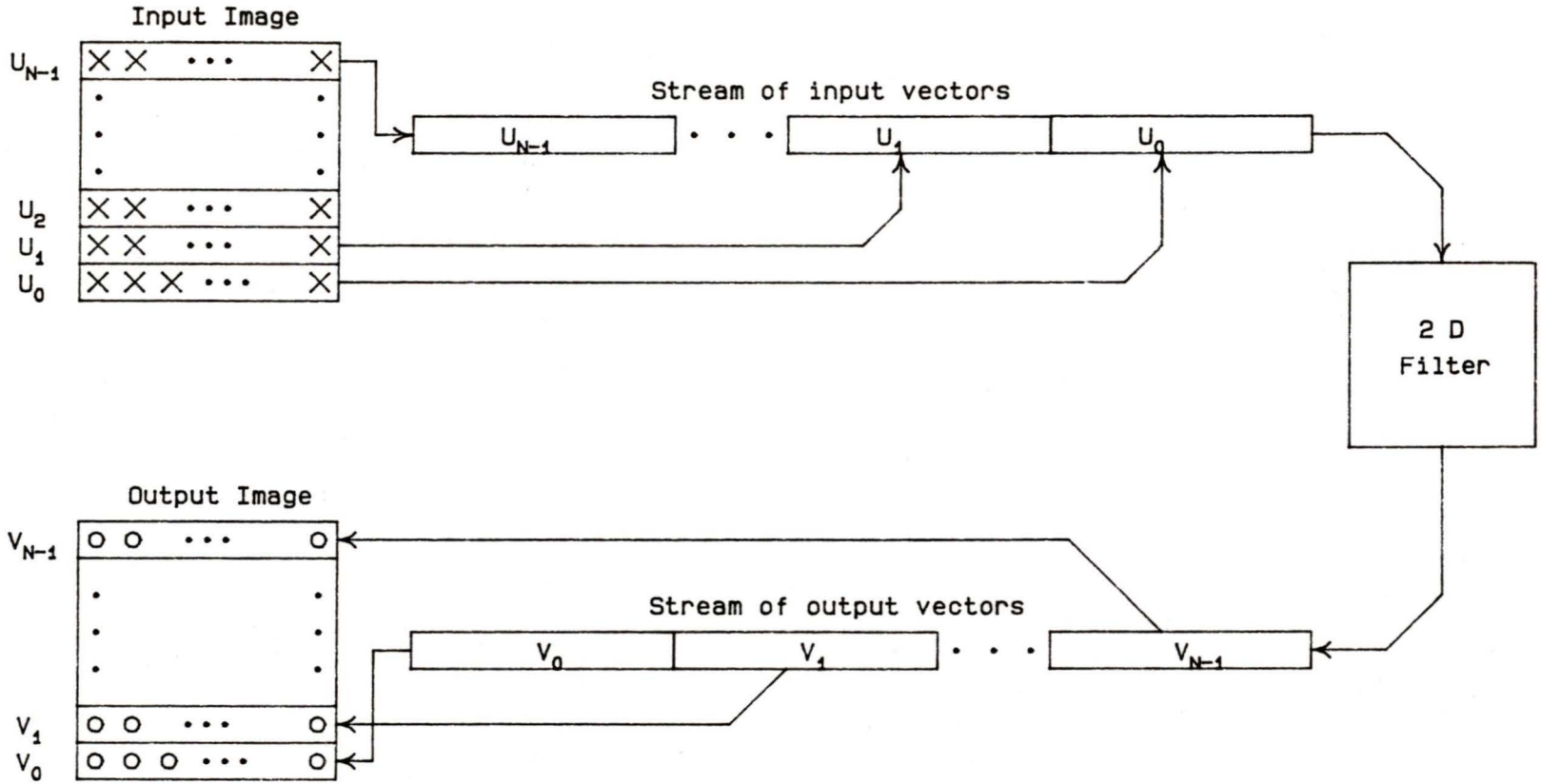


Figure 2.4 Row-Recursive Image Processing

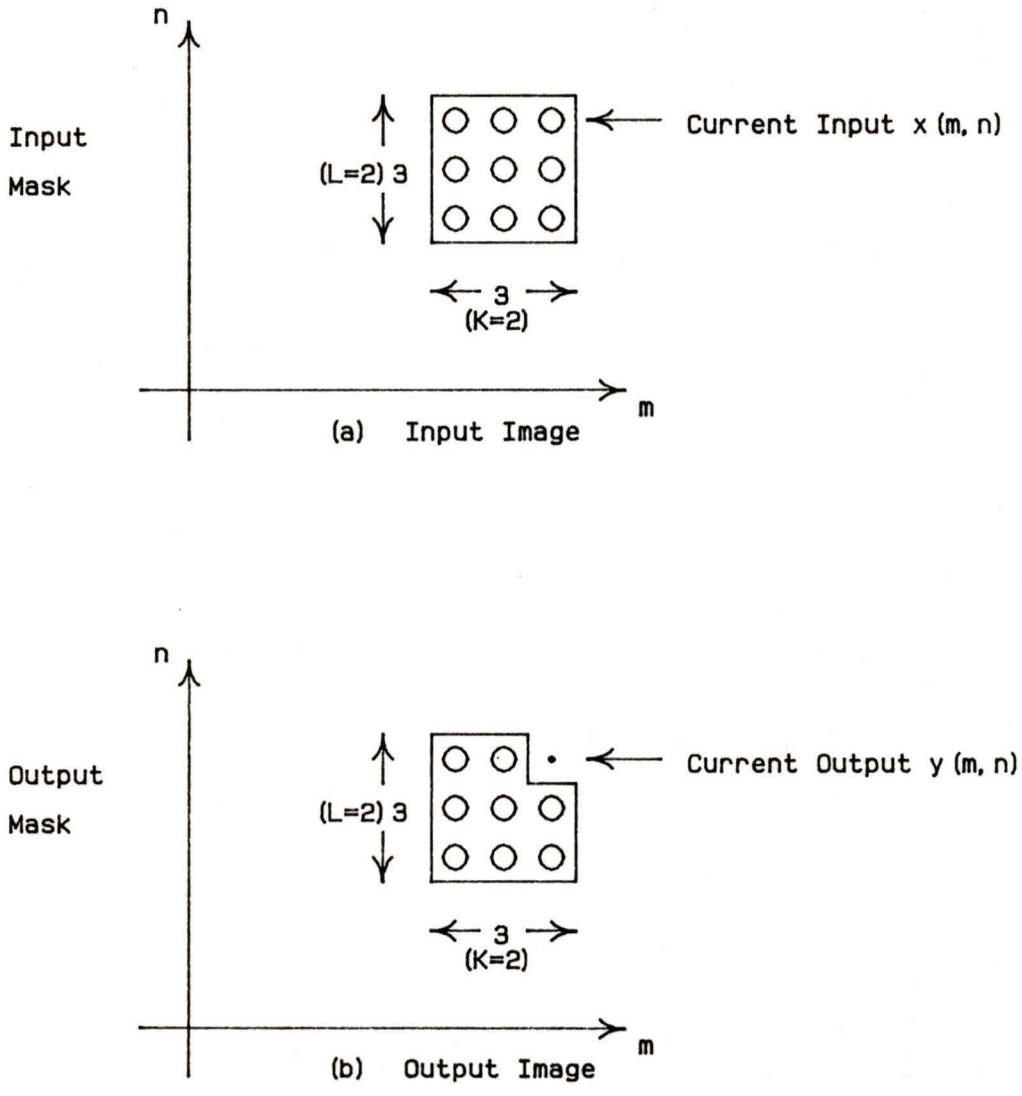


Figure 2.5 Mask for 2x2 Order QP Filter

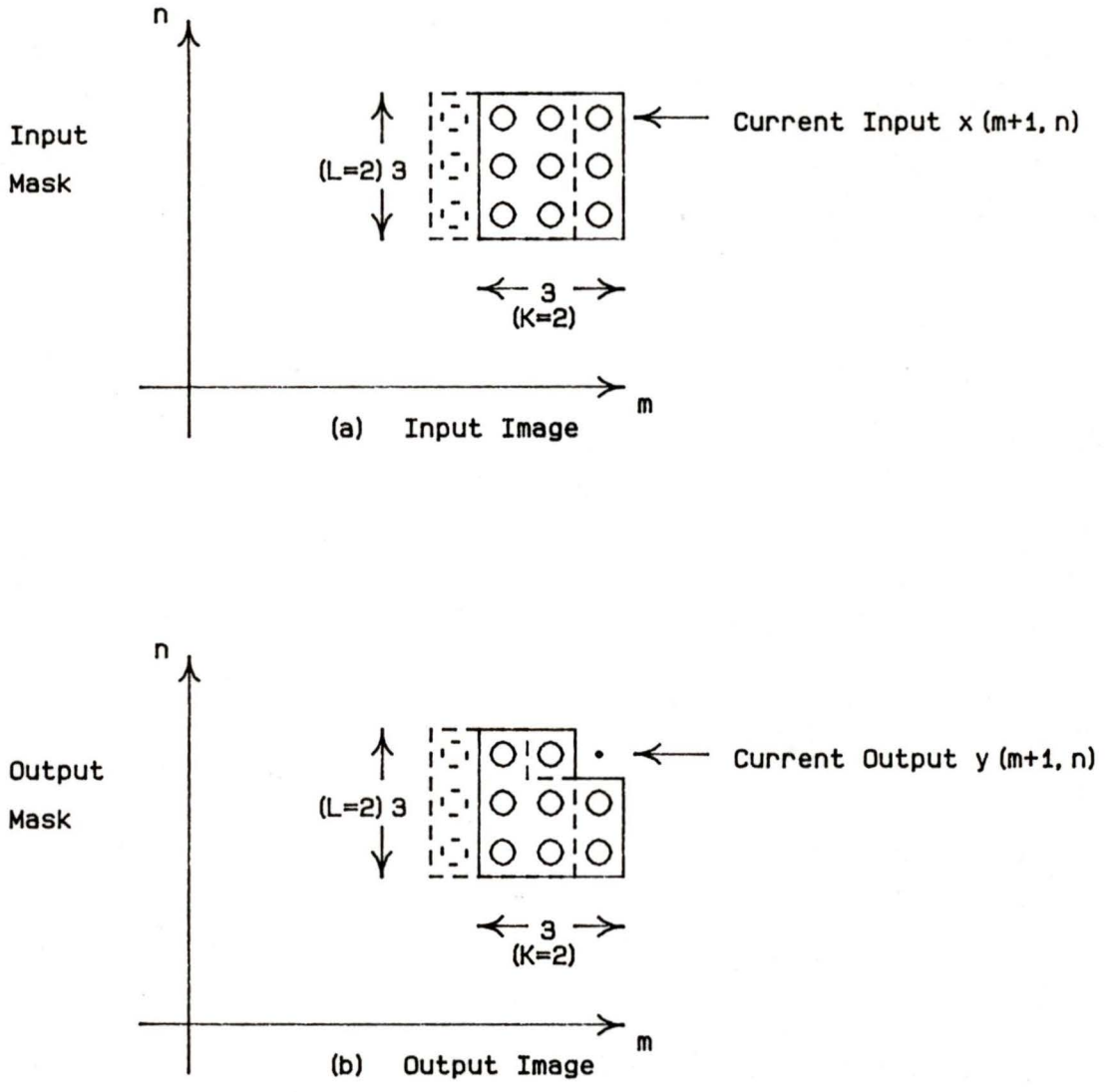


Figure 2.6 Masks for Computation of y (m+1, n)

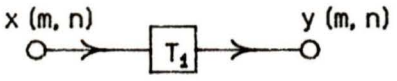
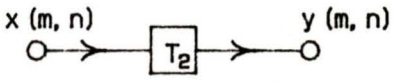
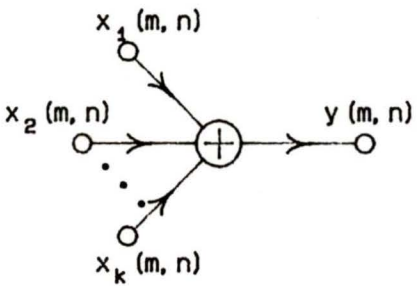
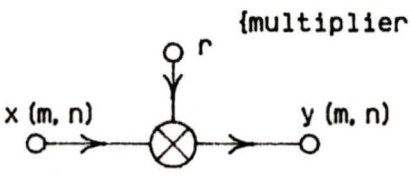
	Symbol	Equation
Unit Row Shift		$y(m, n) = x(m-1, n)$
Unit Column Shift		$y(m, n) = x(m, n-1)$
Adder		$y(m, n) = \sum_{i=1}^{k} x_i(m, n)$
Multiplier		$y(m, n) = r\ x(m, n)$

Figure 2.7 2D Digital Filter Elements

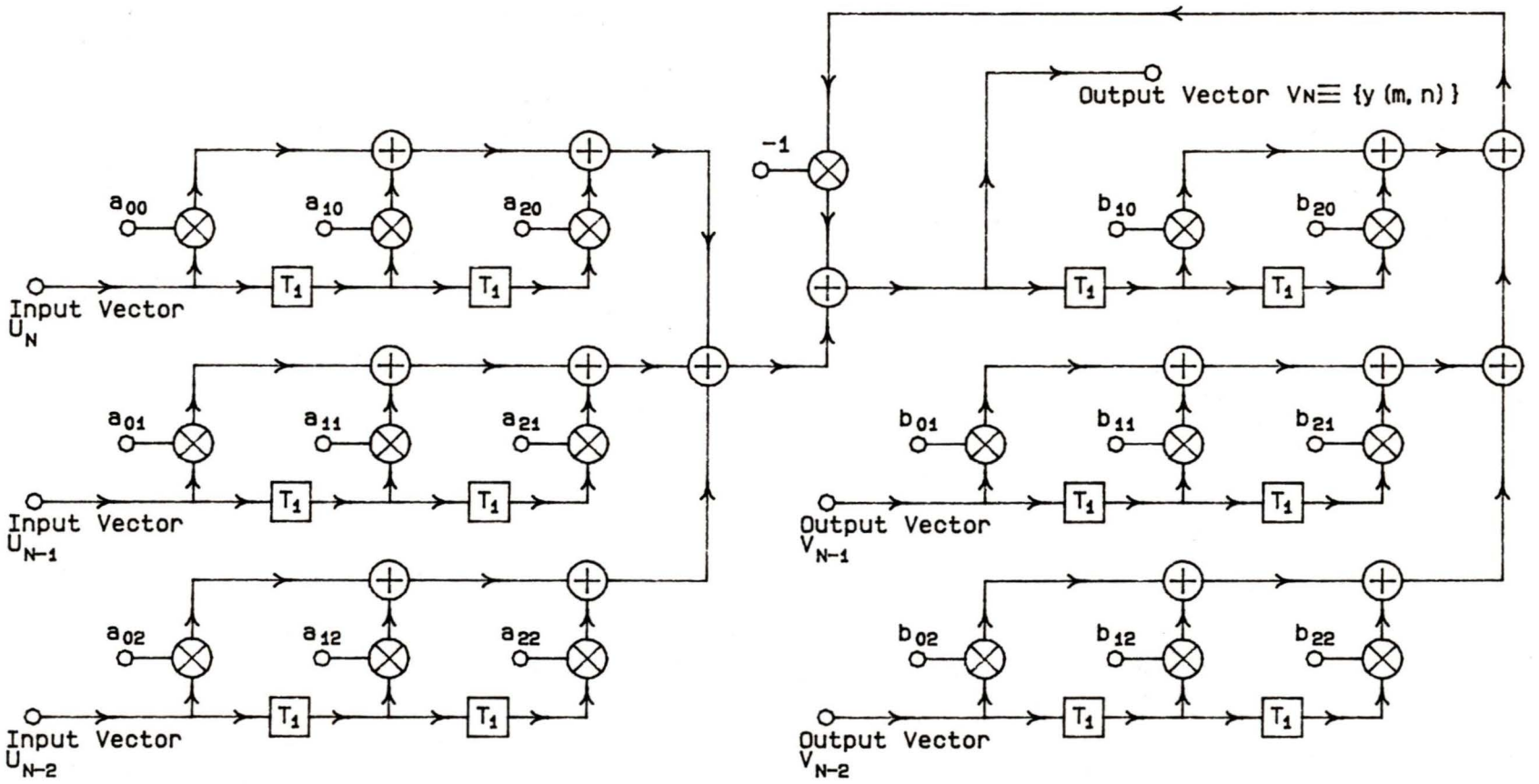


Figure 2.8 2 x 2 Order 2D Filter Structure which Processes Input and Output Row Vectors

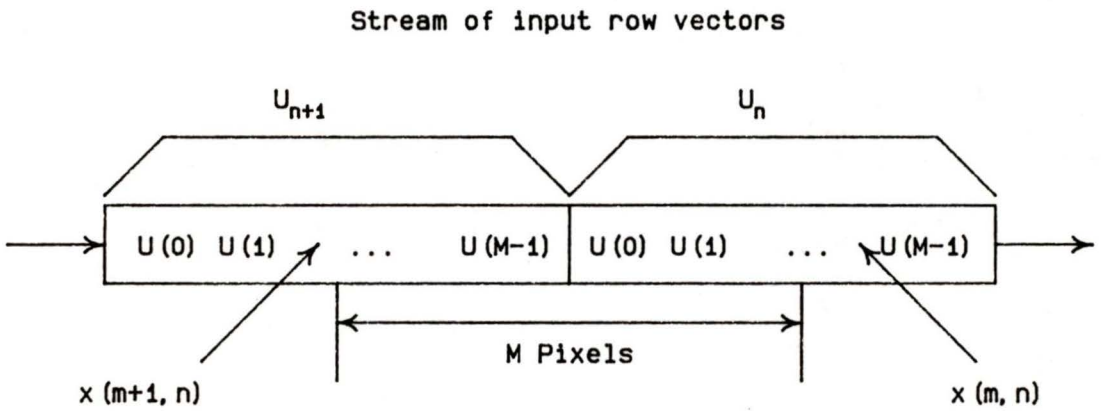
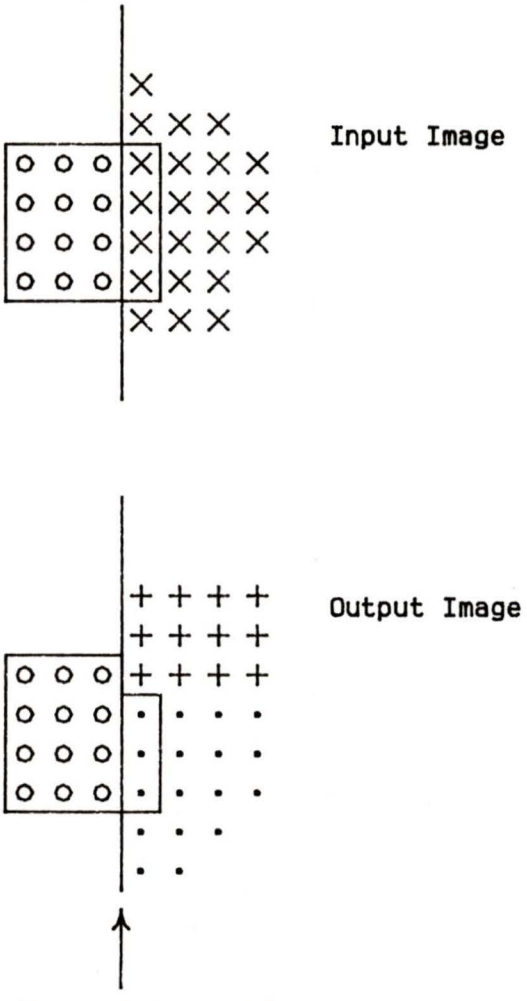


Figure 2.9 Separation Between Adjacent Column Pixels x (m, n) and x (m+1, n) in Adjacent Row Vectors U_n and U_{n+1}



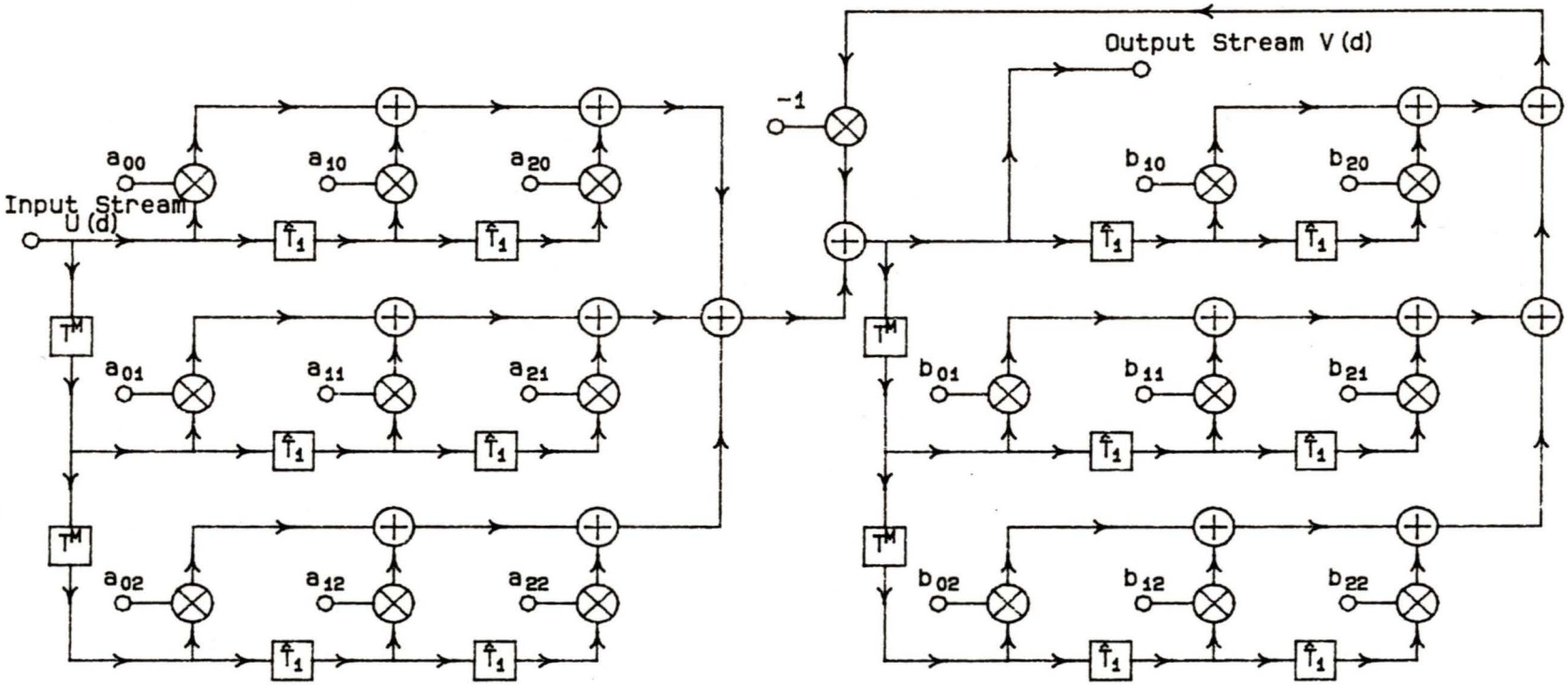
× Input Image Pixels

• Computed Output Image Pixels

\+ Output Image Pixels to be Computed

o Mask Element Values Initialized to Zero

Figure 2.10 Mask Initialization for Row-Recursive Processing of 2D Filters



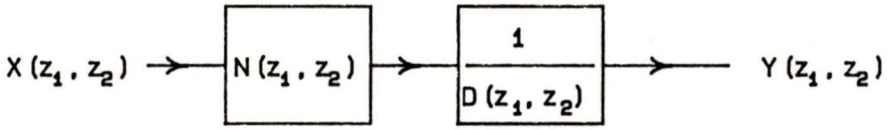
Input = U(d) = x(m, n) m = 0, 1, 2 ... M; n = 0, 1, 2 ... N

Output = V(d) = y(m, n) d = m + n x M

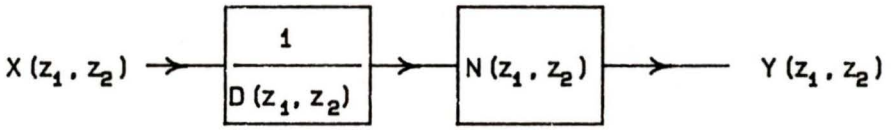
Column Delay $r(d) \rightarrow T^k \rightarrow s(d)$ $s(d) = r(d-k)$

Row delay and Row Initialization $w(d) \rightarrow T_1 \rightarrow v(d)$ $v(d) = w(d-1)$; $m \neq 0$; $v(d) = 0$; $m = 0$ (Start of a Row)

Figure 2.11 Direct Form Spatially Bounded Row-Recursive 2D Filter



(2.12 a)



(2.12 b)

Figure 2.12 Cascade Realization of the Transfer Function $H(z_1, z_2)$

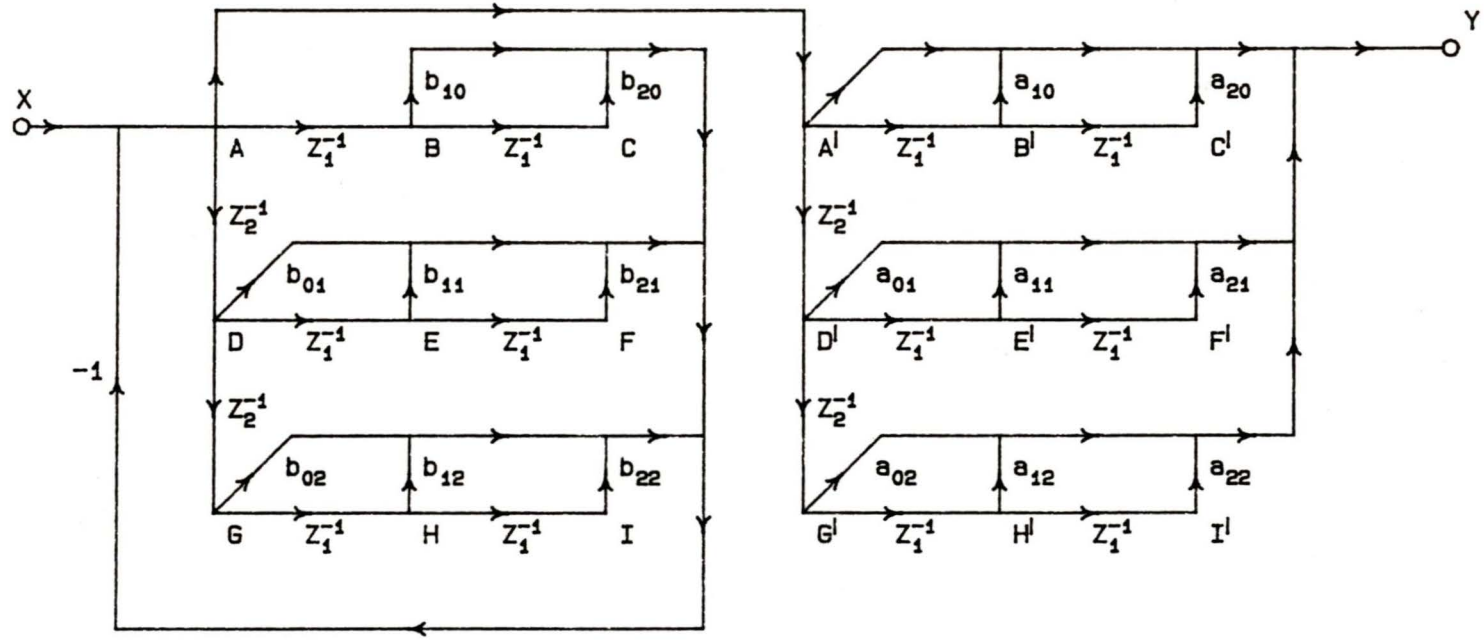


Figure 2.13 2 x 2 Order 2D Filter Direct Form Structure

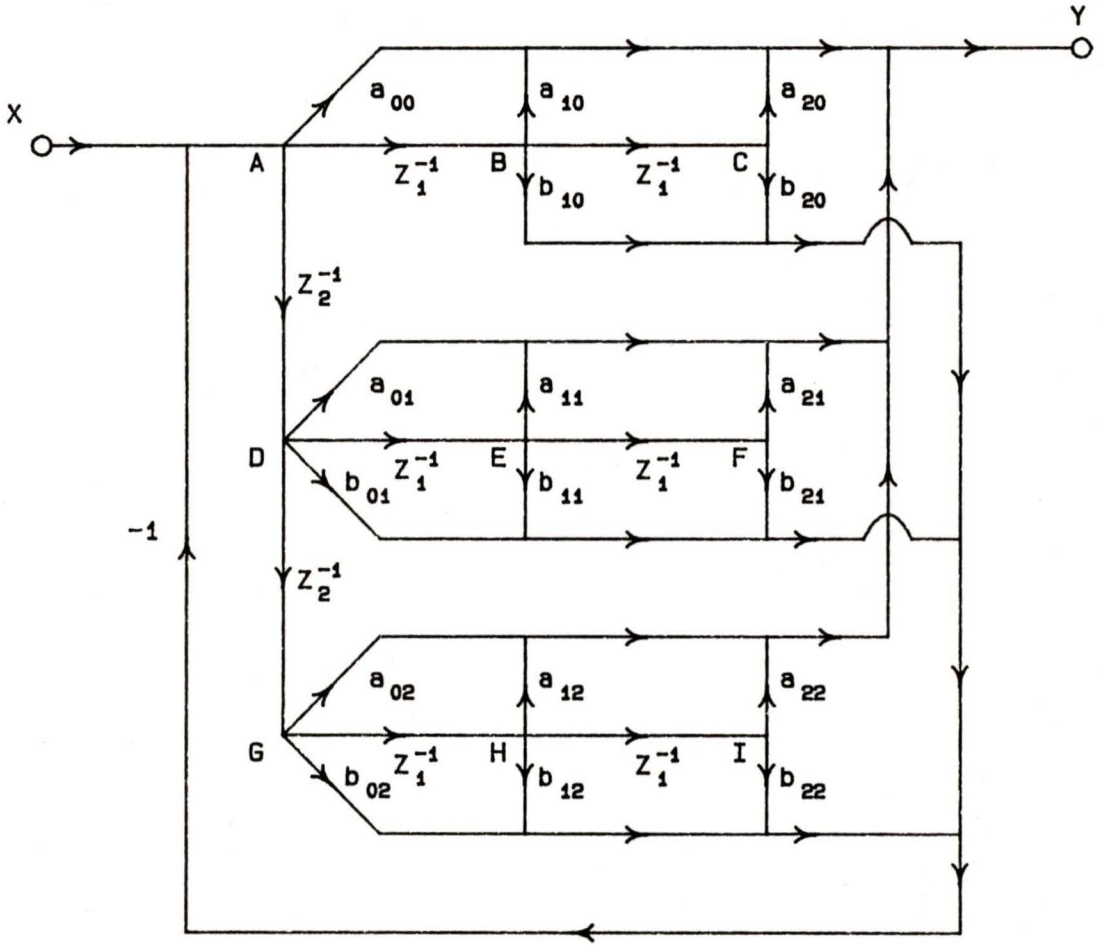


Figure 2.14 2 x 2 Order 2D Reduced Column Delay Direct Form Structure

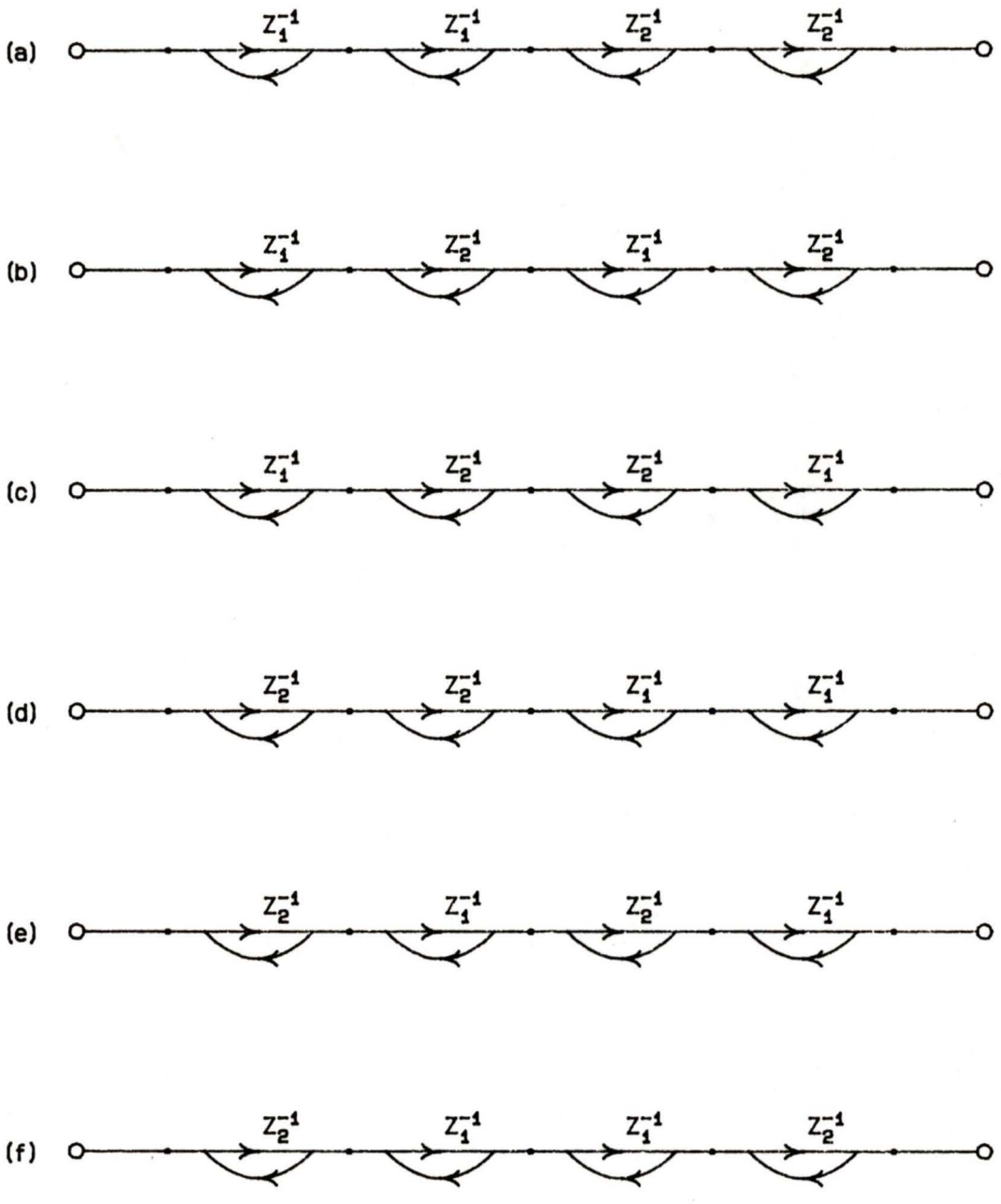


Figure 2.15 Six Ways to Calculate $\hat{Z}_1^{-1}$ $\hat{Z}_2^{-1}$

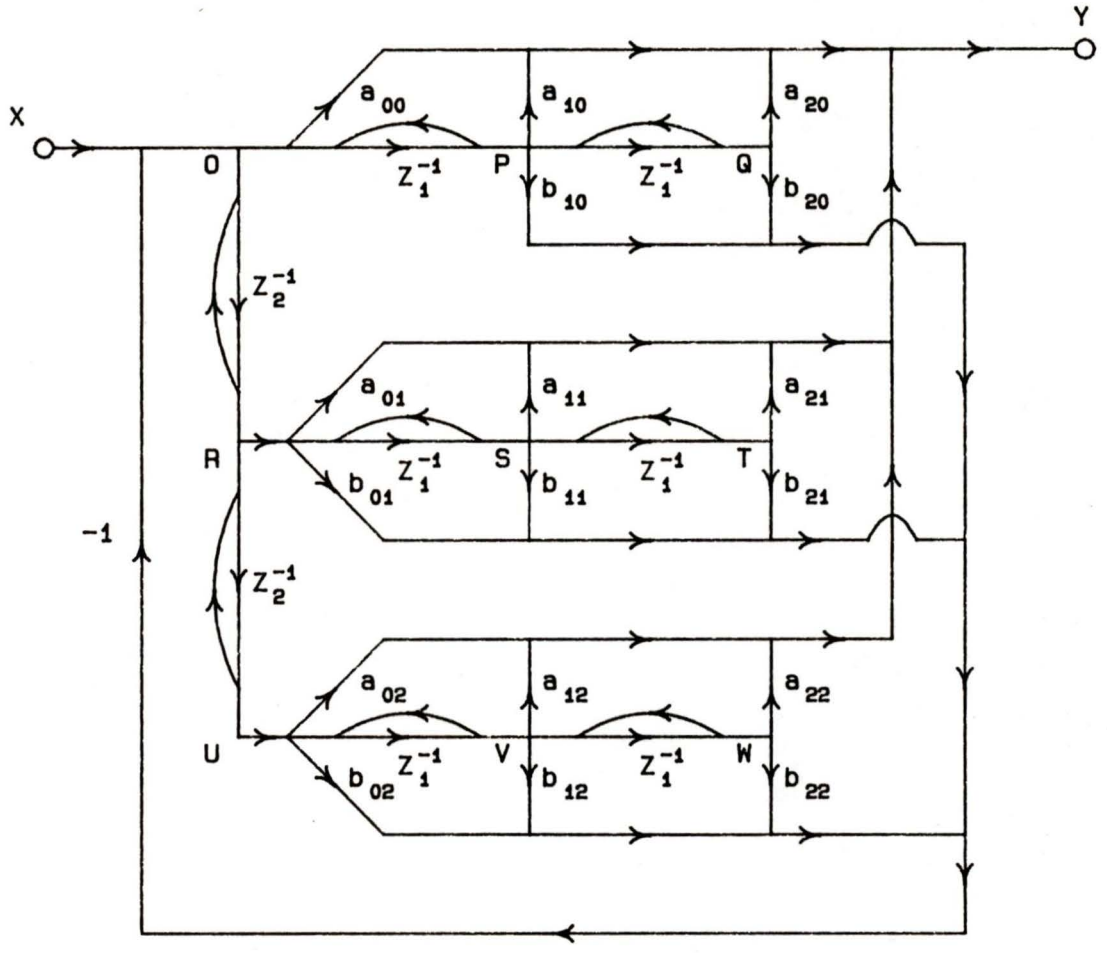
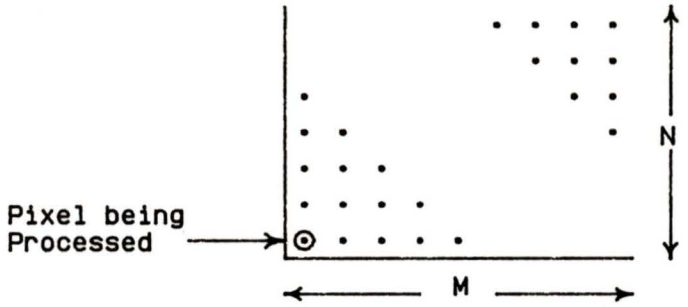
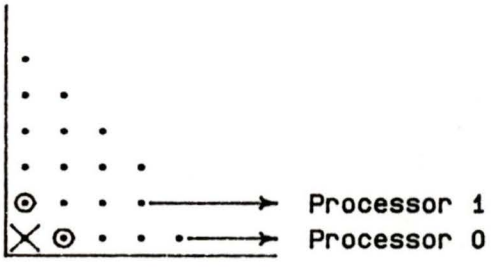


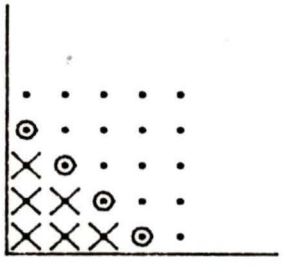
Figure 2.16 2 x 2 Order 2D Spatial Integrator Structure



a) 1 Pixel Processed



b) 2 Pixels Processed



c) Diagonal Processing

• Pixels to be Processed

⊙ Pixels being Processed

× Computed Pixels

Figure 2.17 Multi-processor Realization

3. AN INVESTIGATION OF THE ARITHMETIC FOR 2D RECURSIVE FILTERS

3.1. INTRODUCTION

The objective of this chapter is to compare the non-ideal performance of the Spatial Integrator 2D filter with that of the Direct Form 2D filter. The previous work by Bruton and Strecker [5] contains sensitivity analysis of the Direct Form and Spatial Integrator filters. In this chapter, the spatial domain performance of the Direct Form and Spatial Integrator structures are examined in detail.

Recursive (IIR) digital filters are more complex than non-recursive (FIR) filters. In the design and implementation of 2D IIR filters, issues such as stability and the direction of recursion must be addressed. With FIR 2D filters these issues are not a concern. Despite these problems, IIR filters are often favored over FIR filters *because of the reduced number of arithmetic operations.* The use of previous outputs $y(m-k, n-l)$ in the calculation of the current output $y(m, n)$ presents enough information so that an IIR filter will have a similar magnitude transfer function with fewer coefficients than a FIR. When processing speed is a design criterion, and therefore the rate at which output $y(m, n)$ is calculated, IIR filters can meet the magnitude transfer function specifications with the least number of multiplier

coefficients.

The type of arithmetic used can also affect the speed at which the output $y(m,n)$ is calculated. Addition or multiplication of floating-point numbers involves the manipulation of two fixed-point values. Therefore, an arithmetic operation using floating-point numbers will take significantly longer than an arithmetic operation using fixed-point numbers.

The number of significant digits used to represent the multiplier coefficients also affects the processing speed. There may exist trade-offs between the number of arithmetic operations and the use of double or single precision arithmetic. This trade-off needs to be investigated before deciding on the filter implementation.

The Direct Form structure and the Spatial Integrator structure are examined in this chapter. The analysis involves filtering the Swept Frequency Test Image described in section 3.2 (Figure 3.1b and Figure 3.1c) to determine the effects of reducing the number of bits used to represent the multiplier coefficients. The analysis is performed with both floating-point and fixed-point representations of the multiplier coefficients. The fixed-point representation is also examined to determine the number of bits necessary to prevent overflow of internal signals. In section 3.5, these results are used to describe a 16-bit microprocessor implementation using the Spatial Integrator structure.

3.2. SPATIAL DOMAIN VERIFICATION OF THE 2D FILTER RESPONSE USING A SWEPT FREQUENCY SINUSOID TEST IMAGE

The steady state frequency response of 1D digital filters is tested by measuring the steady state output of the filter for a set of sampled sinusoids at the input. An alternative method of measuring the frequency response is to measure the output response of the filter to a swept frequency sinusoid. In the 2D case, the determination of the amplitude response is required as a function of frequency and direction. This suggests a *2D Swept Frequency Sinusoid Test Image* having all frequencies and all directions as shown in Figure 3.1a, Figure 3.1b and Figure 3.1c.

The Swept Frequency Test Image (Figure 3.1a, Figure 3.1b, Figure 3.1c) incorporates a linear sweep of the spatial frequency in all radial directions. If the frequency in the radial direction varies sufficiently slowly, the Test Image is an indication of the 2D steady state frequency response. The proposed Swept Frequency Test Image is given by

$$x(i,j) = \frac{p_{\max}}{2} + \frac{p_{\max}}{2} \times \sin\left(\left(f_{\min} + \frac{2(f_{\max} - f_{\min})}{x_{\max}} radius\right) \cdot radius\right) \tag{3.1}$$

where $p_{\max}$ is the the maximum value of $x(i,j)$. It is observed that $x(i,j)$ is chosen such that $(0 \leq x(i,j) \leq p_{\max})$. $f_{\min}$ is the initial frequency at

the center of the image and f_{max} is the frequency at the locations A and B of Figure 3.1b. x_{max} is the number of pixels per row of the image and the value of the *radius* in Equation 3.1 is given by

$$radius = \sqrt{(\frac{x_{max}}{2} - i)^2 + (\frac{x_{max}}{2} - j)^2} \tag{3.2}$$

The parameters for the Swept Frequency Test Image in Figure 3.1 are

$$f_{min} = 0.0$$
$$f_{max} = 0.75$$
$$x_{max} = 255$$
$$p_{max} = 255$$

Figure 3.2a is a (64 x 64) pixel square window taken from the upper left hand corner of the Test Image (Figure 3.1). (The bottom left hand corner of Figure 3.2a is the point $x(32,160)$.) Examining this small portion of the Swept Frequency Test Image, around any point $x(m,n)$ the spatial frequency in the neighborhood of this point is approximately constant and in a single direction. Figure 3.2b is the magnitude response of the 2D FFT of this segment. The single impulses in the upper left hand quadrant and lower right hand quadrant indicates the presence and direction of the 2D sine wave of Figure 3.2a. The impulse nature of the 2D FFT (Figure 3.2b) shows that the segment (Figure 3.2a) is a good approximation to a spatial sine wave in a single direction.

Because the complete Swept Frequency Test Image can be considered as an ensemble of slowly changing sine waves, the output of a 2D filter indicates the steady state frequency response of the filter in all directions and for all frequencies between f_{min} and f_{max} of the original Test Image. For example, Figure 3.5 is the output response to the Swept Frequency Test Image (Figure 3.1c) of a 2D circularly symmetric *lowpass* filter to be specified and described in section 3.3.1 with a cutoff frequency of 0.15 π radians. Comparison of Figure 3.5 with Figure 3.1c reveals that the low frequency components in the center of the image are similar in amplitude whereas the higher frequency components along the top, bottom, and edges of Figure 3.5 (points G,H,I, and J) are much smaller than the edges of Figure 3.1c (points C,D,E, and F) because they are in the stop band.

The advantages of using the Swept Frequency Test Image (Figure 3.1), as opposed to computing sensitivities or the frequency response are

(1) by processing only *one* input image, frequencies of a specific interest can be tested in all directions.

(2) specific frequencies of interest can be chosen with infinite resolution allowing narrow passband and stopband transition regions to be investigated.

(3) non-linearities such as underflow and overflow are easily investigated by scaling and their effects in

the output image can be observed directly.

3.3. USE OF THE SWEPT FREQUENCY TEST IMAGE TO COMPARE DIRECT FORM AND SPATIAL INTEGRATOR FORM FILTER RESPONSES

3.3.1. Introduction

An experimental comparison of the Spatial Integrator and Direct Form SFG structures will be described in order to determine the effects of multiplier coefficient quantization. It is anticipated that multiplier coefficient quantization will effect the magnitude response and perhaps even the stability of the filtering algorithms.

The Swept Frequency Test Image is used to measure the effect of quantization on the magnitude response by comparing the output of the filter using quantized multiplier coefficients with the output of the same filter using double precision floating-point coefficients. Changes in the stability margin can often be observed in the output response Y of a quantized filter in the form of spurious non-ideal oscillations. This method of testing the frequency response is particularly advantageous for the detection of non-linearities since the filter processing is performed on an actual image rather than by nondirect mathematical methods.

The *Test Filter* used in the experiment has a lowpass (3 x 3) order circularly symmetric response [8] with a radian cutoff frequency of 0.15 π

radians. For the Direct Form structure, the transfer function is given by

$$H(z_1, z_2) = \frac{\sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}\ z_1^{-i} z_2^{-j}}{\sum_{i=0}^{3}\sum_{j=0}^{3} b_{ij}\ z_1^{-i} z_2^{-j}} \tag{3.3}$$

where the values of a_{ij} and b_{ij} are given in Table 3.1a.

For the Spatial Integrator structure the transfer function may be written as [5]

$$H(z_1, z_2) = \frac{\sum_{i=0}^{3}\sum_{j=0}^{3} c_{ij}\ \frac{z_1^{-i}}{(1-z_1^{-1})^i}\ \frac{z_2^{-j}}{(1-z_2^{-1})^j}}{\sum_{i=0}^{3}\sum_{j=0}^{3} d_{ij}\ \frac{z_1^{-i}}{(1-z_1^{-1})^i}\ \frac{z_2^{-j}}{(1-z_2^{-1})^j}} \tag{3.4}$$

where the values of c_{ij} and d_{ij} are given in Table 3.1b.

The 2D magnitude transfer function of the Test Filter used for these experiments is shown in Figure 3.3. The transfer function coefficients are determined using the Ramamoorthy/Bruton method [3,4].

The result of using double precision arithmetic and the Direct Form structure with lowpass coefficients from Table 3.1a to filter the Swept Frequency Test Image is shown in Figure 3.5. The output from processing the Swept Frequency Test Image using double precision arithmetic and the Spatial Integrator structure with the multiplier coefficients from Table 3.1b is shown in Figure 3.6. A comparison of Figures 3.5 and 3.6 shows little

difference, as expected, between the two output images since they have the same magnitude transfer function and double precision arithmetic is used for the computation.

Experiments will be described for both floating-point and fixed-point number representations under multiplier coefficient quantization in order to determine the best hardware implementation.

3.3.2. The Transfer Function Quantization Experiment

Quantized multiplier coefficient output responses Y are compared to the double precision output response (Figure 3.5) since the double precision output response is assumed to be of infinite precision. In addition to the subjective visual interpretation, an error function E is used to quantify the error in the output image Y under quantized coefficients. The Error E is defined as

$$E = \frac{\sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \left| y_d(k,l) - y_q(k,l) \right|}{M \times N} \tag{3.5}$$

where $y_d(k,l)$ is an output pixel of the double precision image, $y_q(k,l)$ is an output pixel of the Test Image with quantized coefficients, M is the maximum number of pixels per row and N is the maximum number of pixels per column of both images. The error E may be physically interpreted as the *average error per pixel* over the entire image. A value of E that is less than

unity is highly acceptable.

3.3.3. The Floating-point Numerical Representation

A floating-point number consists of two fixed-point numbers, one specifying the mantissa W and the other specifying the exponent V. Figure 3.4 shows the floating-point format used for double precision numbers on the Charles River Data Systems model PB07-D microcomputer that was used for the analysis. The format consists of 1 sign bit, 8 bits for the exponent and 55 bits for the mantissa. The computer uses a hidden bit in its representation of the mantissa W, that is the mantissa of a floating-point number is represented as $1.b_{54}b_{53}b_{52}b_{51}\cdots b_0$ where b_{54} to b_0 are the binary digits. Except for the number 0.0 which is represented by an exponent V of zero, the normalization of the mantissa W is performed such that a binary floating-point number will always start with a 1. Therefore, the leading 1 is excluded from the floating-point representation and is called a hidden bit. If a mantissa W has been quantized to 15 bits b_{54}–b_{30}, it is actually represented by 16 bits due to the hidden bit. The mantissa W is composed of 1 hidden bit and 15 bits of data b_{54}–b_{30} from the floating-point representation.

On the Charles River Data Systems computer the exponent V is a power of 2. The computer represents the exponent in "excess 128" notation which means that if the exponent is between 1 and 255, one obtains the

correct value of the exponent by taking the given unsigned 8-bit value and subtracting 128. If the exponent is zero the value of the floating-point number is zero.

For example, the decimal equivalent of the double precision "excess 128" number

01101000110010000000000000000000000000000000000000000000000000000

is

$$1(hidden\ \ bit) + 2^{-1} + 2^{-4} \times 2^{(2^7 + 2^6 + 2^4 + 1 - 128)} = 1.5625 \times 2^{81}$$

$$\approx 3.77789 \times 10^{24}$$

3.4. 2D FILTERING USING FLOATING-POINT ARITHMETIC

3.4.1. Multiplier Coefficient Quantization Effects

The Error function E versus the mantissa word length W_{length} for both the Direct Form and the Spatial Integrator SFG structures are shown in Figure 3.7, Table 3.2. It can be observed from Figure 3.7, an error $\log_{10}E \approx 0.5$ occurs using a mantissa word length $W_{length} = 3$ for the Spatial Integrator structure whereas the Direct Form Structure SFG requires a $W_{length} = 12$. This has significant hardware implications [5]. For the (3 x 3) order Test Filter, *the mantissas W of the Spatial Integrator multiplier coefficients can be implemented with less than one third the number of bits that are required for the mantissas of the Direct Form multiplier coefficients.*

The results in Figure 3.7 indicate that the 3-bit Spatial Integrator implementation has less quantization error than a 9 bit Direct Form implementation.

Tables 3.3 and 3.4 contain the quantized multiplier coefficients used to obtain the output responses for the 3-bit Spatial Integrator implementation in Figure 3.8 and the corresponding 9-bit Direct Form implementation in Figure 3.9.

The result of filtering the test image using the Spatial Integrator structure with 2-bit mantissa multiplier coefficients is shown in Figure 3.10. The oscillations in the upper right hand corner of the image indicates that the filter has become unstable.

3.4.2. Precision of the Internal Signals

The above analysis was performed using double precision arithmetic for all internal calculations. (i.e. calculation and storage of z_1^{-1} and z_2^{-1} delay elements.) This essentially eliminates any effects of round-off noise from the signals. It is expected that if single precision arithmetic is used for the internal calculations, round-off error would become significant and stability would become a problem. This could be overcome by increasing the length of the multiplier coefficient mantissas.

3.5. 2D FILTERING USING FIXED-POINT ARITHMETIC

3.5.1. Introduction

A fixed-point number is a simple integer. Usually both positive and negative numbers are required and a twos complement representation is used. The range of values that a fixed-point number can take depends upon its word length. For a 32 bit twos complement number the range is -2^{31} to $2^{31}-1$ or -2,147,483,648 to 2,147,483,647. Compared to floating-point arithmetic, fixed-point arithmetic is simple because less hardware and/or fewer programming steps are required to manipulate integers. The disadvantage of fixed-point arithmetic is that significant overflows may occur on multiplication, addition, or subtraction and round-off errors may occur on division operations; that is, the dynamic range of the algorithm deteriorates significantly relative to floating-point implementation.

3.5.2. Fixed-point Arithmetic SFG Structures

The structure for a (3 x 3) order Direct Form fixed-point implementation is shown in Figure 3.11 where A_{ij} and B_{ij} coefficients are integers R and the N and M division operations are powers of 2. (Powers of 2 were chosen so that a simple right shift may be used for the division.) The points marked $S_0 - S_{15}$ and D_1, D_2 on Figure 3.11 are of interest since $S_0 - S_{15}$ are the row and column shifts which must be saved for the calculation of the adjacent pixels and D_1, D_2 are the values of the critical nodes in the SFG

where the accumulation of feedforward and feedback variables occur. The minimum and maximum values of $S_0 - S_{15}$ determines the word length for internal storage and the minimum and maximum values of D_1 and D_2 determines the number of bits required to prevent overflow.

Table 3.8a contains the values of the A_{ij}, B_{ij}, N, and M coefficients for a multiplier word length R_{length} of 16 bits using the Direct Form structure of Figure 3.11.

3.5.3. Quantization Effects

The structure for a fixed-point Spatial Integrator implementation is shown in Figure 3.12. The Error Function E versus the multiplier coefficient word length R_{length} is shown in Figure 3.13 (Table 3.5) for both the Direct Form structure in Figure 3.11 and the Spatial Integrator Structure in Figure 3.12. The Error Function was computed using Equation 3.5 with the same double precision floating-point Test Image output response as the comparison.

3.5.4. Overflow Effects

For the Direct Form structure the minimum and maximum values for each of the S_i points will be the same since each S_i value comes from a simple row shift or column shift of S_0. With the Spatial Integrator structure, each internal signal S_i will be different since each is a sum of all the row or

column input values to that point. Table 3.7 is a comparison of the $S_0 - S_{15}$ values of Figures 3.11 and 3.12 for a transfer function multiplier coefficient word length R_{length} of 16 bits. From this Table, *a Direct Form structure requires 17 bits to implement each of the saved values and Spatial Integrator structure requires 10 bits for S_0 to 17 bits for S_{15}.*

A comparison of the D_1 and D_2 values versus the multiplier coefficient word length for the two structures is shown in Table 3.6. From this table, *a transfer function implemented using 16-bit coefficients would require at least 34 bits to prevent overflow for the Direct Form structure and 31 bits for the Spatial Integrator structure.*

3.6. SUMMARY AND IMPLICATIONS FOR HARDWARE IMPLEMENTATION

3.6.1. Multiplier Coefficient Arithmetic

A Swept Frequency Test Image that indicates the magnitude frequency response of a 2D filter is presented. This Test Image is used to evaluate two structures to implement the same transfer function using both floating-point and fixed-point arithmetic. The results show that for a (3 x 3) order Test Filter, the Direct Form structure requires at least 9 bits for the transfer function multiplier coefficients mantissas W when using floating-point arithmetic and the Spatial Integrator structure requires only 3 bits for the multiplier coefficient mantissas W.

Results using fixed-point arithmetic indicate that the Direct Form structure of Figure 3.11 requires multiplier coefficients of at least 14 bits and the Spatial Integrator structure of Figure 3.12 requires multiplier coefficients of at least 9 bits.

When converting the filter network into dedicated hardware, a number of issues must be addressed. The type of arithmetic must be determined. From Table 3.2, the Error function E of a *floating-point* implementation of the Test Filter using the Direct Form structure with multiplier coefficient mantissas of 16 bits is 0.546967. This means that the output response of the Test Filter to the Test Image (Figure 3.1) differs from the ideal infinite precision output by an average of 0.546967 per pixel in a signal having 256 possible levels. Therefore, little is gained by increasing the precision of the mantissas of the multiplier coefficients beyond 16 bits in the case of floating-point arithmetic.

From Table 3.5, a *fixed-point* Direct Form structure with an acceptable Error function E less than 1 can be implemented with multiplier coefficients of 20 bits. The extra hardware or software required to manipulate the two integers of floating-point arithmetic can be eliminated if the fixed-point word length is at least 20 bits and the internal signals do not overflow.

3.6.2. Filter SFG Structure

The next consideration is the SFG structure that is the algorithm used to implement the transfer function. From Table 3.5, a fixed-point implementation of the Test Filter using the Spatial Integrator structure and with an Error function E less than 1 can be implemented with multiplier coefficients of *16 bits*. The Spatial Integrator structure has $N(N+2)$ more additions per pixel than the $2N(N+2)$ required for the Direct Form structure for an $N \times N$ order filter. The trade-off is between the extra hardware or time taken for implementation of the algorithm using the longer multiplier coefficients for the Direct Form structure compared with the time it takes for the extra additions using the Spatial Integrator structure.

3.6.3. Dynamic Range and Overflow

There are other considerations before deciding the type of SFG structure. From Table 3.6, for a 20-bit multiplier ($R_{length} = 20$) Direct Form structure the maximum value of D_1 or D_2 is 1.46×10^{11} which requires 39 bits to implement in twos complement arithmetic. The 16-bit multiplier coefficient ($R_{length} = 16$) Spatial Integrator structure (Table 3.6) has a maximum value of D_1 or D_2 of 5.71×10^8 which requires 31 bits for its implementation.

The word length of the internal signals required for the calculation of adjacent pixels is another consideration. (The $S_0 - S_{15}$ values of Figure 3.11

and 3.12.) From Table 3.7, all these signals for the Direct Form structure require 17 bits for their implementation in twos complement arithmetic. For the Spatial Integrator structure points S_0 to S_{14} require 15 bits or less with only point S_{15} requiring 17 bits.

3.6.4. Comparison of Direct Form and Spatial Integrator Structures

Summarizing, a Direct Form implementation of the Test Filter using fixed-point arithmetic with an Error function less than unity requires:

(1) $R_{length} = 20$, 20-bit multiplier coefficients.

(2) Maximum value of D_1 and $D_2 = 1.46 \times 10^{11}$ requiring 39-bit arithmetic to prevent overflow.

(3) 17 bits to save the internal signals S_i for the calculation of adjacent pixels.

(4) 30 additions per pixel for (3 x 3) order Test Filter

(5) 31 multiplications per pixel for (3 x 3) order Test Filter

A Spatial Integrator implementation of the Test Filter using fixed-point arithmetic with an Error function less than unity requires:

(1) $R_{length} = 16$, 16-bit multiplier coefficients.

(2) Maximum value of D_1 and $D_2 = 5.71 \times 10^8$ requiring 31-bit arithmetic to prevent overflow.

(3) 15 bits for points $S_0 - S_{14}$ of Figure 3.12 and 17 bits for point S_{15} for the calculation of adjacent pixels.

(4) 45 additions per pixel for (3 x 3) order Test Filter

(5) 31 multiplications per pixel for (3 x 3) order Test Filter

From the above results, for the Spatial Integrator implementation most of the arithmetic operations can be performed with a 16 by 16 bit multiplier and a 32-bit accumulator. This suggests using a 16-bit microprocessor to perform the calculations, since most 16-bit microprocessors have 16 by 16 bit multiply instructions that return a 32-bit result and 32-bit addition and subtraction instructions. The only signal for the Spatial Integrator structure that requires a 17-bit integer to prevent overflow is S_{15}. This signal can be scaled down to 16 bits using a slight modification to the structure of Figure 3.12. This modification is shown in Figure 3.13 for the implementation of the (3 x 3) order Test Filter.

The Direct Form structure is rejected because double precision 32-bit arithmetic is required for the internal signals S_i, for the multiplier coefficients and for the accumulation at D_1 and D_2 to prevent overflow.

The increased number of arithmetic operations required for the Spatial Integrator structure is not a significant problem and takes less time to execute than the double precision arithmetic required for the Direct Form structure. It is estimated that a double precision add instruction takes 5 times longer than a single precision add instruction, and a double precision multiply instruction takes 40 times longer than a single precision multiply instruction. Furthermore, if a single precision add instruction and a double precision add instruction execute at the same rate then it will take *over 18 times* more arithmetic instructions to implement the Direct Form structure than to implement the Spatial Integrator structure. Therefore, the Spatial Integrator SFG is chosen as a superior structure.

Table 3.1a Coefficients for (3 x 3) - Test Filter Direct Form Structure	
Numerator Coefficients	
a_{00} = .704635728E-04	a_{01} = .211390718E-03
a_{10} = .211390718E-03	a_{11} = .634172155E-03
a_{20} = .211390718E-03	a_{21} = .634172155E-03
a_{30} = .704635728E-04	a_{31} = .211390718E-03
a_{02} = .211390718E-03	a_{03} = .704635728E-04
a_{12} = .634172155E-03	a_{13} = .211390718E-03
a_{22} = .634172155E-03	a_{23} = .211390718E-03
a_{32} = .211390718E-03	a_{33} = .704635728E-04
Denominator Coefficients	
b_{00} = .100000000E+01	b_{01} = -.210005616E+01
b_{10} = -.210005616E+01	b_{11} = .449585428E+01
b_{20} = .157007849E+01	b_{21} = -.342860032E+01
b_{30} = -.404931745E+00	b_{31} = .899436503E+00
b_{02} = .157007849E+01	b_{03} = -.404931745E+00
b_{12} = -.342860032E+01	b_{13} = .899436503E+00
b_{22} = .266303615E+01	b_{23} = -.707247813E+00
b_{32} = -.707247813E+00	b_{33} = .188199325E+00

Table 3.1b Coefficients for (3 x 3) - Test Filter Spatial Integrator Structure	
Numerator Coefficients	
c_{00} = .704635728E-04	c_{01} = .422781437E-03
c_{10} = .422781437E-03	c_{11} = .253668862E-02
c_{20} = .845562874E-03	c_{21} = .507337724E-02
c_{30} = .563708582E-03	c_{31} = .338225149E-02
c_{02} = .845562874E-03	c_{03} = .563708582E-03
c_{12} = .507337724E-02	c_{13} = .338225149E-02
c_{22} = .101467545E-01	c_{23} = .676450299E-02
c_{32} = .676450299E-02	c_{33} = .450966866E-02
Denominator Coefficients	
d_{00} = .100000000E+01	d_{10} = .899943841E+00
d_{10} = .899943841E+00	d_{11} = .895517319E+00
d_{20} = .369966170E+00	d_{21} = .372838266E+00
d_{30} = .650905844E-01	d_{31} = .619060537E-01
d_{02} = .369966170E+00	d_{03} = .650905844E-01
d_{12} = .372838266E+00	d_{13} = .619060537E-01
d_{22} = .151848999E+00	d_{23} = .258068608E-01
d_{32} = .258068608E-01	d_{33} = .444766072E-02

Table 3.2 Error Function Comparison
Floating-point Arithmetic

Number of Mantissa bits	Direct Form Structure	Integrator Form Structure
36	0.0	0.0
32	.305176E-04	0.0
24	.427246E-02	0.0
18	.354172	.244141E-03
16	.546967	.579834E-03
14	6.983	.161743E-02
12	3.17805	.256958E-01
10	21.619	.177017
9	5.82727	.472412E-01
8	120.5 unstable	.973663E-01
6		1.68269
4		2.57484
3		3.69315
2		36.24 unstable

Table 3.3 Quantized Coefficients for (3 x 3) - Test Filter represented by a 3-bit floating-point mantissa Spatial Integrator Structure

Numerator Coefficients	
c_{00} = .610351563E-04	c_{01} = .366210938E-03
c_{10} = .366210938E-03	c_{11} = .244140625E-02
c_{20} = .732421875E-03	c_{21} = .488281250E-02
c_{30} = .488281250E-03	c_{31} = .292968750E-02
c_{02} = .732421875E-03	c_{03} = .488281250E-03
c_{12} = .488281250E-02	c_{13} = .292968750E-02
c_{22} = .976562500E-02	c_{23} = .585937500E-02
c_{32} = .585937500E-02	c_{33} = .390625000E-02
Denominator Coefficients	
d_{00} = .100000000E+01	d_{01} = .875000000E+00
d_{10} = .875000000E+00	d_{11} = .875000000E+00
d_{20} = .312500000E+00	d_{21} = .312500000E+00
d_{30} = .625000000E-01	d_{31} = .546875000E-01
d_{02} = .312500000E+00	d_{03} = .625000000E-01
d_{12} = .312500000E+00	d_{13} = .546875000E-01
d_{22} = .125000000E+00	d_{23} = .234375000E-01
d_{32} = .234375000E-01	d_{33} = .390625000E-02

Table 3.4 Quantized Coefficients for (3 x 3) - Test Filter represented by a 9-bit floating-point mantissa Direct Form Structure

Numerator Coefficients	
a_{00} = .703334808E-04	a_{01} = .211238861E-03
a_{10} = .211238861E-03	a_{11} = .633239746E-03
a_{20} = .211238861E-03	a_{21} = .633239746E-03
a_{30} = .703334808E-04	a_{31} = .211238861E-03
a_{02} = .211238861E-03	a_{03} = .703334808E-04
a_{12} = .633239746E-03	a_{13} = .211238861E-03
a_{22} = .633239746E-03	a_{23} = .211238861E-03
a_{32} = .211238861E-03	a_{33} = .703334808E-04
Denominator Coefficients	
b_{00} = .100000000E+01	b_{01} = -.209375000E+01
b_{10} = -.209375000E+01	b_{11} = .448437500E+01
b_{20} = .156640625E+01	b_{21} = -.342187500E+01
b_{30} = -.404296875E+00	b_{31} = .898437500E+00
b_{02} = .156640625E+01	b_{03} = -.404296875E+00
b_{12} = -.342187500E+01	b_{13} = .898437500E+00
b_{22} = .265625000E+01	b_{23} = -.707031250E+00
b_{32} = -.707031250E+00	b_{33} = .187988281E+00

Table 3.5 Error Function Comparison Fixed-point Arithmetic

Number of bits for Multiplier Coefficients	Direct Form Structure	Integrator Form Structure
32	.478516	.477524
24	.41716	.47670
20	.686798	.490082
18	1.92467	.503891
16	17.1559	.307266
14	29.1746	3.8201
12	120.9 unstable	4.81192
10		17.2638
9		16.9095
8		52.8 unstable

Table 3.6 Maximum and Minimum Accumulator values versus Multiplier coefficient word length fixed-point arithmetic

Direct Form Structure				
Number of bits for Multiplier Coefficients	D_1 Maximum Denominator	D_1 Minimum Denominator	D_2 Maximum Numerator	D_2 Minimum Numerator
32	2.56E11	-1.63E13	5.99E14	-8.76E10
24	4.29E09	-6.51E10	2.34E12	-4.29E09
20	4.29E09	-4.28E09	1.46E11	-4.29E09
18	4.50E06	-9.93E08	3.46E10	-1.24E09
16	7.37E05	-2.64E08	8.61E09	-5.99E08
14	1.19E07	-5.61E07	2.05E09	-3.75E08
Spatial Integrator Structure				
32	1.59E12	-1.03E12	3.74E13	-5.39E09
24	4.35E09	-4.29E09	1.46E11	-4.29E09
20	3.87E08	-2.53E08	8.66E09	-7.05E08
18	9.67E07	-6.31E07	2.15E09	-5.34E06
16	2.32E07	-1.58E07	5.71E08	-8.22E04
14	6.05E06	-3.94E06	1.43E08	-1.94E04
12	1.51E06	-9.82E05	3.56E07	-4403
10	3.75E05	-2.45E05	9.55E06	-988
9	1.82E05	-1.15E05	4.76E06	-1113

Table 3.7 Comparison of Internal Stored values for (3 x 3) Direct Form and Spatial Integrator 2D filters for Fixed-point Arithmetic

S_i - Number (Figure 3.11, 3.12)	Direct Form Structure		Integrator Form Structure	
	Minimum	Maximum	Minimum	Maximum
0	-137	64783	-497	502
1	-137	64783	-750	751
2	-137	64783	-1361	1375
3	-137	64783	-4486	3893
4	-137	64783	-750	751
5	-137	64783	-967	935
6	-137	64783	-1538	1533
7	-137	64783	-5205	5170
8	-137	64783	-1361	1375
9	-137	64783	-1538	1533
10	-137	64783	-2835	2768
11	-137	64783	-9955	9682
12	-137	64783	-4486	3893
13	-137	64783	-5204	5170
14	-137	64783	-9955	9682
15	-137	64783	-906	60884

Maximum Multiplier Word Length R_{len} is 16 bits

Table 3.8 Quantized Coefficients for (3 x 3) - Test Filter represented by 16-bit fixed-point arithmetic

a) Direct Form Structure

Numerator Coefficients

$A_{00} = 2364$	$A_{01} = 7093$	$A_{02} = 7093$	$A_{03} = 2364$
$A_{10} = 7093$	$A_{11} = 21279$	$A_{12} = 21279$	$A_{13} = 7093$
$A_{20} = 7093$	$A_{21} = 21279$	$A_{22} = 21279$	$A_{23} = 7093$
$A_{30} = 2364$	$A_{31} = 7093$	$A_{32} = 7093$	$A_{33} = 2364$

Numerator Divide Value
N = 33554432

Denominator Coefficients

$B_{00} = 1$	$B_{01} = -8602$	$B_{02} = 6431$	$B_{03} = -1659$
$B_{10} = -8602$	$B_{11} = 18415$	$B_{12} = -14044$	$B_{13} = 3684$
$B_{20} = 6431$	$B_{21} = -14044$	$B_{22} = 10908$	$B_{23} = -2897$
$B_{30} = -1659$	$B_{31} = 3684$	$B_{32} = -2897$	$B_{33} = 771$

Denominator Divide Value
M = 4096

b) Spatial Integrator Structure

Numerator Coefficients

$C_{00} = 148$	$C_{01} = 887$	$C_{02} = 1773$	$C_{03} = 1182$
$C_{10} = 887$	$C_{11} = 5320$	$C_{12} = 10640$	$C_{13} = 7093$
$C_{20} = 1773$	$C_{21} = 10640$	$C_{22} = 21279$	$C_{23} = 14186$
$C_{30} = 1182$	$C_{31} = 7093$	$C_{32} = 14186$	$C_{33} = 9457$

Numerator Divide Value
N = 2097152

Denominator Coefficients

$D_{00} = 1$	$D_{01} = 29489$	$D_{02} = 12123$	$D_{03} = 2133$
$D_{10} = 29489$	$D_{11} = 29344$	$D_{12} = 12217$	$D_{13} = 2029$
$D_{20} = 12123$	$D_{21} = 12217$	$D_{22} = 4976$	$D_{23} = 846$
$D_{30} = 2133$	$D_{31} = 2029$	$D_{32} = 846$	$D_{33} = 146$

Denominator Divide Value
M = 32768

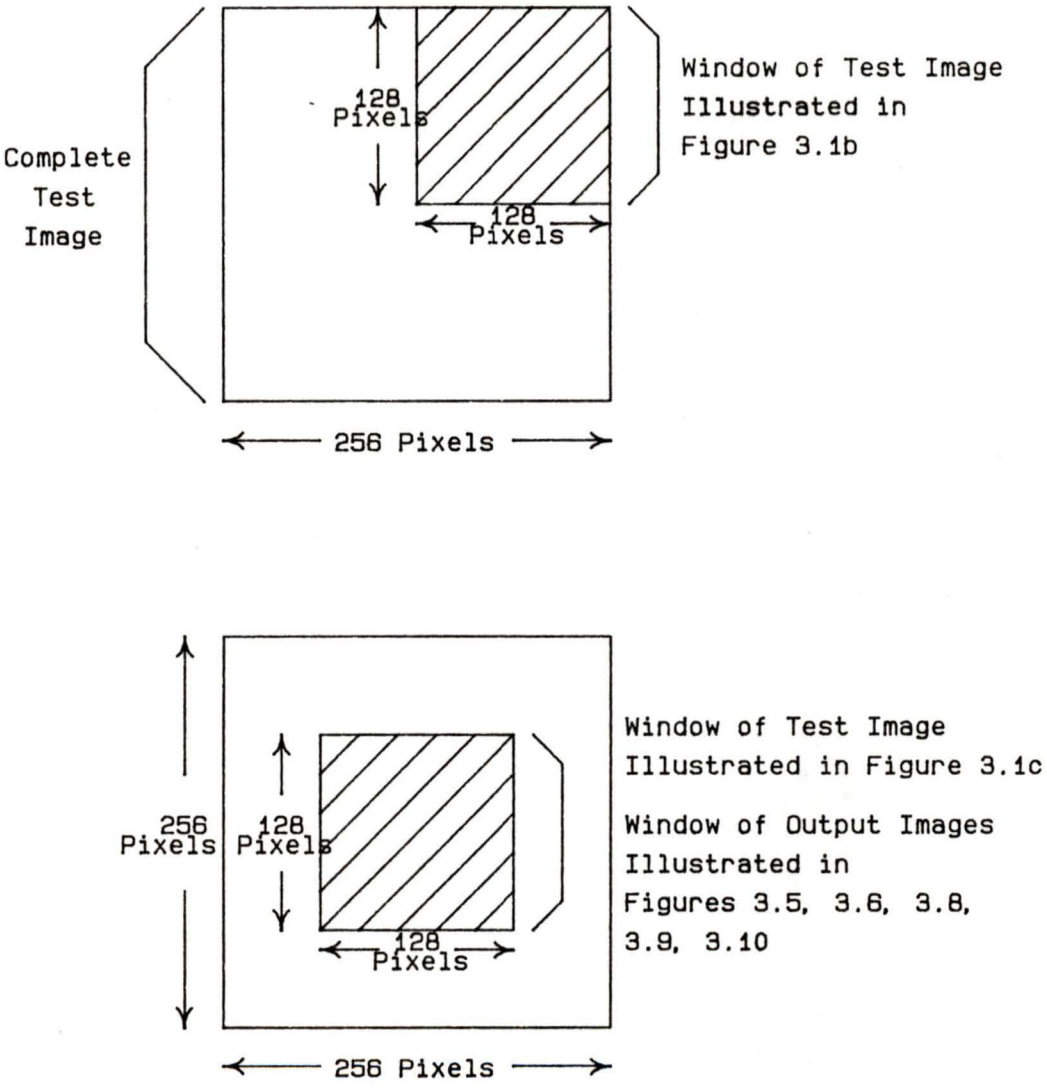


Figure 3.1a Outline of Test Image

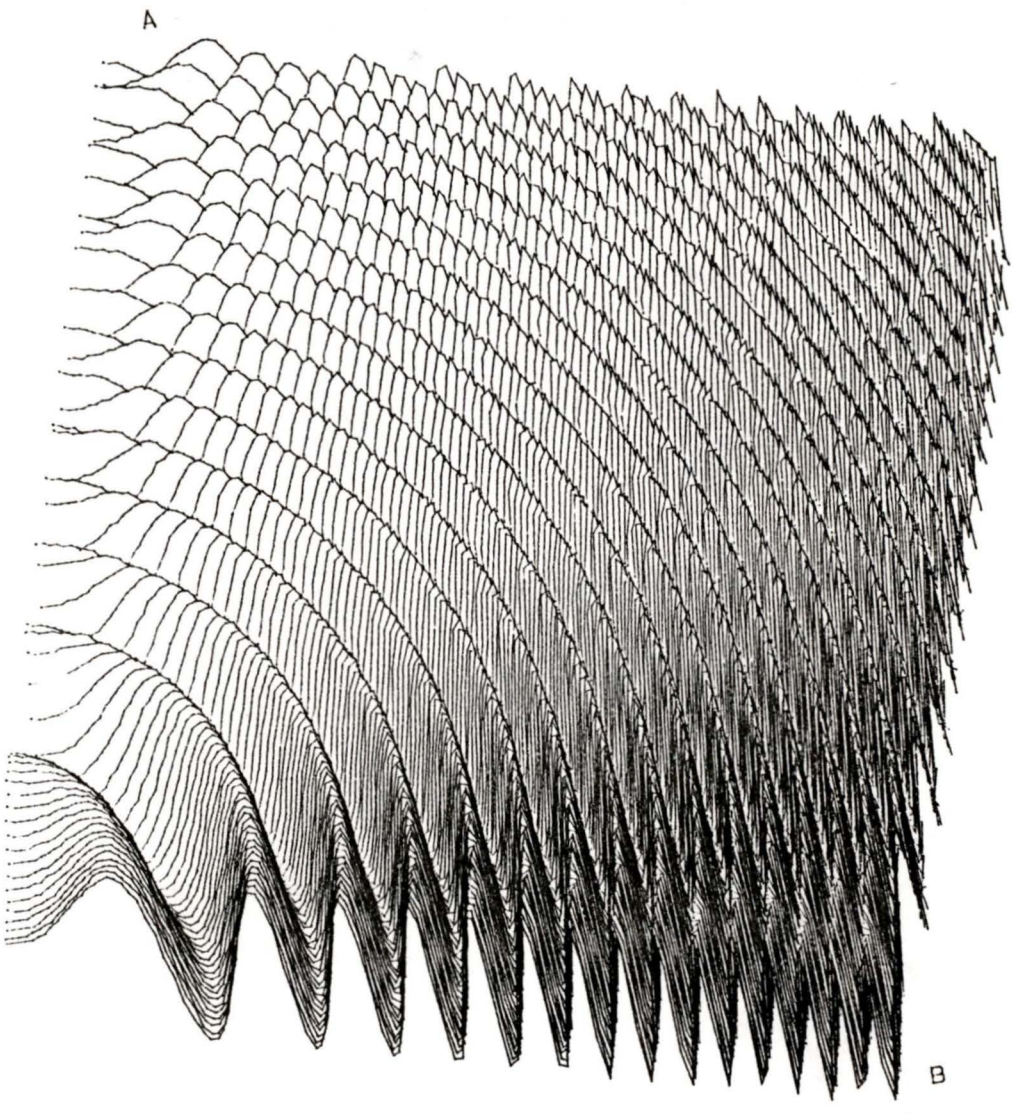


Figure 3.1b Upper Left Hand 1/4 of Input Test Image

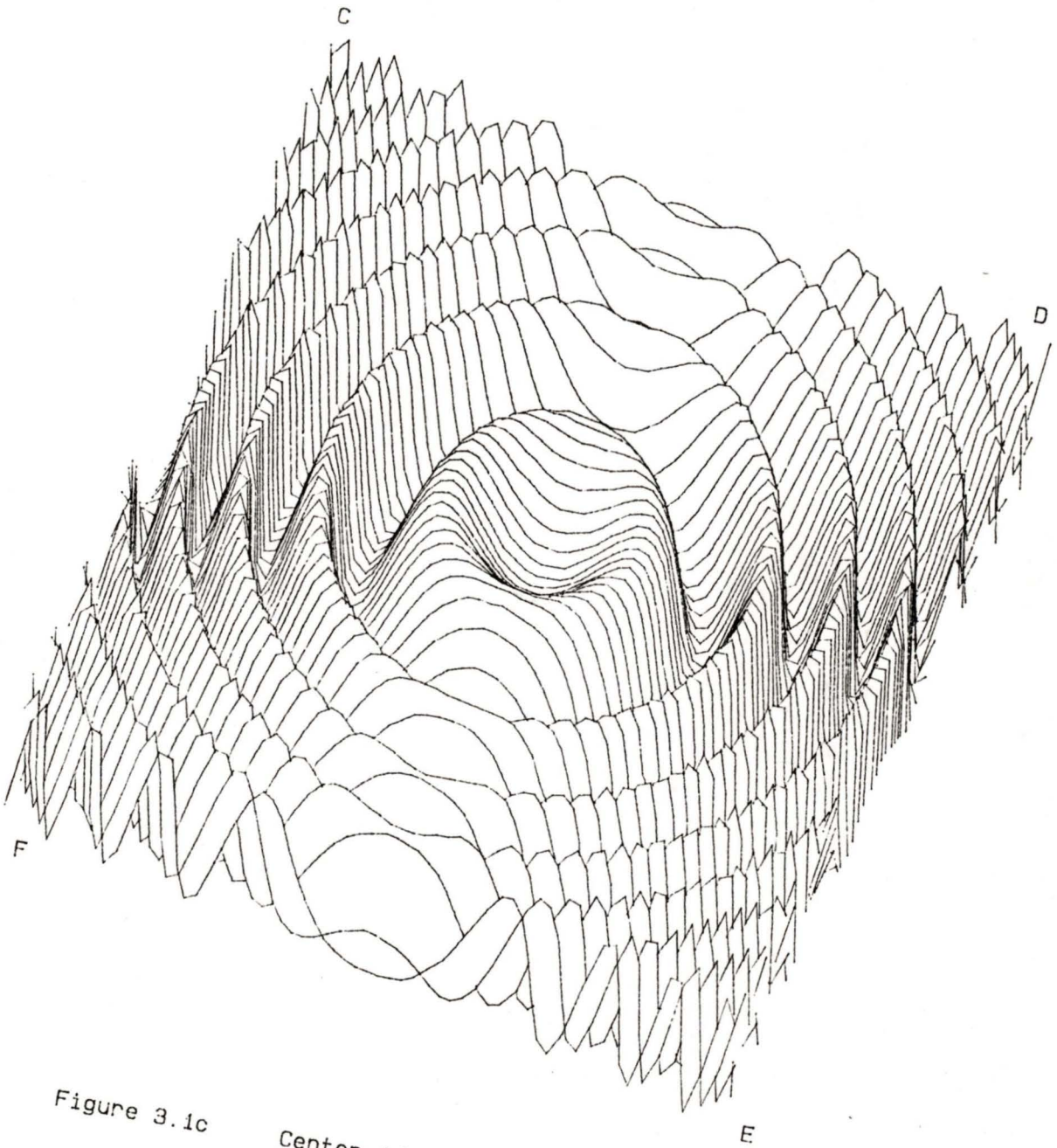


Figure 3.1c Center 1/4 of Input Test Image

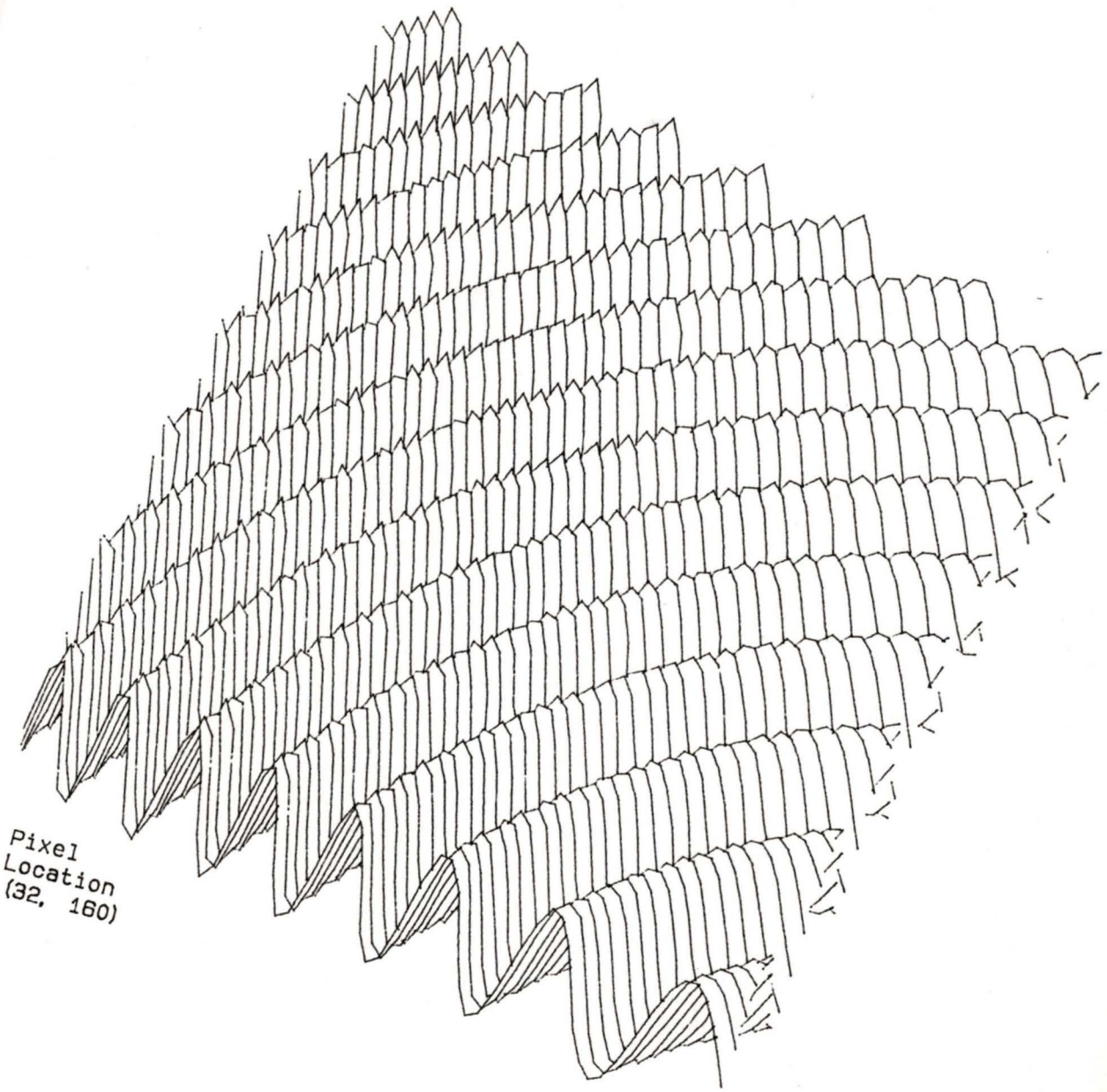


Figure 3.2a

64 x 64 Pixel Window of Test Image

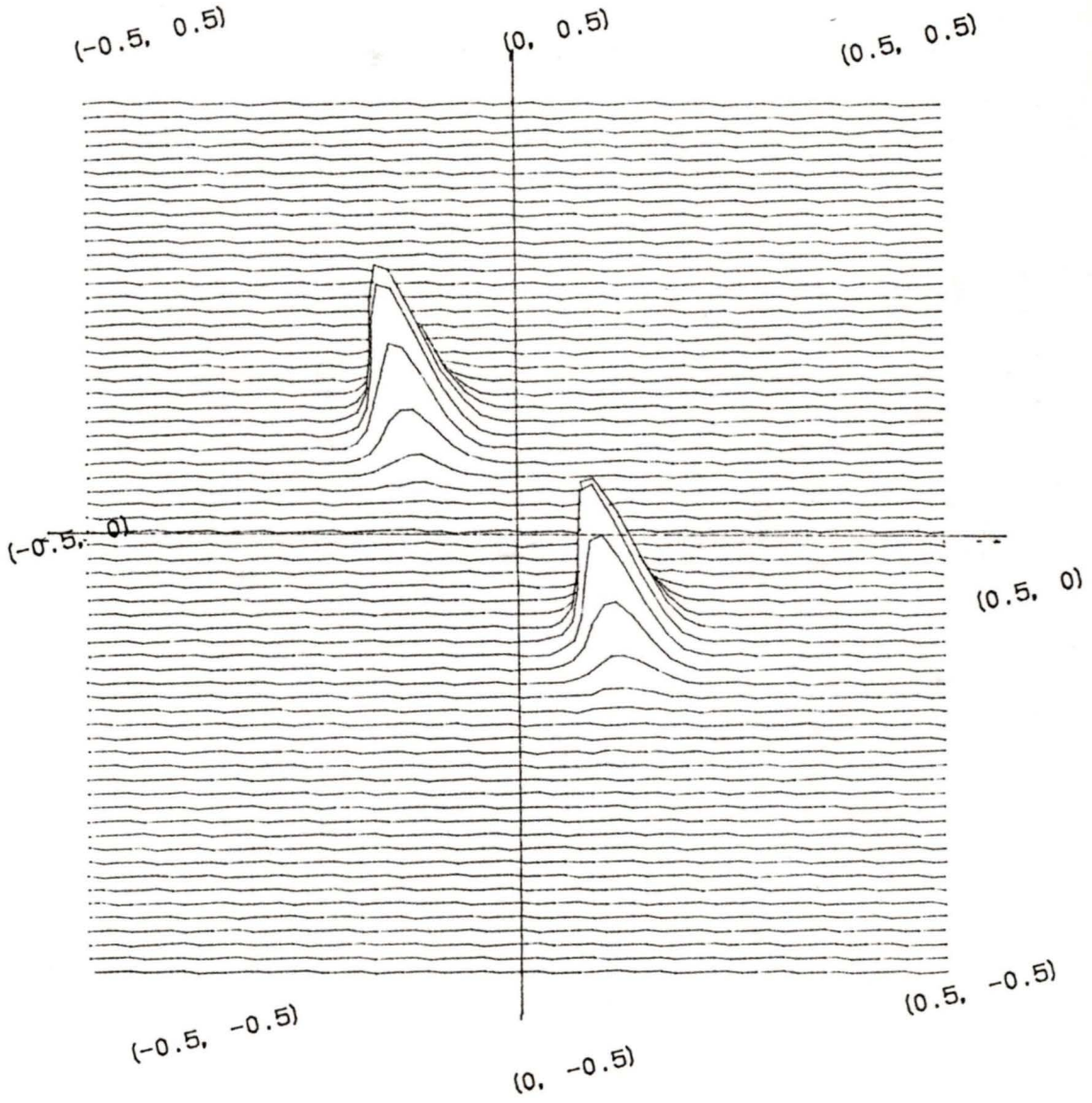
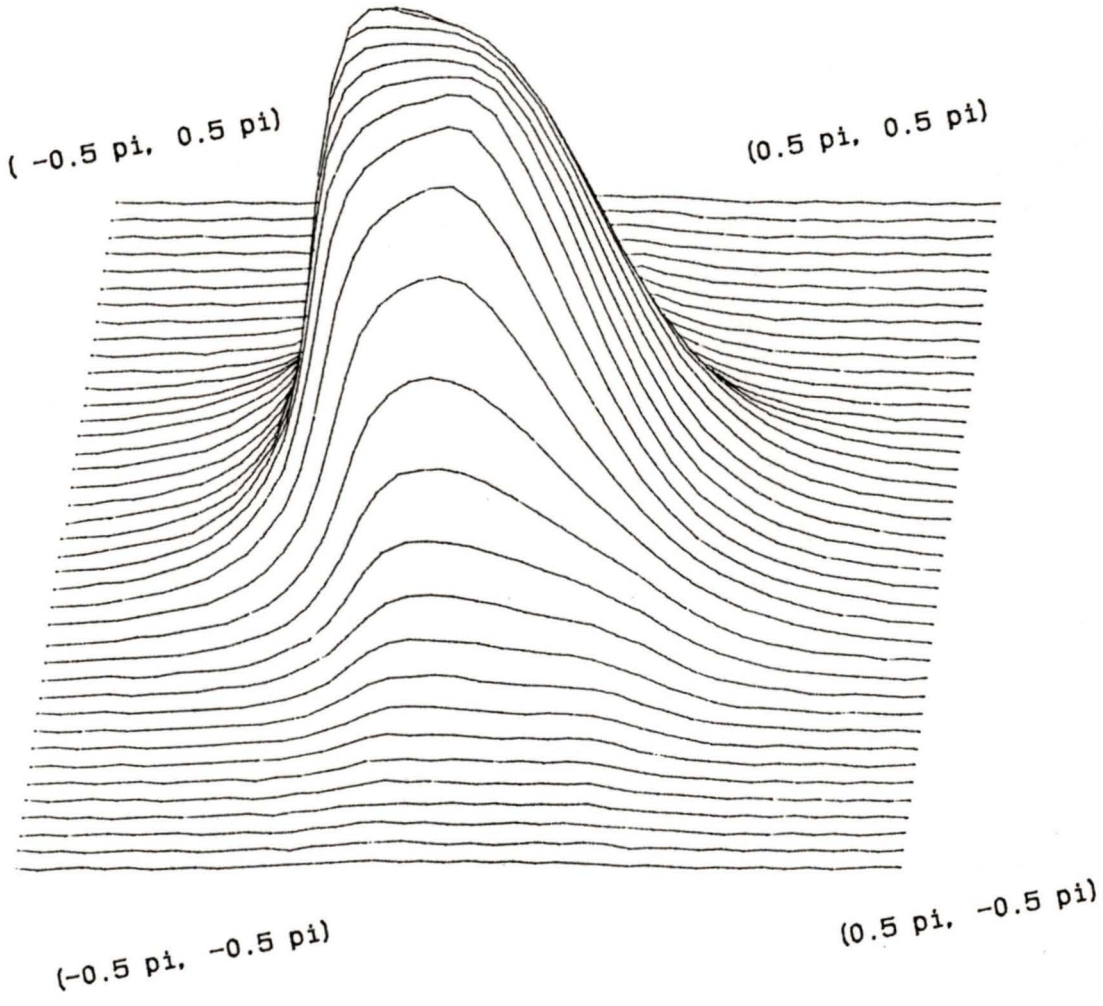


Figure 3.2b 2D FFT Magnitude Response of Figure 3.2a



Order: 3, 3
Cutoff: 0.15 pi
Gain: 1.0

Figure 3.3 2D Magnitude Response of Circularly Symmetric Lowpass Filter

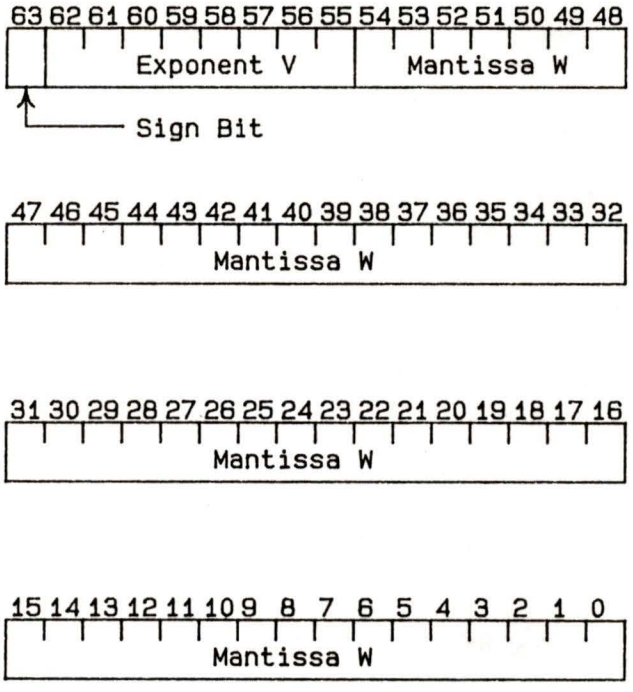
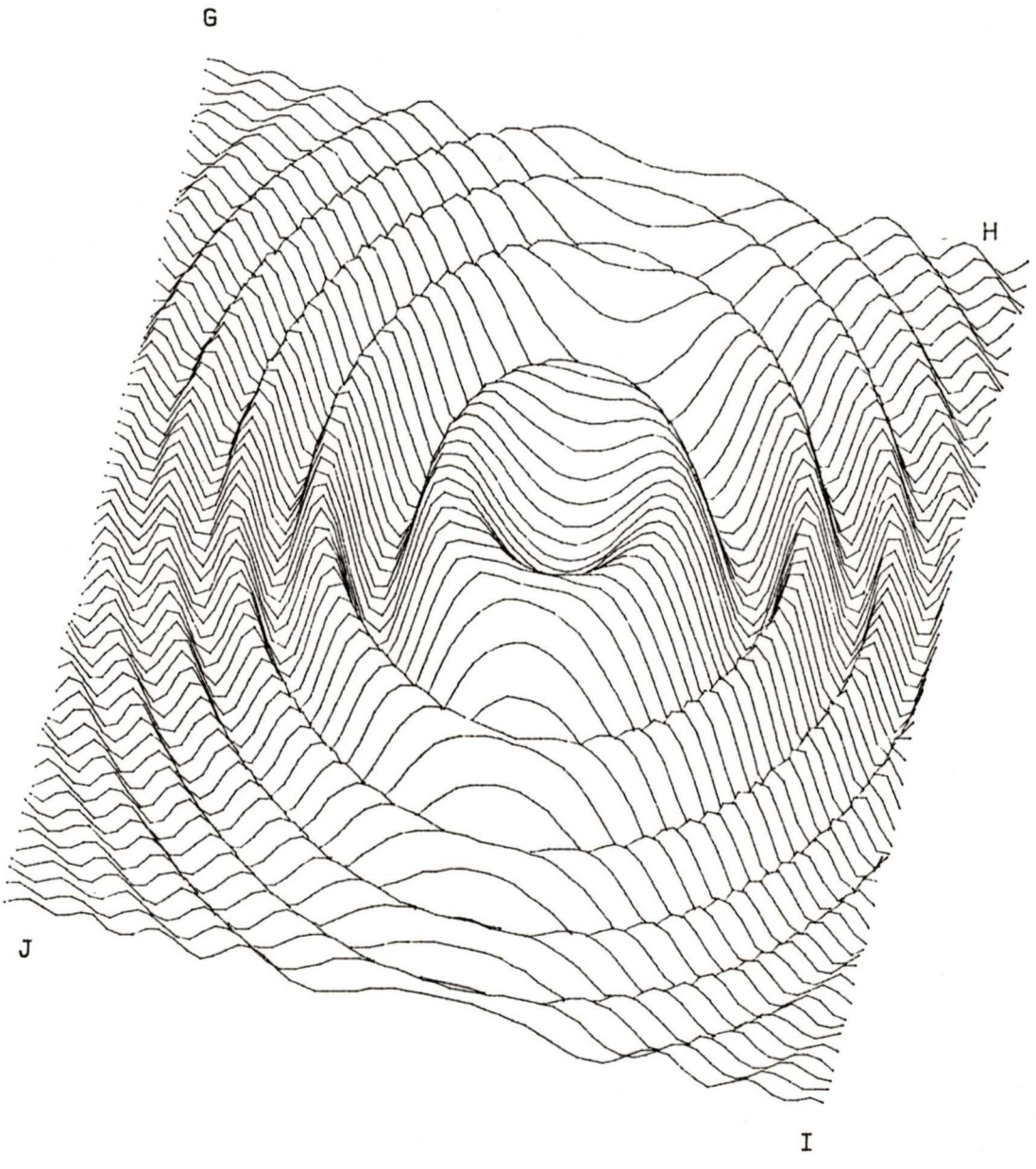
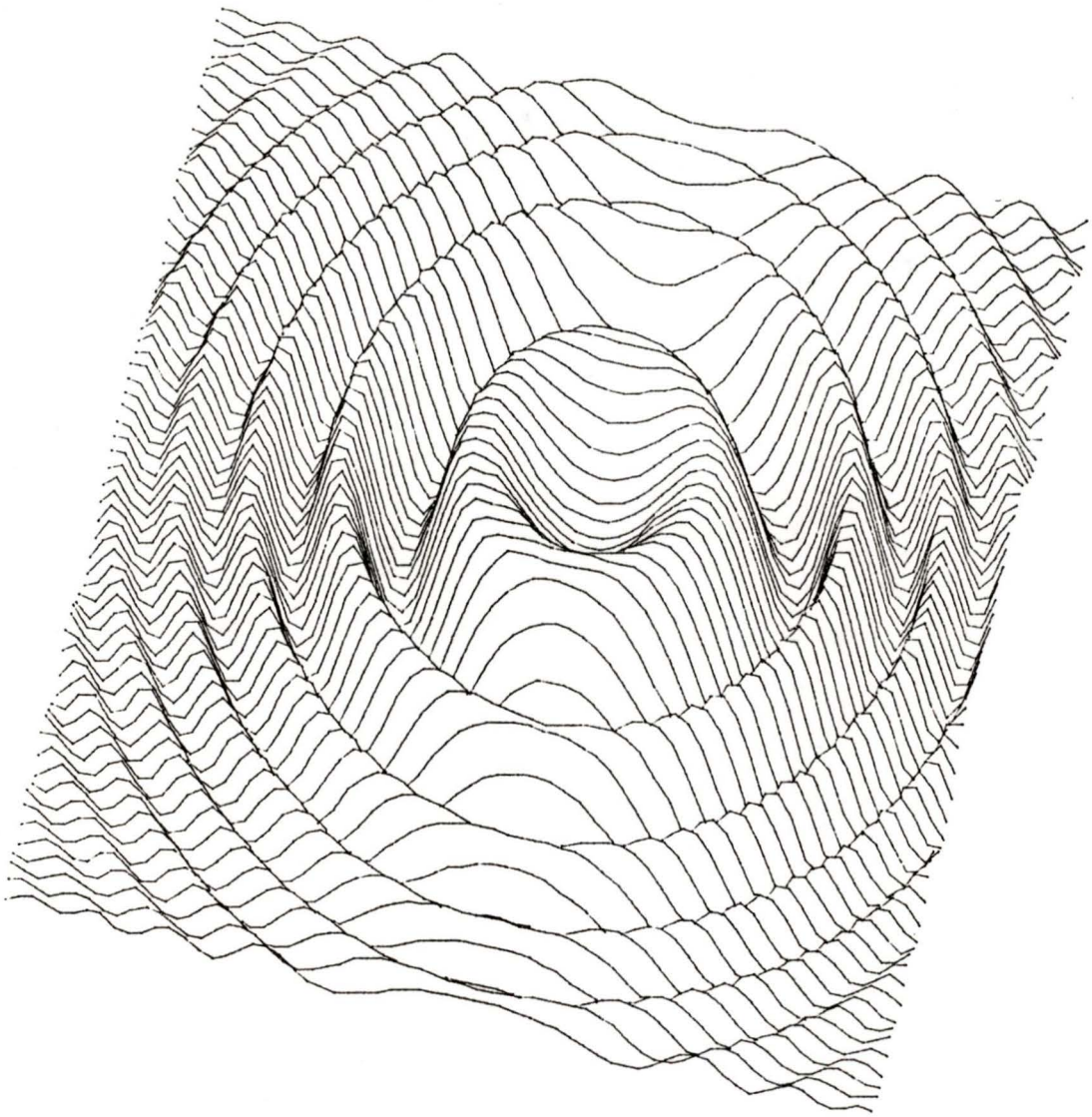


Figure 3.4 Floating Point Format for Charles River Data Systems Microcomputer



Calculated Using Double Precision Floating Point Arithmetic

Figure 3.5 Center 1/4 of the Output Image Y
of a 2D Direct Form Lowpass Filter



Calculated Using Double Precision Floating Point Arithmetic

Figure 3.6 Center 1/4 of the Output Image Y
of a 2D Spatial Integrator Lowpass Filter

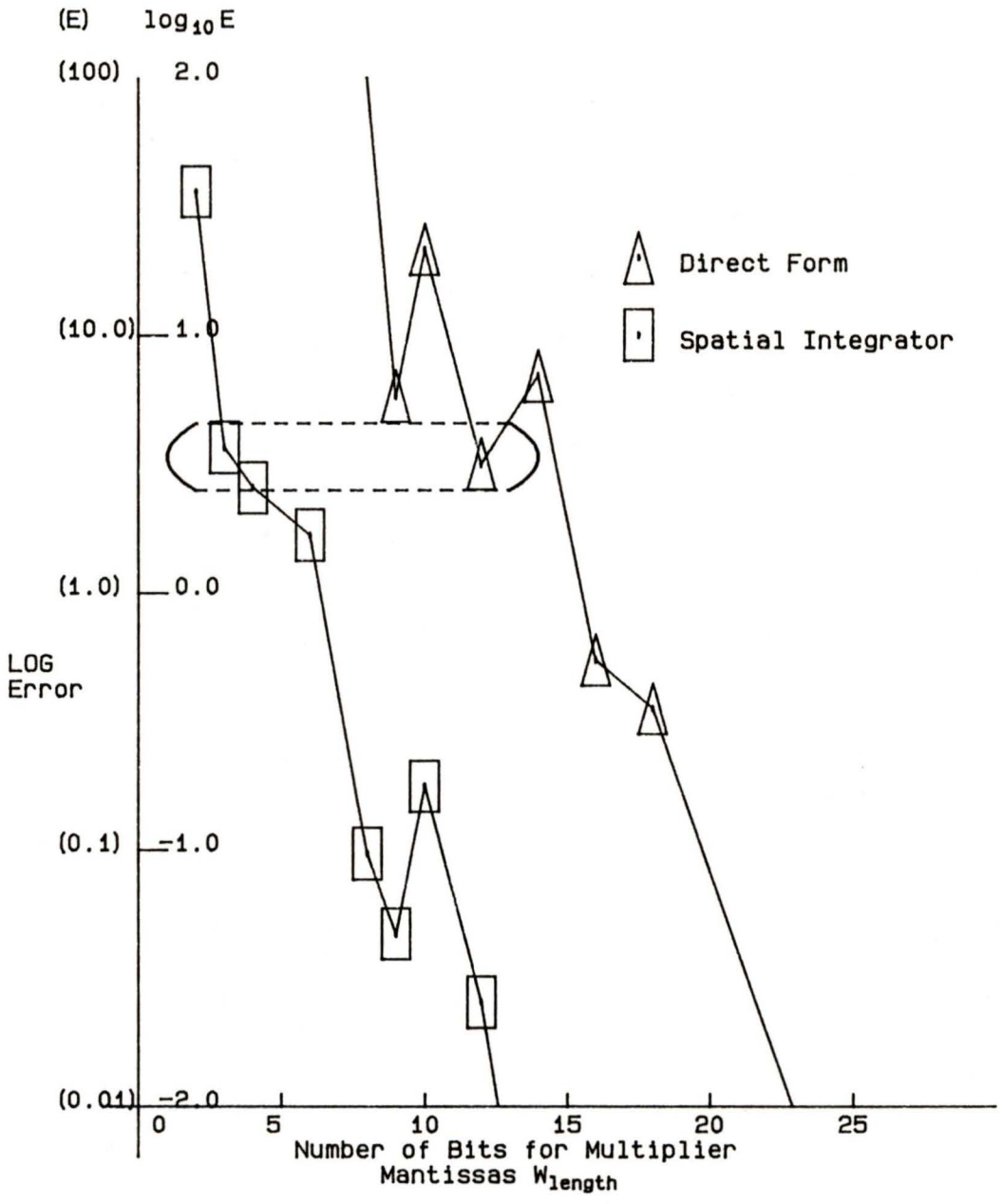


Figure 3.7 Error Comparison - Floating-point

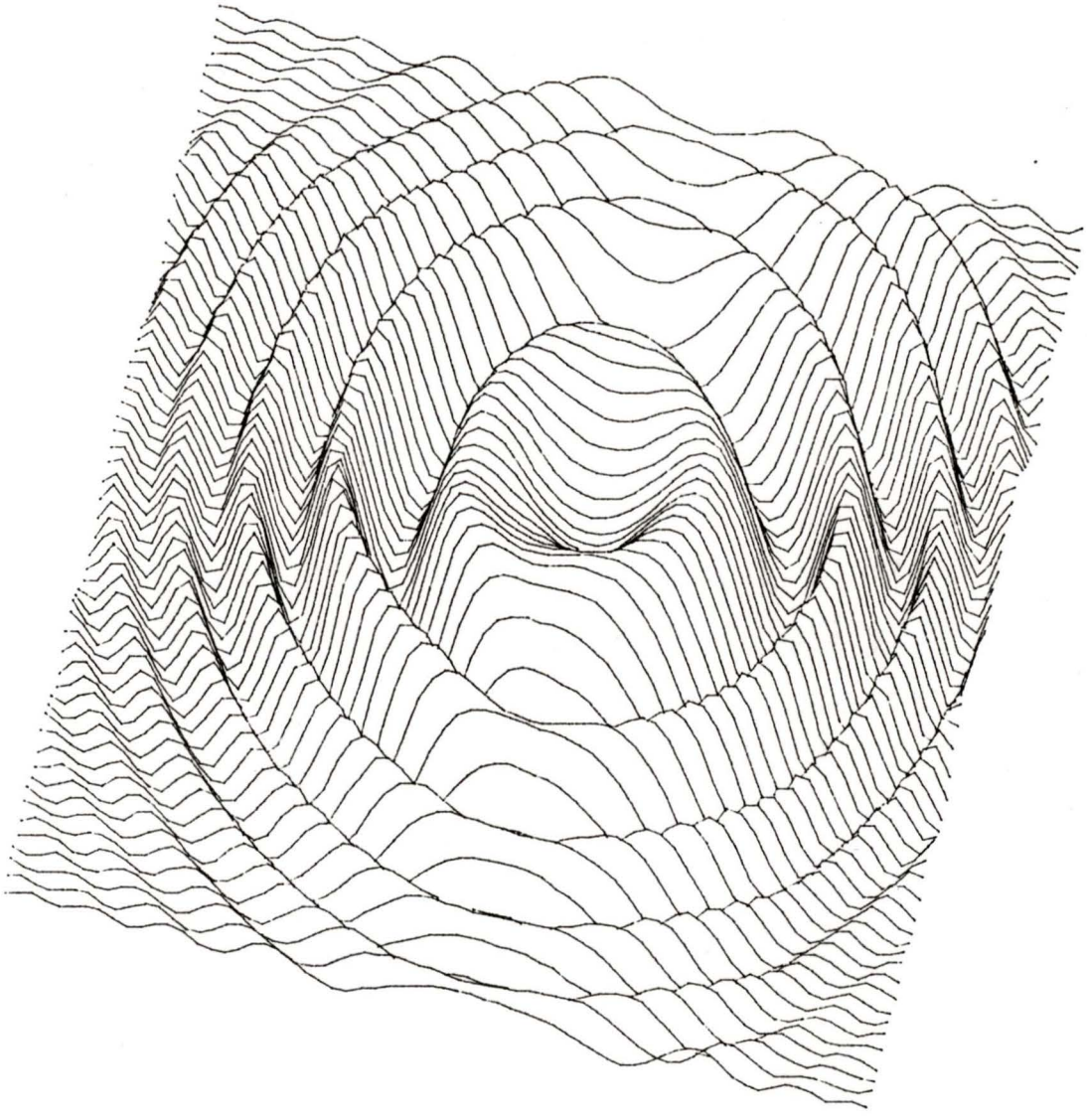


Figure 3.8 Center 1/4 of the Output Image Y of a Spatial Integrator Lowpass Filter Calculated Using Floating Point Arithmetic with 3 Bit Mantissa Multiplier Coefficients

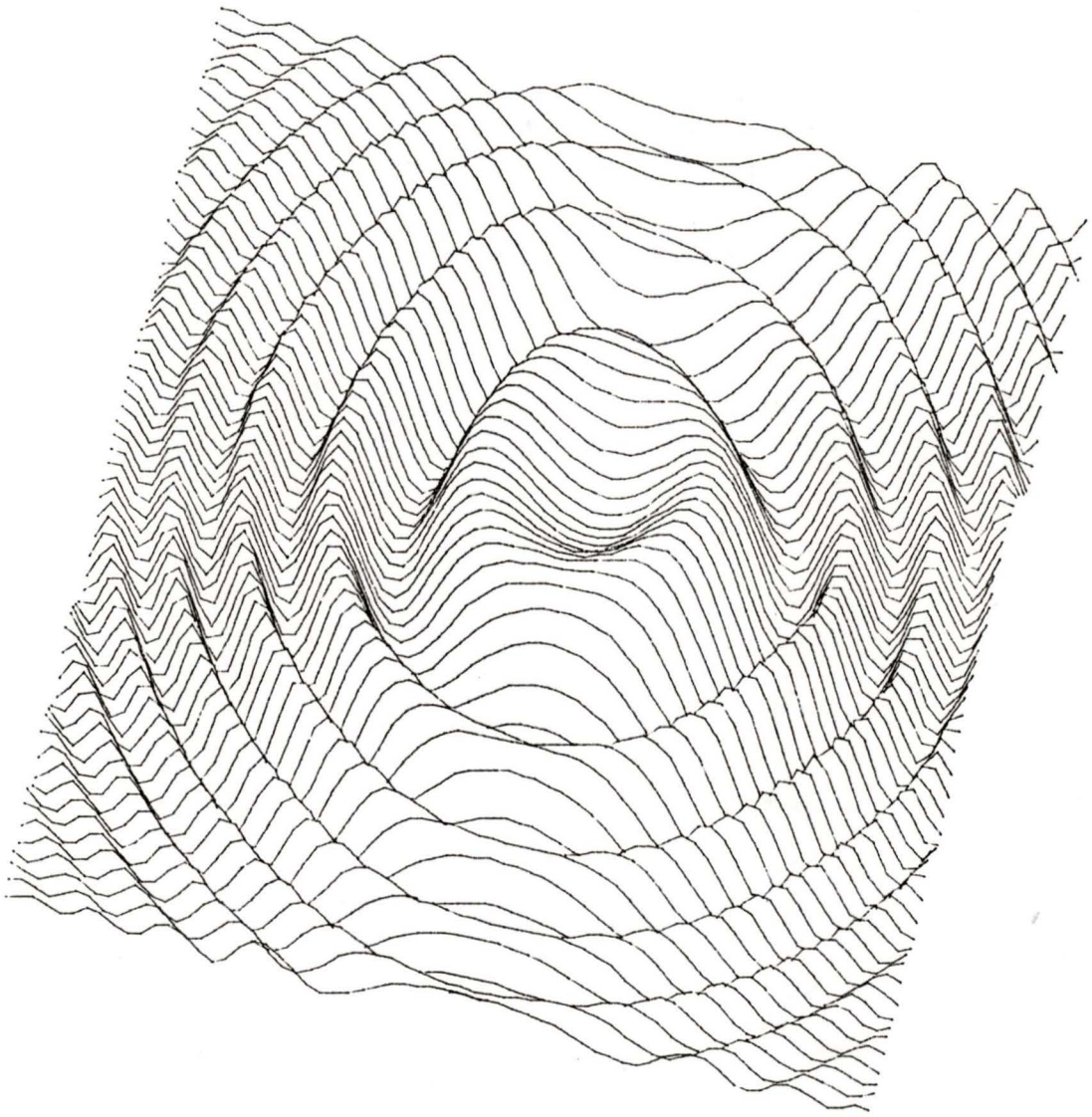


Figure 3.9 Center 1/4 of the Output Image Y of a 2D Direct Form Filter Calculated Using Floating Point Arithmetic with 9 Bit Mantissa Multiplier Coefficients

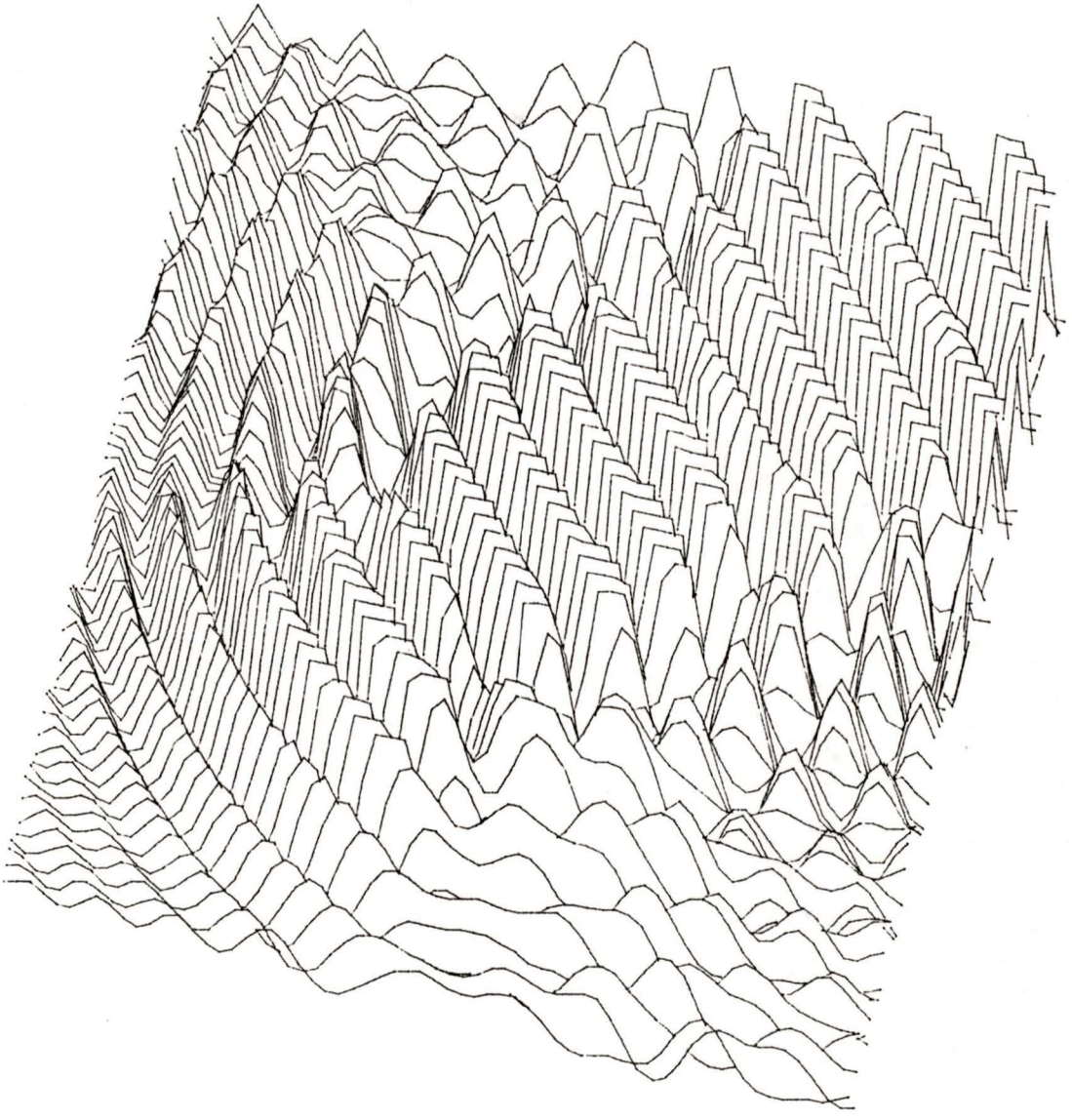


Figure 3.10 Center 1/4 of Output Image Y of a 2D Spatial Integrator Lowpass Filter Calculated Using Floating Point Arithmetic with 2 Bit Mantissa Multiplier Coefficients (Unstable Filter)

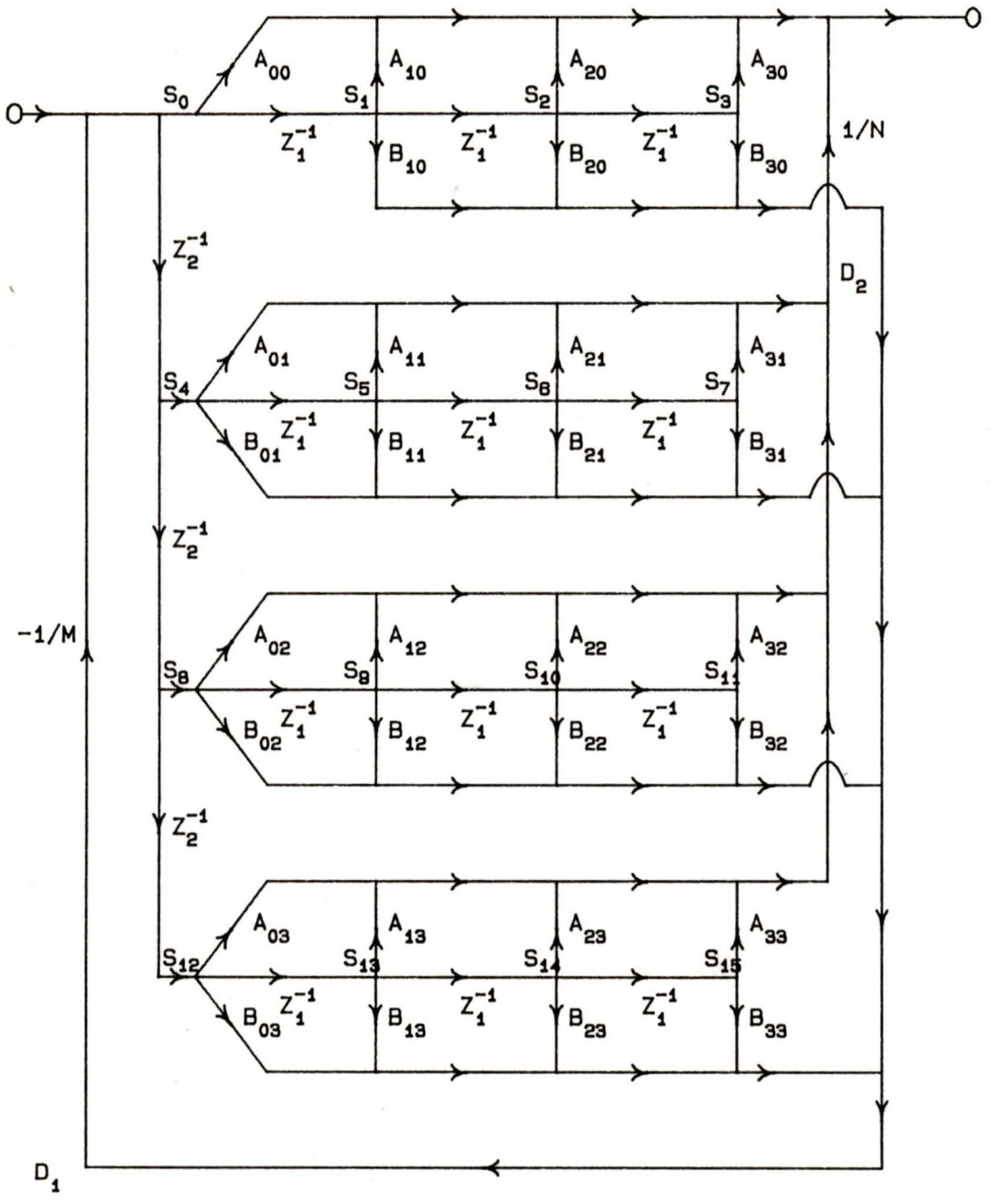


Figure 3.11 3 x 3 Order 2D Filter Direct Form Structure

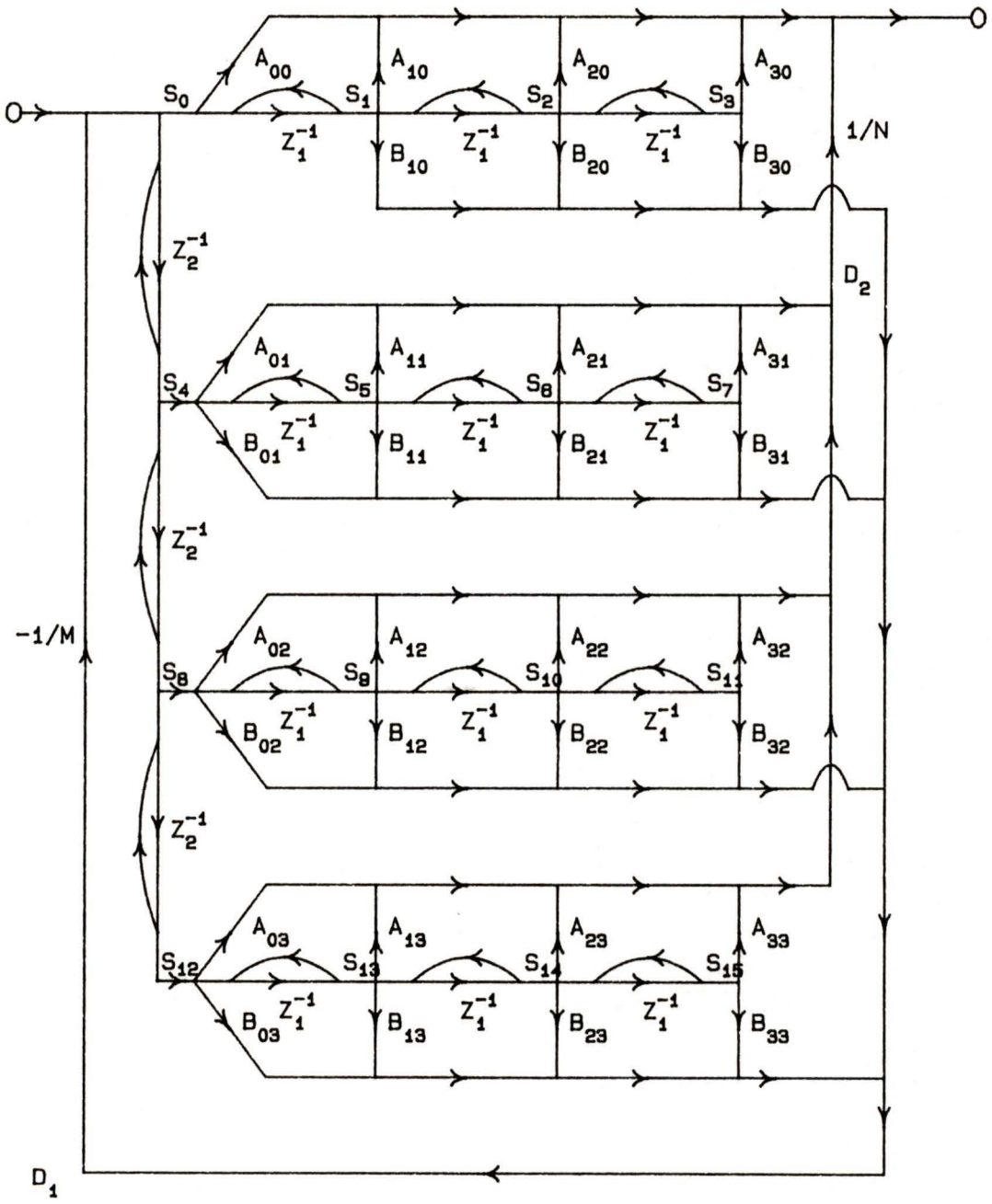


Figure 3.12 3 x 3 Order 2D Filter Spatial Integrator Structure

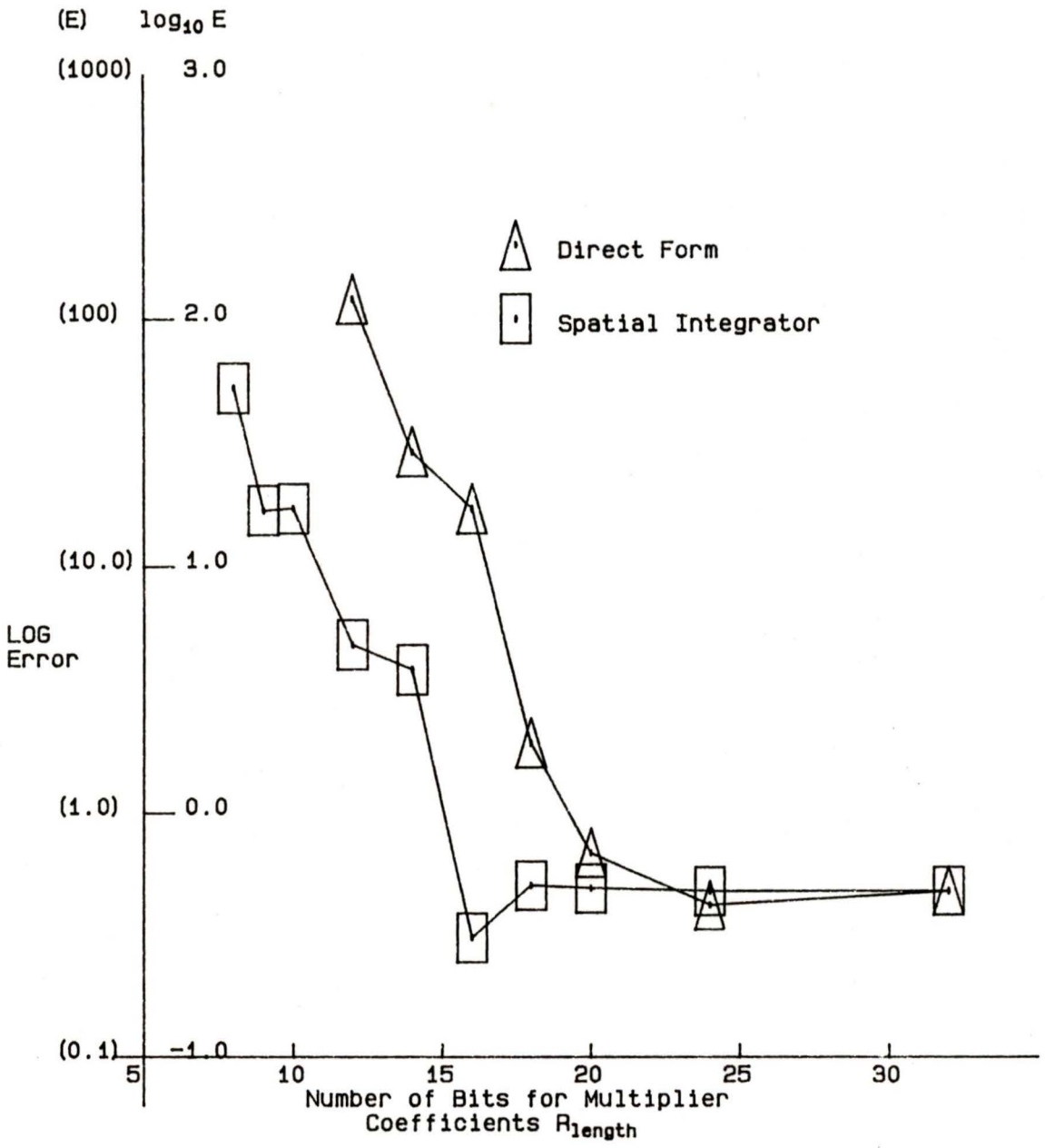


Figure 3.13 Error Comparison -- Fixed-point

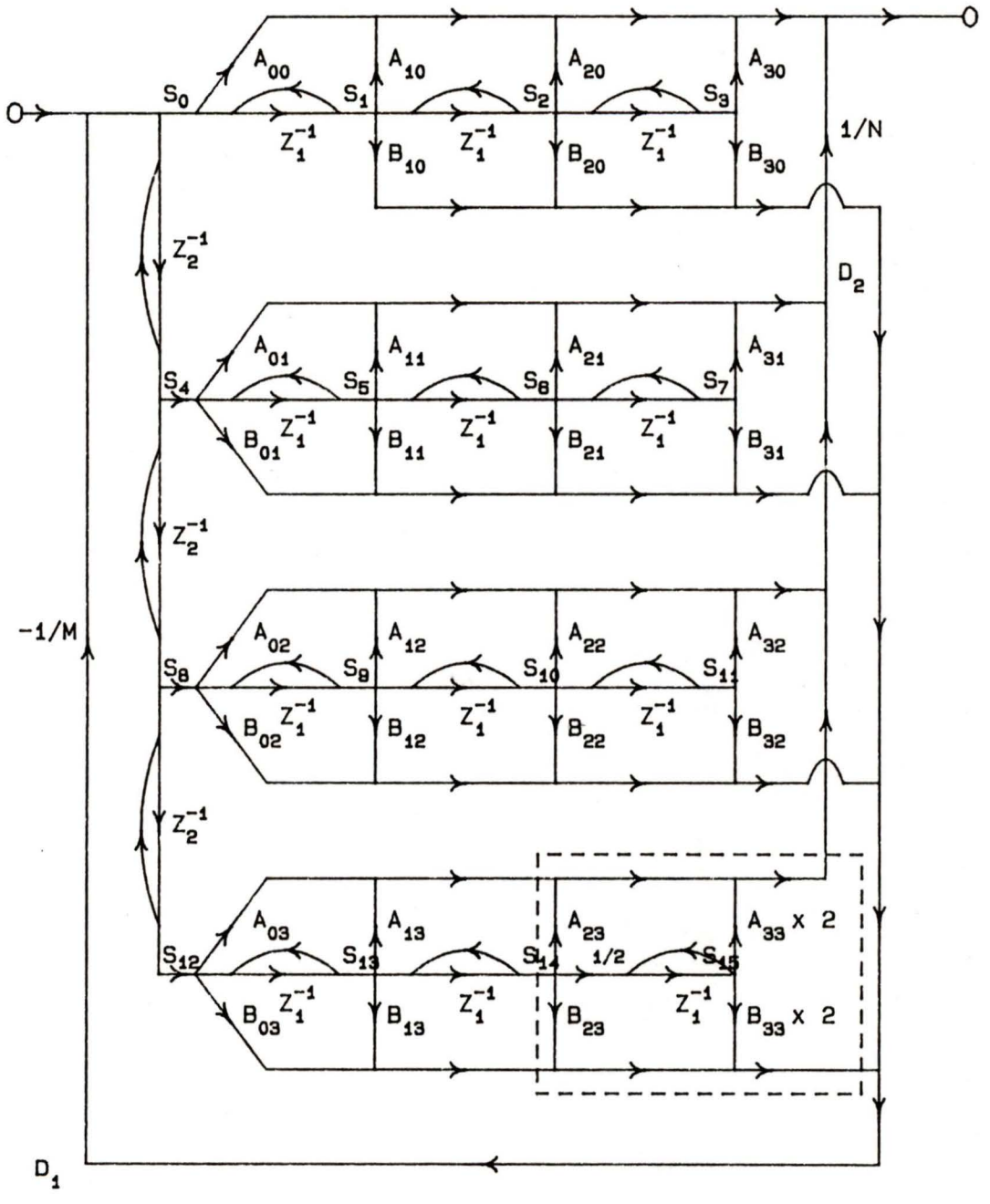


Figure 3.14 3 x 3 Order 2D Modified Spatial Integrator Structure

4. A VIDEO ENHANCEMENT INSTRUMENT USING 2D IIR FILTERS

4.1. OPERATING PRINCIPLES OF THE VIDEO ENHANCEMENT INSTRUMENT

The ideas and methods of the previous chapters have been used to construct a Video Enhancement Instrument (hereafter referred to as the Instrument) that is portable, relatively inexpensive and high speed. It offers a particularly attractive alternative to the use of a dedicated supermini computer or mainframe computer with an array processor.

A photograph of the Instrument is shown in Figure 4.1. The architecture of the Instrument shown in Figure 4.2 is designed to implement the 2D Spatial Integrator filtering algorithm described in chapter 2 and summarized in Figure 2.4. That is, the input image to the 2D Processor, X in Figure 4.2 is a stream of rows (input vectors U_n in Figure 2.4) and results in an output image Y as a similar stream of rows (output vectors V_n in Figure 2.4). The 2D Processor in Figure 4.2 is the hardware corresponding to the 2D Filter in Figure 2.4. The Controller, the Display Memory, the Frame Grabber, and the other input/output (I/O) circuitry in Figure 4.2 are for the purposes of acquiring and displaying images and for controlling the flow of image pixels

into and out of the 2D Processor. In this sense, the 2D Processor is the central part of the Instrument.

The Instrument shown in Figure 4.3 is connected to a standard B/W television camera for image acquisition and is capable of converting a composite video image to a (512 x 512) by 8-bit digital image in one video frame cycle (1/30 seconds). The input/output images are displayed on a standard Output TV Monitor connected to the Instrument. The Instrument operates by interacting with the user and providing a menu of selectable 2D filter algorithms on the screen of a Video Display Terminal (VDT) having a keyboard. A photograph of the Video Enhancement system is given in Figure 4.4.

The Instrument combines the following elements in a single portable unit:

(1) standard video input - composite video.

(2) digitization of a video image in 1 frame (1/30 seconds).

(3) control of the gain and offset of the input video signal for effectively increasing image contrast.

(4) standard composite video RGB output.

(5) Look-up Table for loading color information for output display.

(6) DMA access to the stored image for transferring data efficiently.

(7) parallel processing for increased throughput.

(8) large selection of (3 x 3) order recursive 2D filter algorithms for image enhancement.

(9) standard RS-232C interface for user access and for hardware testing.

4.2. THE VIDEO INPUT FRAME GRABBER

The hardware used to acquire the TV image is manufactured by Matrox Electronic Systems[9], Montreal, Quebec, and is known as a Frame Grabber. A photograph of the Frame Grabber is given in Figure 4.5.

A composite video signal from a camera or video recorder is received by the input port of the Frame Grabber. At this point, the horizontal and vertical synchronization pulses of the composite video signal are separated from the video information. The period of the synchronization pulses from the camera are used to obtain a reference frequency to keep the video display synchronized to the video input. The video portion of the composite video signal is channeled to an analogue to digital converter (A/D) where an 8-bit converter continuously digitizes the video information at 30 frames/second. The 8-bit output of the A/D is stored in a (512 x 512) by 8-bit Display Memory via the Matrox video bus shown in Figure 4.2. The data acquisition

circuitry also provides a mechanism for adjusting the gain and static offset of the input video signal. These adjustments are useful since they allow the video signal level to be adjusted so that digitization can occur on that portion of the video signal that contains the most relevant information. A video signal is illustrated in Figure 4.6a that has a high level of contrast; the static offset is set to zero and the gain is maximized so that the full analogue range is digitized. The contrast of the video signal of Figure 4.6b is limited; the gain and static offset are adjusted so that digitization occurs at the point that will maximize the contrast of the input video signal. The system allows for 16 different levels of gain and 16 different levels of offset. These adjustments are under software control via an 8-bit I/O port on the Multibus.

4.3. THE DISPLAY MEMORY

The hardware used to store the TV image is also made by Matrox Electronic Systems[10], Montreal, Quebec, and is resident on two separate boards, each of which contains four (512 x 512) bit planes. Thus, the two boards constitute the 8 (512 x 512) bit plane Display Memory of Figure 4.2.

The Display Memory can be accessed via the Multibus using one of two methods. The first method of access is through the use of programmed Row and Column Registers which are pointers to a specific pixel $x(m,n)$ in the Display Memory. One can read or write to this pixel via a third register, known as the Data Register.

The second method of access is by mapping a 1 Kbyte block of the Multibus address space (1024 pixels or two rows) into the Display Memory. The Row and Column Registers are loaded with the lowest Display Memory location of a 1 Kbyte sequential block; that is they identify two rows. When a pixel is accessed via the Multibus the Row and Column Registers are automatically incremented (for two rows) to point to the next pixel in the Display Memory. This second method of access is chosen for the Instrument. It allows for a 1 Kbyte block of Display Memory to be accessed sequentially by DMA before it is necessary to re-initialize the DMA Controller in Figure 4.2.

The Matrox Video Bus (Figure 4.2) transfers data from the input A/D converter to the Display Memory and from the Display Memory to the Look-up Table for display on a Output TV Monitor. The on board logic arbitrates these requests so that they are transparent to the Multibus user.

4.4. LOOK-UP TABLE

The Matrox Frame Grabber board also contains a Look-up Table to establish a range of colors or intensities for the display of RGB composite TV signals (Figure 4.7). The Look-up Table contains 3 independently accessible 8-bit wide columns each 256 words long. The screen on the Output Monitor is refreshed each frame by sending pixels from the Display Memory to the Look-up Table. The 8-bit values are used for addresses to the 24 bits

(8 red, 8 green, 8 blue) of data information at each address location in the Look-up Table which are used to drive three 8-bit Digital to Analogue (D/A) converters. The three D/A converters generate an analogue video signal for the RGB monitor. Each address location in the Look-up Table may be loaded with any 24-bit combination. Thus, each address location may consist of 1 of a possible 16 million colors (2^{24}).

4.5. 2D PROCESSOR HARDWARE

The 2D Processor in Figure 4.2 has been constructed on a *single* circuit board and is designed to specifically meet the high speed processing requirements for filtering 2D images. It contains 4 Texas Instruments TMS32010 digital signal processors each with (1K x 16) bits of Program RAM, Input and Output FIFO buffers (first in first out) and a Transfer FIFO buffer. The Input/Output FIFO buffers allow for the efficient transfer of data between each TMS32010 and the Display Memory and the Transfer FIFO allows for the efficient transfer of data between adjacent TMS32010 processors. A photograph of the 2D Processor is given in Figure 4.10.

4.5.1. TMS32010 Microprocessor

The TMS32010 microprocessor was chosen for the 2D Processor because of its high speed and an architecture that is particularly suited to digital signal processing [11,12]. The TMS32010 combines the following features in a single chip:

(1) 16-bit data bus

(2) (144 x 16) bit on chip RAM for data

(3) double precision 32-bit ALU and accumulator

(4) 200 ns (16 x 16) bit multiplier

(5) barrel shifter for shifting data memory words into the ALU

(6) on chip oscillator

The TMS32010 uses a modified Harvard architecture in which the program memory and the data memory are on separate buses. This permits simultaneous execution and fetching of program instructions. For most instructions, this is a rate of 5 million instructions per second at the maximum clock frequency.

4.5.2. TMS32010 Program RAM

The Program RAM, 1K x 16-bit words, for each of the four TMS32010s is shown in Figure 4.2 in the 2D Processor. The Program RAM is down loaded from the 16 Kbyte EPROM with the 2D filtering algorithm and the multiplier coefficients for nine filters through the DMA interface.

4.5.3. Input/Output FIFO Buffers

The Input FIFO (Figure 4.8) is a 512 x 8-bit buffer which is loaded with a row of input image pixels (U_n in Figure 2.4) from the Display Memory.

The row U_n is transferred by DMA via the Multibus using the DMA Controller in Figure 4.2.

The Output FIFO (Figure 4.8) is a 512 x 8-bit buffer which contains a row of output pixels (V_n in Figure 2.4) that have been computed using the 2D filtering algorithm. An output row V_n is transferred by the DMA Controller (Figure 4.2) from the Output FIFO to the Display Memory.

4.5.4. Transfer FIFO

The Transfer FIFO (Figure 4.8) is a 16-bit wide buffer that allows for one way asynchronous data transfers from one TMS32010 microprocessor to the next TMS32010 microprocessor. The organization of the Transfer FIFOs for each TMS32010 microprocessor is shown in Figure 4.9. The number 1 TMS32010 can send data to the number 2 TMS32010 processor but cannot send data to the number 0 or number 3 TMS32010 processor.

4.5.5. Operation of 2D Processor

The 2D Processor implements the (3 x 3) order SFG of the Spatial Integrator structure shown in Figure 3.14. A flow chart showing the steps involved in transforming a row of input pixels U_n into a row of output pixels V_n is shown in Figure 4.11.

The Program RAM of the TMS32010 contains the multiplier coefficients for 9 separate filters. On initialization, the microprocessor controller (Figure

4.2) sends a single byte of value 0-8 to each of the four Input FIFOs via the Multibus. The TMS32010 uses this byte to select multiplier coefficients. This completes step (c) of Figure 4.11.

All the internal signals of the SFG (Figure 3.14) are stored in the TMS32010 on chip RAM. At the start of each row U_n, internal signals $S_1 - S_3$, $S_5 - S_7$, $S_9 - S_{11}$, and $S_{12} - S_{15}$ (Figure 3.14) are set to zero (step (e) Figure 4.11). The values of internal signals S_4, S_8, S_{12} (Figure 3.14) which are calculated from the computation of output row V_{n-1} are read from the Transfer FIFO (Figure 4.8). The input pixel $x(m,n)$ is read from the Input FIFO (step (g) Figure 4.11). This completes the gathering of the necessary information for the calculation of the output pixel $y(m,n)$ (step (h) Figure 4.11). The output pixel $y(m,n)$ is stored into the output FIFO (Figure 4.7) until the DMA Controller can transfer a row of pixels V_n to the Display Memory.

The next step (j) (Figure 4.11) is to calculate new values for internal signals $S_1 - S_3$, $S_5 - S_7$, $S_9 - S_{11}$, and $S_{12} - S_{15}$ (Figure 3.14) by row integrations (Equation 3.4) for the processing of output pixel $y(m+1,n)$. New values for S_4, S_8 and S_{12} are calculated by column integrations (Equation 3.4) and the results are stored to the Transfer FIFO for the calculation of $y(m,n+1)$ in the next row V_{n+1}. The row index is decremented (step m) and the program branches to step (d) to calculate the next output pixel

$y\,(m+1,n\,)$.

4.5.6. Efficient Buffering Between TMS32010 Microprocessors

The TMS32010 to TMS32010 Transfer FIFOs for all four processors are shown in Figure 4.8. FIFO Transfer buffer number 3 is 1536 words. This larger buffer is necessary to accommodate all the column integrations (signals S_4, S_8, S_{12}, Figure 3.13 (512 x 3)) necessary for the calculation of a complete row of output pixels $(512)V_n$ for the (3 x 3) order filters implemented. The buffering between TMS32010 processors allows each processor to work independently. For example, consider when the number 2 TMS32010 has just completed processing a row V_n and its Output FIFO is full (Figure 4.7). While the number 2 TMS32010 is waiting for DMA service, the number 1 TMS32010 may still calculate output pixels placing them into its Output FIFO. The number 1 TMS32010 will place the column integrations necessary for number 2 TMS32010 into FIFO Transfer buffer number 1 (Figure 4.9). When the Input FIFO for TMS32010 number 2 receives data number 2 TMS32010 will use the values from the number 1 Transfer FIFO while the number 1 TMS32010 is waiting for DMA service. The FIFO buffers contain Full and Empty flags which are polled before each transfer to prevent the loss of data.

4.6. SYSTEM CONTROLLER

The system Controller (Figure 4.2) contains a general purpose 8-bit microprocessor with a serial interface between the Instrument and a VDT terminal. The Controller also contains a programmable DMA Controller for transferring data from the Display Memory to the 2D Processor. The Controller contains 3 - 16-bit parallel ports that are used for down loading the TMS32010 Program RAM.

The Controller's microprocessor is a MC68B09 with 8 Kbytes of scratch pad RAM and 16 Kbytes of EPROM. There are two asynchronous serial interfaces, one for a terminal and the other to connect to a device like a computer for remote testing. The EPROM space contains all the DMA, Display Memory, Look-up Table, terminal interface, TMS32010 program and down loading routines. Also in the EPROM is a monitor for debugging 6809 programs. The 6809 controls the initialization of the Matrox boards and the initialization of the TMS32010 Program RAM.

4.7. SOFTWARE CONSIDERATIONS

The Instrument is used to implement (3 x 3) order 2D IIR filters using the Spatial Integrator structure of Figure 3.13. A TMS32010 assembly language program corresponding to the algorithm for this SFG contains approximately 250 instructions per pixel and requires 61.4 micro-seconds per pixel at a clock rate of 5 MHz. The number of 6809 instructions required to

initialize the DMA Controller to transfer each row U_n of the image to the TMS32010 2D Processor is 54 instructions or 1.36 micro-seconds per row for a 6809 clock rate of 2 MHz. Using four TMS32010 processors with an image size of (512 x 512) pixels, the processing time T for one complete image is therefore approximately

$$T = \frac{512 \times 512 \times 61.4 \cdot 10^{-6}}{4} + 512 \times 1.36 \cdot 10^{-6} = 4.025 \; seconds. \quad (4.1)$$

4.8. SUMMARY

A description of the architecture (Figure 4.2) of the Video Enhancement Instrument used for implementing 2D IIR filters is presented in this chapter. The Instrument consists of five Multibus cards. The (512 x 512) by 8-bit Display Memory is contained on two of the cards. The Instrument continuously digitizes a composite video input image for storage in the Display Memory. The output circuitry converts the digital information from the Display Memory into an RGB composite video output for display on a TV Monitor. The 2D Processor contains four high performance 16/32 bit microprocessors (TMS32010) which are used to implement the IIR Spatial Integrator filter algorithm. The system Controller contains an 8-bit microprocessor, a DMA Controller and an interface between the Instrument and a terminal, which are used to control the transfer of data between the Display Memory and the 2D Processor.

Extensive use of buffering, DMA and the multi-processor Spatial Integrator IIR filter algorithm allows the four high speed microprocessors to operate concurrently. The Instrument can process a (512 x 512) by 8-bit image using a (3 x 3) order filter in about *4 seconds*. The same structure implemented in 'C' on a Digital Equipment Corporation, Vax 11/780 requires approximately 8.5 minutes to execute.

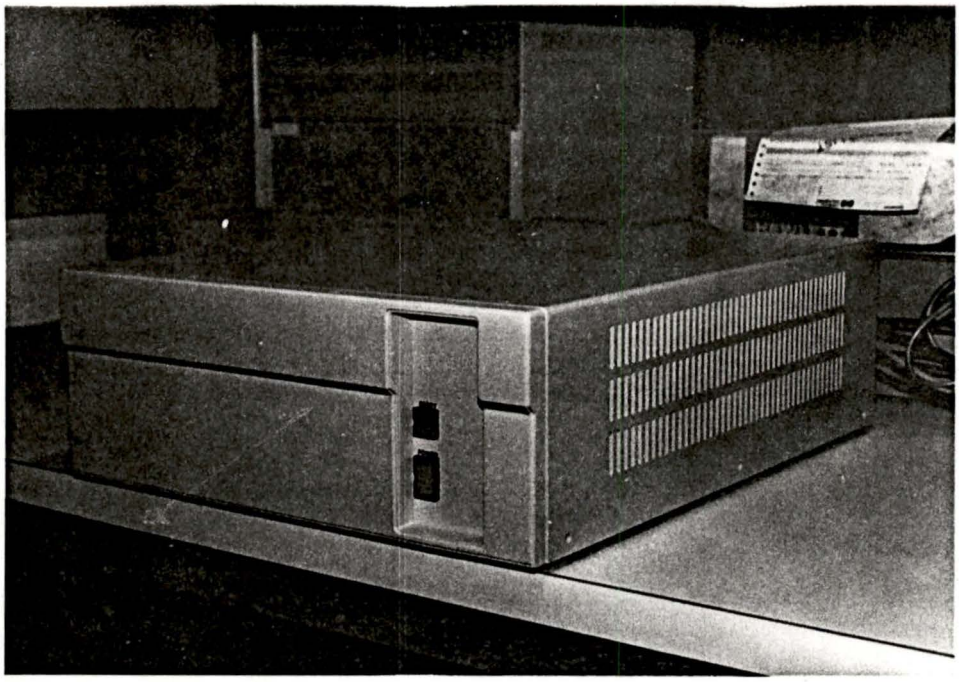


Figure 4.1 The Video Enhancement Instrument

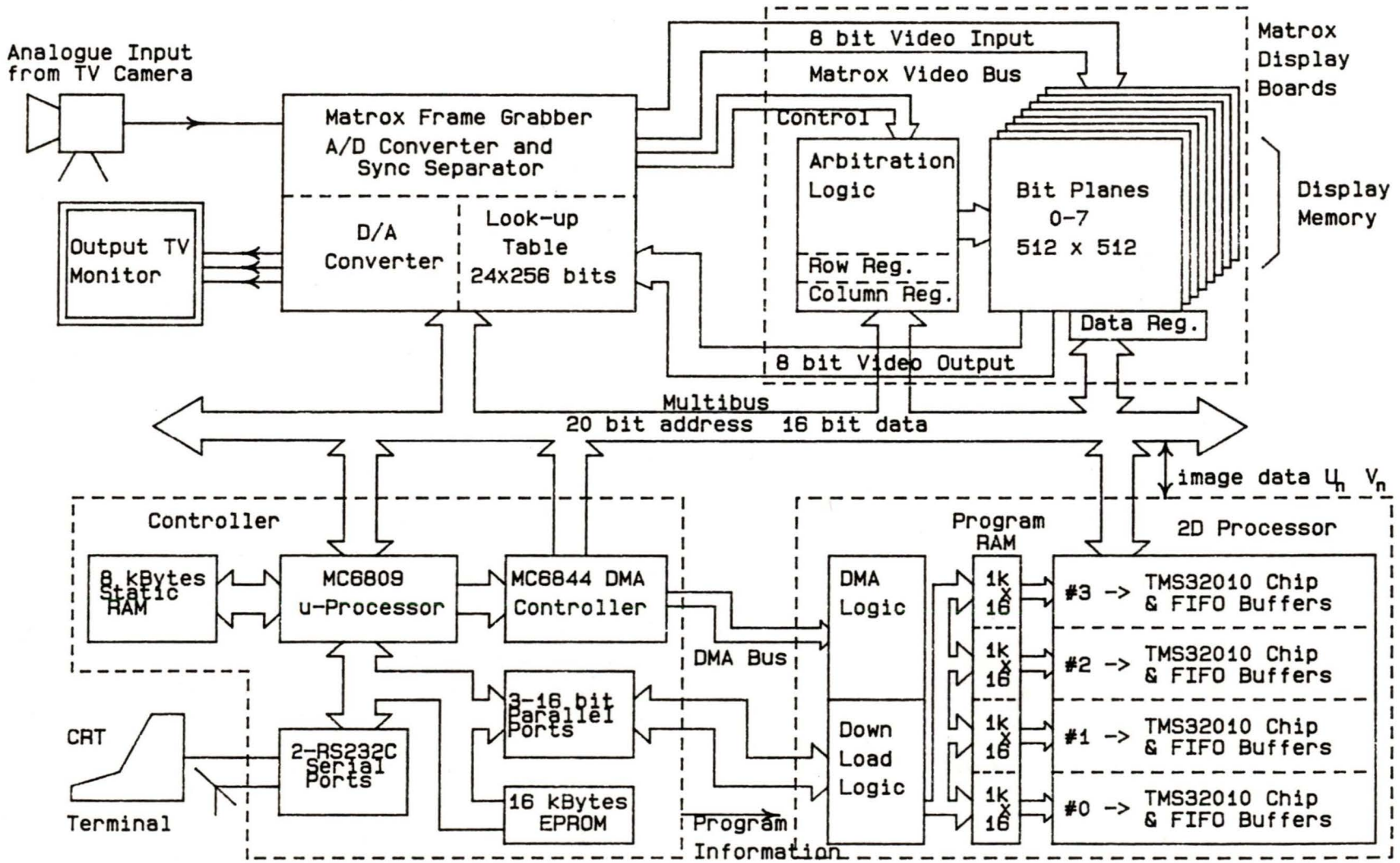


Figure 4.2 System Architecture of the Video Enhancement Instrument

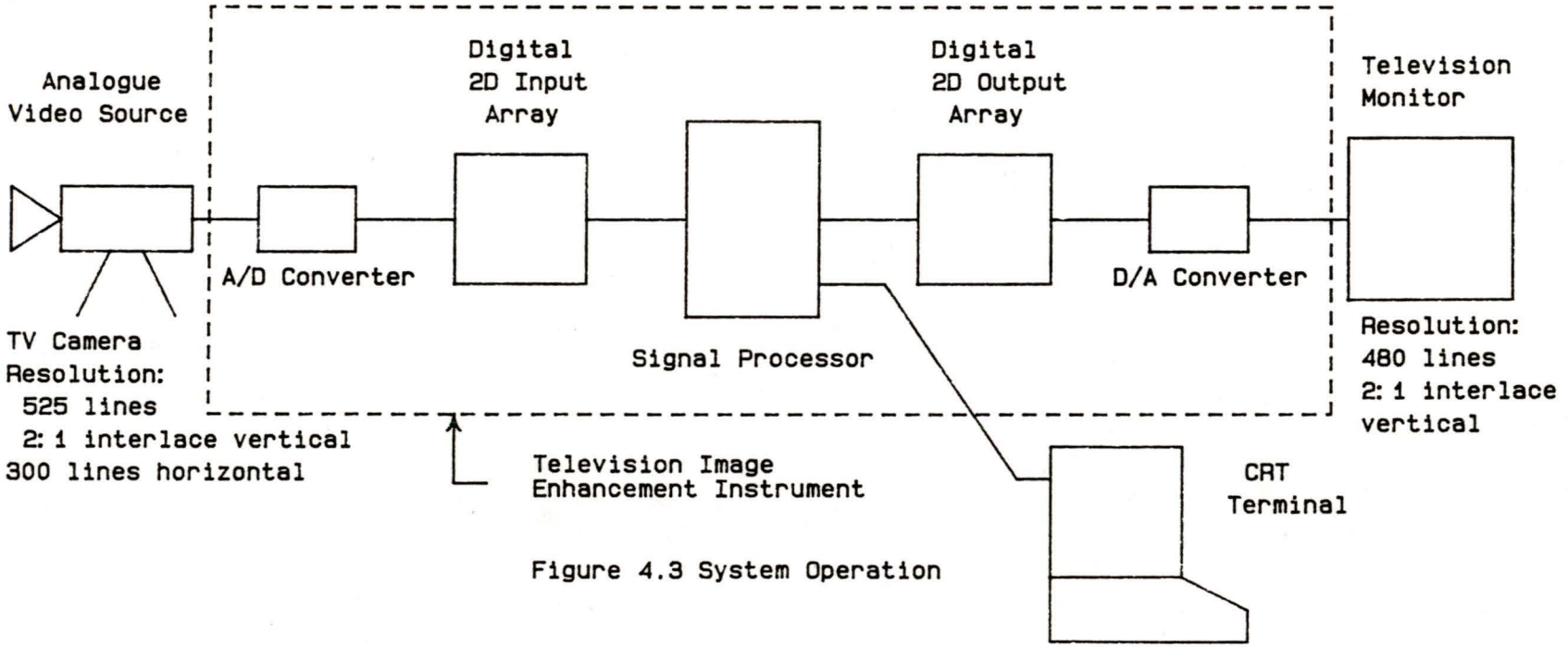


Figure 4.3 System Operation

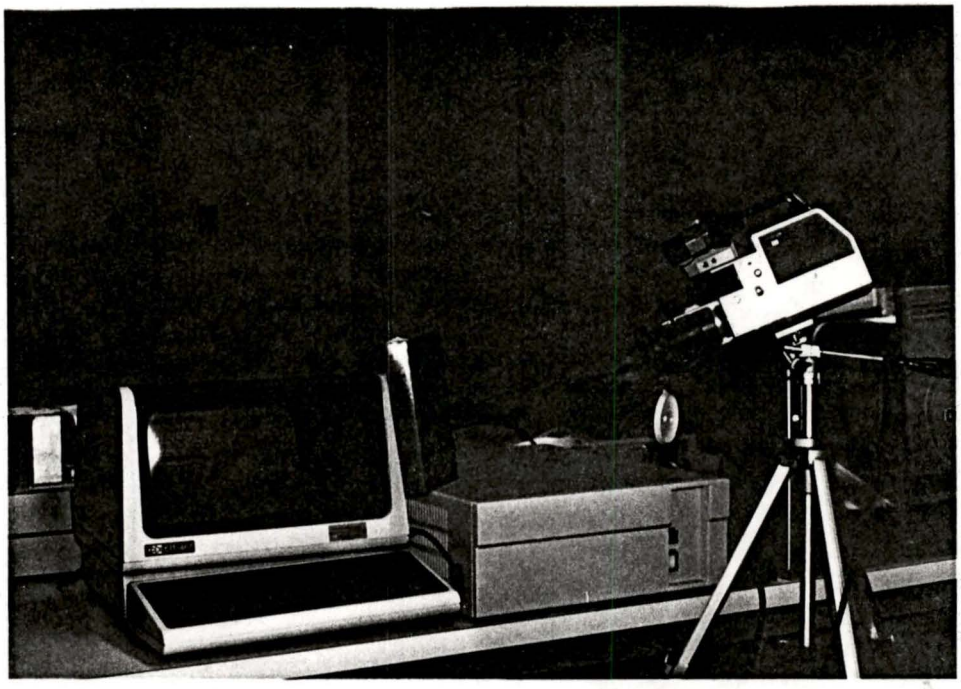


Figure 4.4 System For Image Enhancement

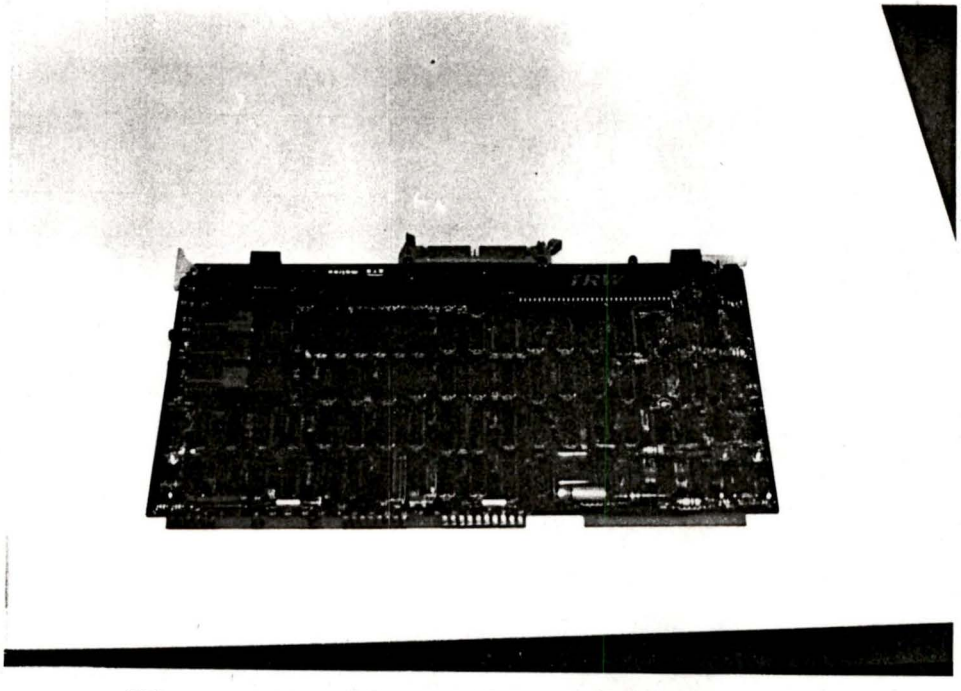
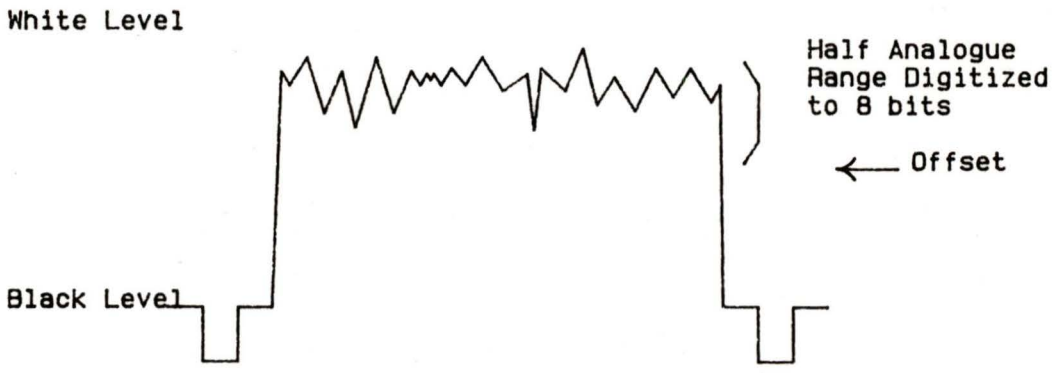


Figure 4.5 Matrox Frame Grabber

a) Digitization of Full Analogue Range



b) Localized Digitization of Analogue Signal

Figure 4.6 Digitization of Input Analogue Signal

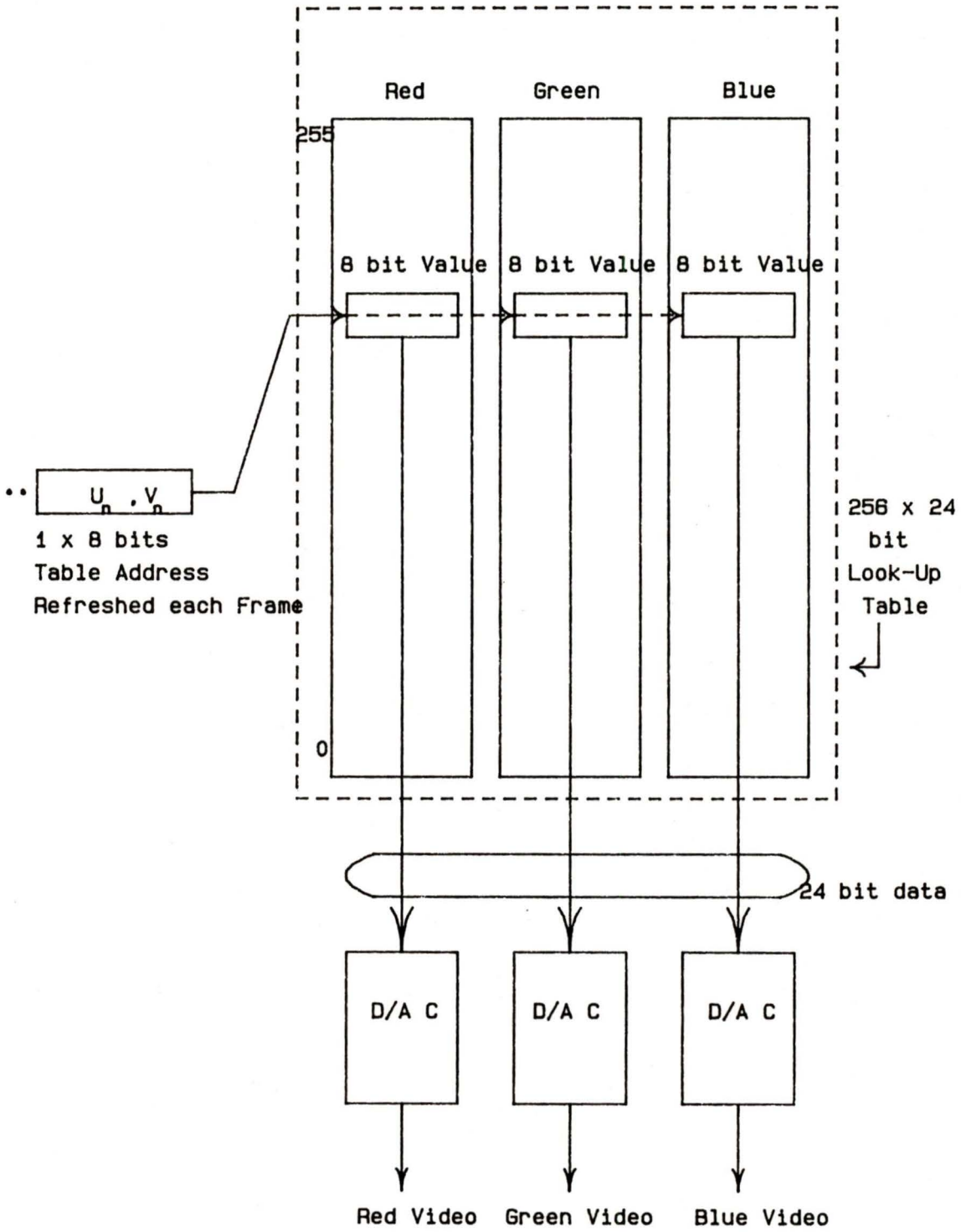


Figure 4.7 Look-up Table

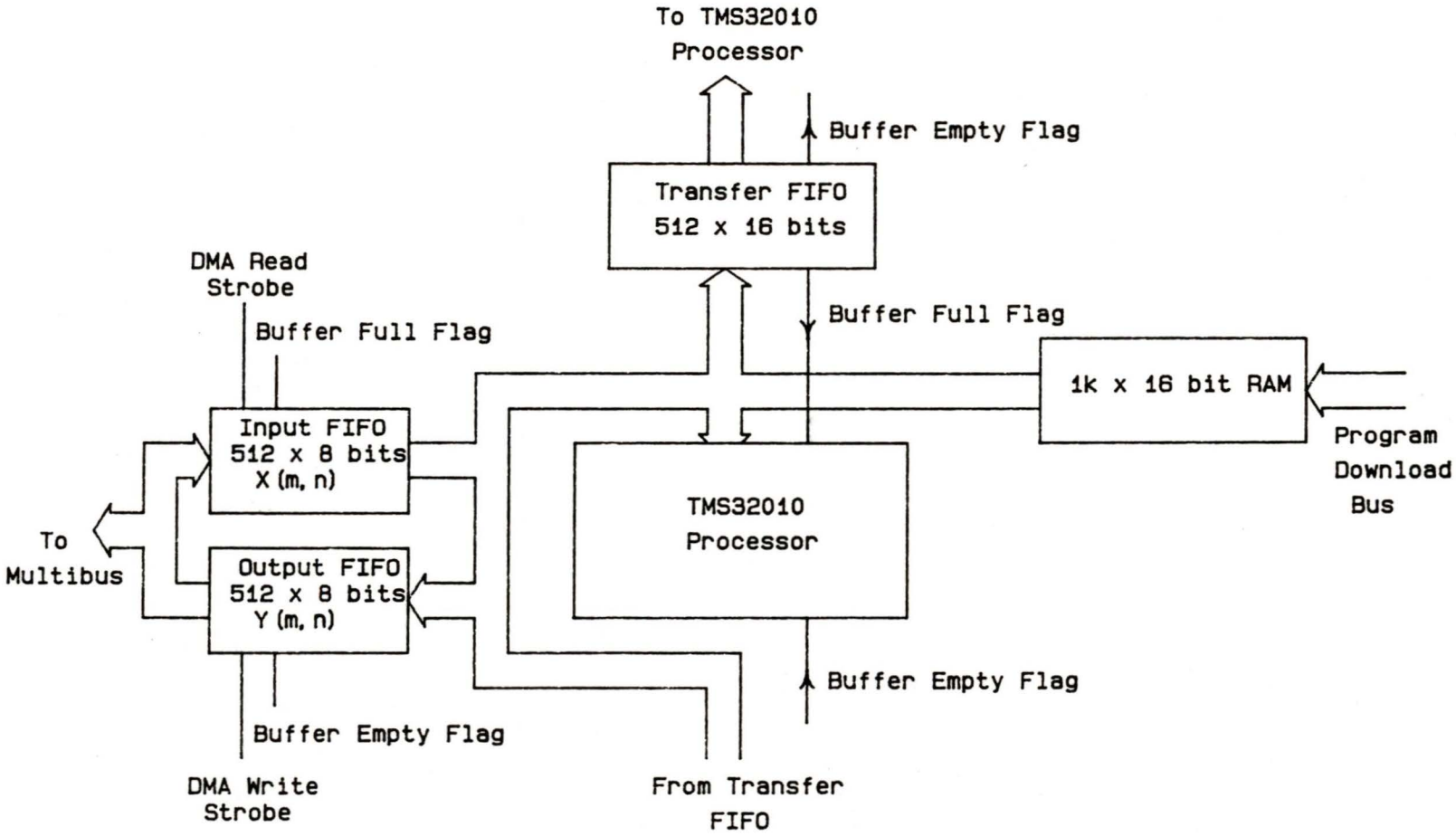


Figure 4.8 TMS32010 Chip and 3 FIFO Buffers

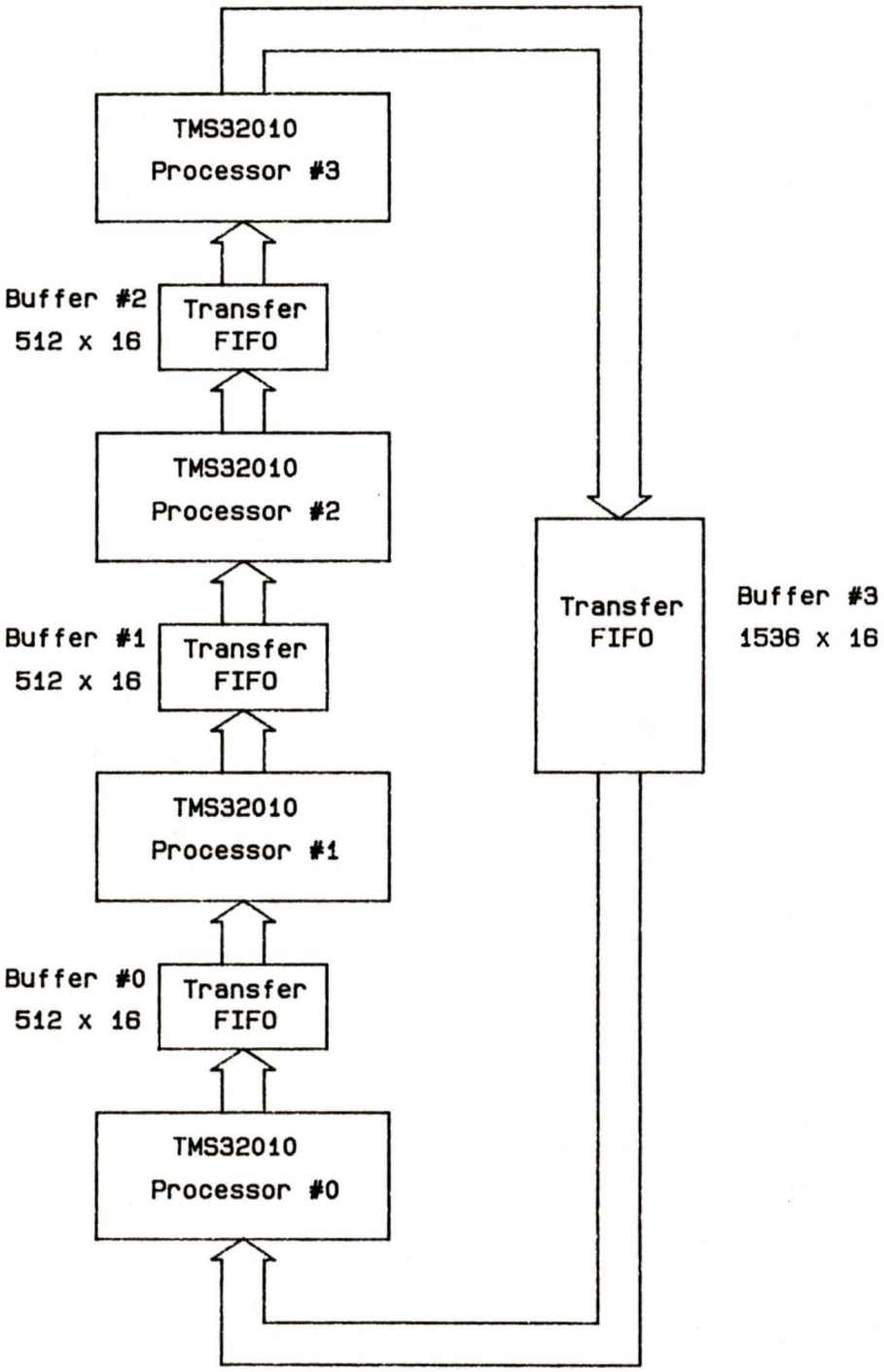


Figure 4.9 Interconnection of the Four TMS32010 Microprocessors and FIFO Buffers

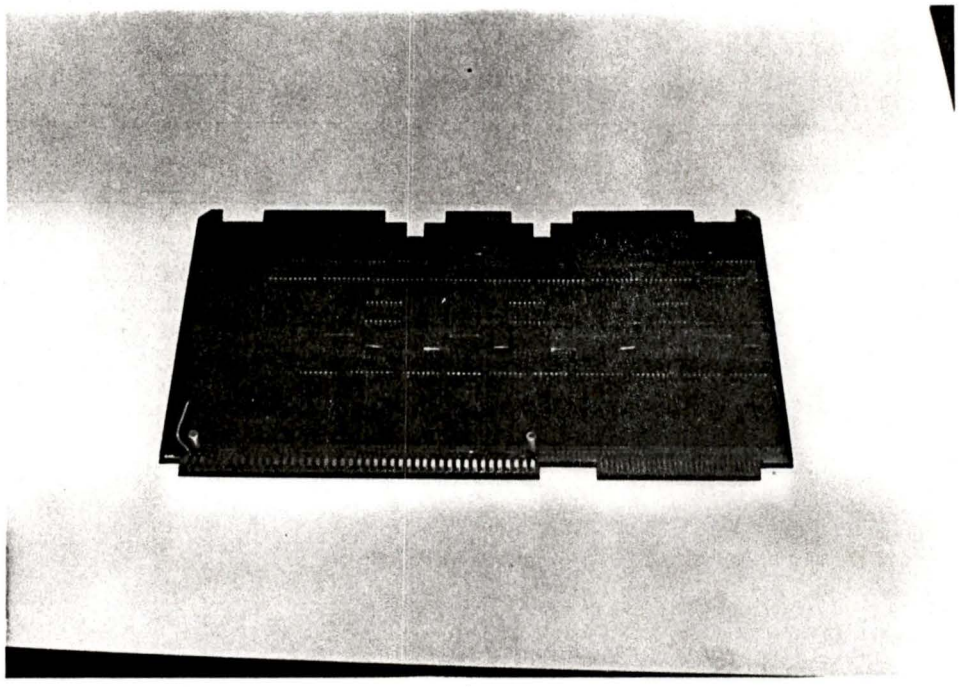


Figure 4.10 2D Processor Hardware

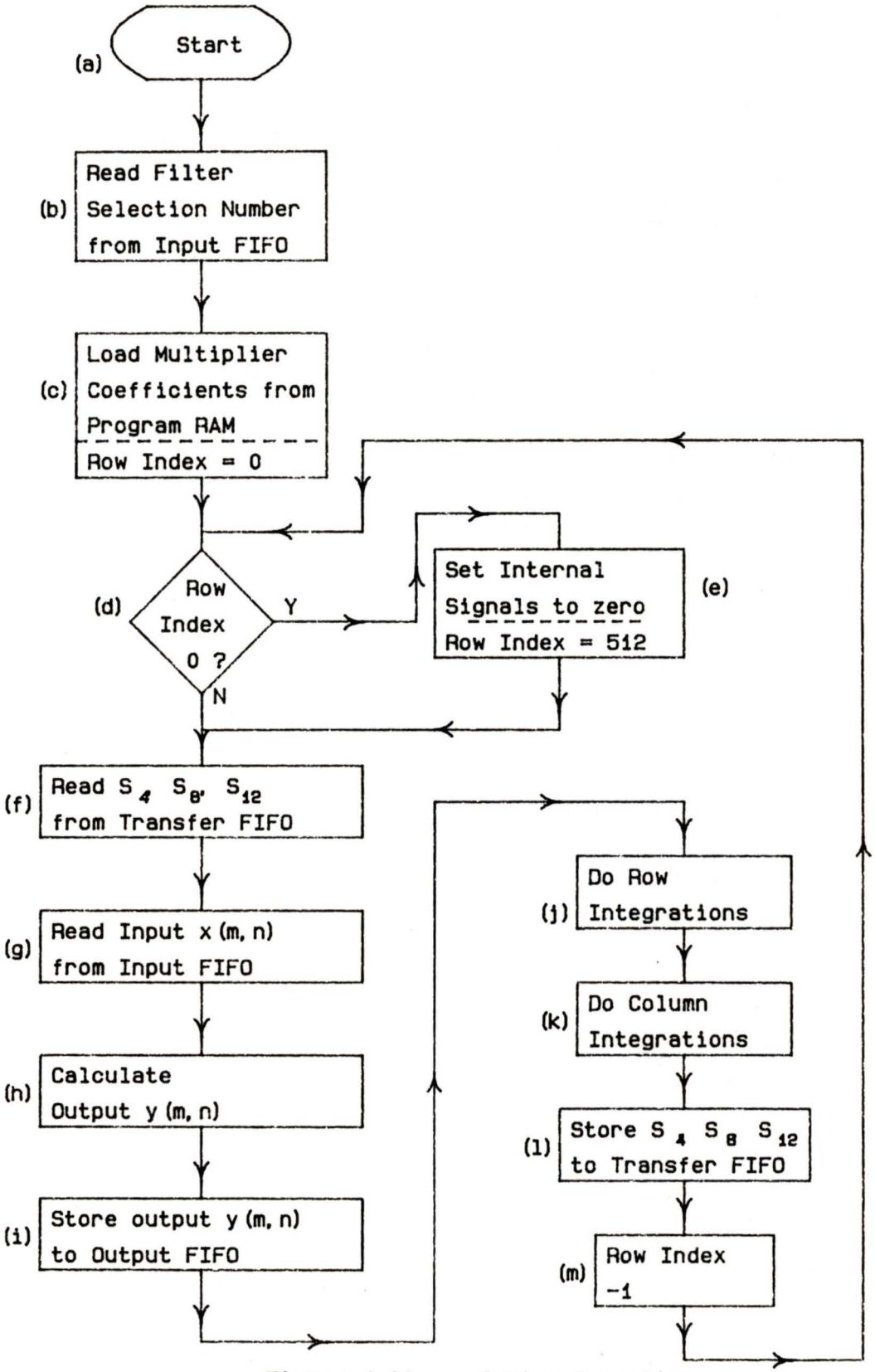


Figure 4.11 System Operation

5. FUNCTIONAL DESCRIPTION OF THE VIDEO ENHANCEMENT INSTRUMENT

5.1. FILTER CLASSES

The menu shown in Figure 5.1 indicates that the Instrument implements three major classes of filters, Lowpass, Highpass and Image Enhancement. A lowpass filter attenuates the frequency components of the input signal X that are above the cutoff frequency of the filter and will pass those frequency components that are below the cutoff frequency. The Instrument implements nine lowpass filters whose magnitude frequency responses are given in the Appendix. A highpass filter attenuates frequency components below the cutoff frequency. Because all the highpass filters used in the Instrument have a gain of two, the Instrument will amplify frequency components above the cutoff frequency. The magnitude frequency response of the nine highpass filters implemented by the Instrument are given in the Appendix.

The Instrument also implements a class of filters called *Image Enhancement filters.* An Image Enhancement filter has unity gain below the cutoff frequency and a gain of two above the cutoff frequency. Image Enhancement filters are a type of highpass filter except that they pass the low frequency background intensity components of the input signal X. It has been found

experimentally that objects in the output image Y are easier to recognize with Image Enhancement filters than with highpass filters because they have approximately the same intensity level as in the input image X. The magnitude frequency response of the nine Image Enhancement filters implemented by the Instrument are shown in the Appendix.

The user selects the desired class of filter from the menu shown in Figure 5.1 by pressing the appropriate number on the keyboard of the VDT.

5.2. CAPABILITIES OF THE VIDEO ENHANCEMENT INSTRUMENT

The subsection titles below correspond to the Video Enhancement Instrument menu items shown in Figure 5.2.

5.2.1. Image Digitization

The Instrument displays a B/W image from the input camera on the screen of the output TV monitor. The acquired image size is (512 x 512) pixels and the displayed image size is (485 x 512) pixels with the value of each pixel corresponding to a linear gray level from 0 black to 255 white. The Instrument displays the menu shown in Figure 5.3. The user may adjust the 16 levels of gain or the 16 levels of offset (discussed in Section 4.2 of chapter 4) or may capture a single digitized TV frame by pressing the appropriate number on the keyboard of the VDT.

5.2.2. Image Filtering

This selection allows the user to choose one of the nine possible cutoff frequencies for the class of filter chosen from the menu given in Figure 5.1. An Example of the menu used for this selection is shown in Figure 5.4. The Instrument processes the image in the Display Memory using the Spatial Integrator multi-processor algorithm and the 2D Processor described in chapters 3 and 4. The output image Y is displayed using the same memory that was occupied by the input image X.

5.2.3. Pan

The term *panning* is used to refer to movement of the display window in x and y, or both, coordinates. The Instrument panning function is available on the Matrox Display Memory board and allows any pixel $x(m,n)$ to be defined as the center of the screen. Panning allows for easier examination of an object that is along one of the edges of the screen.

5.2.4. Zoom

The *zoom* function, which is available on the Matrox Display Memory board, allows the upper left hand portion of the TV monitor display to be enlarged. The zoomed image replaces the original image from which it was taken. The resolution of the output image is not affected since it contains the same number of pixels before and after zooming. The zoom function

allows for magnifications of x2 and x4.

5.2.5. Post Processing

Upon the selection of the Post Processing option in Figure 5.2 the Instrument displays the menu shown in Figure 5.5. The three post processing choices are; 1) reverse the image for zero-phase filtering, 2) contrast stretch, 3) histogram stretch and shift.

The concept of zero-phase filtering is explained in chapter 1. To filter an image X with zero-phase using a 2D recursive LSI system having an impulse response of $h(m,n)$, the output image Y is filtered a second time using a filter having an impulse response of $h(-m,-n)$. The Instrument achieves zero-phase filtering by employing an *image reversal* operator $R[\,]$ that reverses the output image $Y \equiv y(m,n)$ into an intermediate image $Z \equiv z(m,n)$ where

$$z(m,n) = y(M-1-m, N-1-n) \tag{5.1}$$

where $0 \leq m \leq M-1$, $0 \leq n \leq N-1$, M is the maximum number of columns and N is the maximum number of rows for the spatially bounded image X. The reversed image Z is then processed a second time, using the same filter with impulse response $h(m,n)$. The output of this second filtering operation is then image-reversed using the operator $R[\,]$. This is exactly equivalent to filtering the image Y with a filter which has an impulse response of $h(-m,-n)$, therefore, zero-phase filtering of X has been achieved.

The Instrument is capable of changing the output Look-up Table (Section 4.4). The operation of *clipping and contrast stretching* involve the manipulation of the Look-up Table. In the case of clipping a pixel of value r, and all the values less than r, are forced to black and another pixel of value t, where $t > r$, and all pixels of value greater than t, are forced to white. In the case of contrast stretching, intermediate pixels s, where $r \leq s \leq t$, are varied linearly between black (pixel value = 0) and white (pixel value = 255). The Post Processing selection Contrast Stretch in Figure 5.5 allows the user to define values for r and t. For an output image in which all the pixel values are in a small range, contrast stretching can reveal hidden features that may be only a few intensity levels apart. The contrast stretch option in the Post Processing menu is *often* required to fully display the detail that exists in a video image.

The term *histogram* is used to refer to a graph of the relative frequency l_{freq} with which a gray level l occurs in an image X. The histogram of an image X that is quantized to K discrete levels can be represented as a bar graph with l usually displayed horizontally and l_{freq} displayed vertically. The image X will usually contain some levels that are highly populated; that is the relative frequency l_{freq} of certain levels will be large. An image that has a flat histogram has a high level of contrast so it is highly desirable to have all levels l with approximately the same relative frequency l_{freq}. The algorithm used by the Instrument to manipulate an image histogram is

given by

(1) calculate the relative frequency $l_{freq}(i)$ for each gray level i; that is, for each 8-bit pixel in image X count the number of pixels that have a value of i and repeat for $i = 0, 1, 2, 3, \cdots 255$.

(2) determine a new gray level distribution using

$$l_{new}(i) = INT \left[\sum_{j=0}^{i} \frac{K \cdot l_{freq}(j)}{M \cdot N} \right] \tag{5.2}$$

where $l_{new}(i)$ is the output gray level for a pixel of value i, M is the number of pixels per row, N is the number of pixels per column, and K is the number of gray levels per pixel.

5.3. SUMMARY

The Instrument contains multiplier coefficients for 27 filters. There are nine lowpass filters, nine highpass filters and nine Image Enhancement filters. An Image Enhancement filter is a type of highpass filter in which frequencies below the cutoff frequency have unity gain and frequencies above the cutoff frequency have a gain of two.

A description of the capabilities of the Video Enhancement Instrument is presented. The Instrument can capture and digitize a (512 x 512) by 8-bit Black and White TV image in 1/30 second. Output display routines for

zooming in on an object or panning an object into the center of the screen are provided. To facilitate zero-phase IIR filtering a routine for reversing an image in memory is provided. The Instrument implements two functions for improving the quality of low contrast output images. The first of the functions, contrast stretching, linearly varies the output Look-up Table between a lower threshold (black level) and an upper threshold (white level). The second function, histogram stretching and shifting, manipulates the histogram of an image so that it is a closer approximation to the desired histogram in which all gray levels contain the same number of pixels.

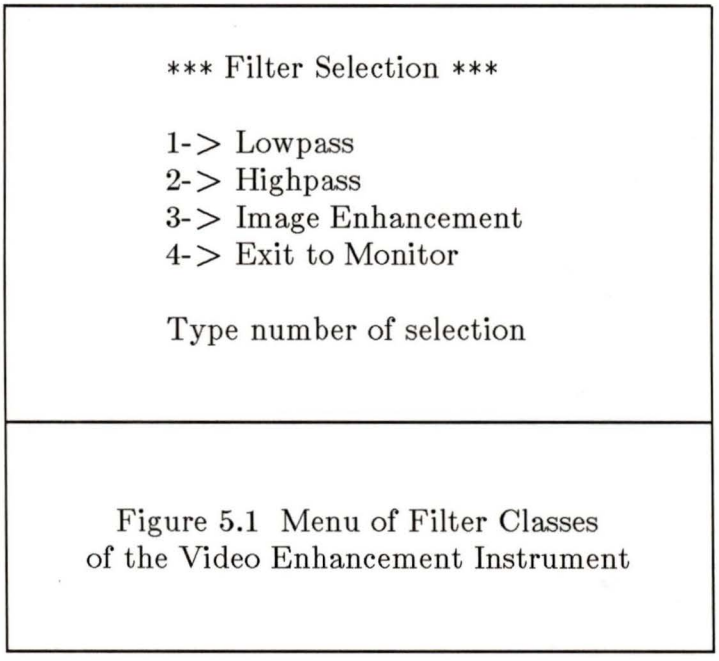


Figure 5.1 Menu of Filter Classes
of the Video Enhancement Instrument

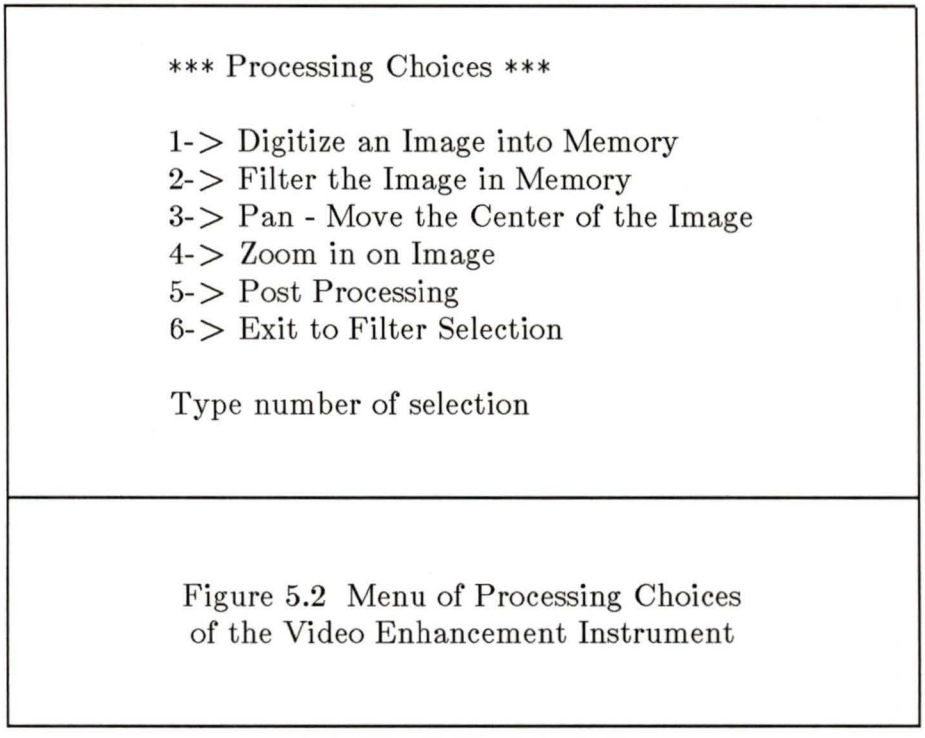


Figure 5.2 Menu of Processing Choices
of the Video Enhancement Instrument

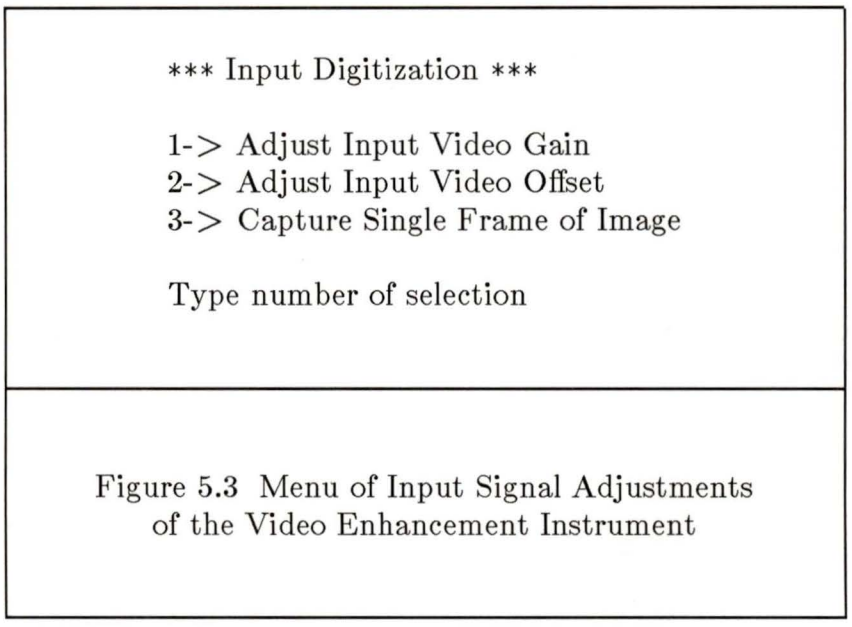


Figure 5.3 Menu of Input Signal Adjustments
of the Video Enhancement Instrument

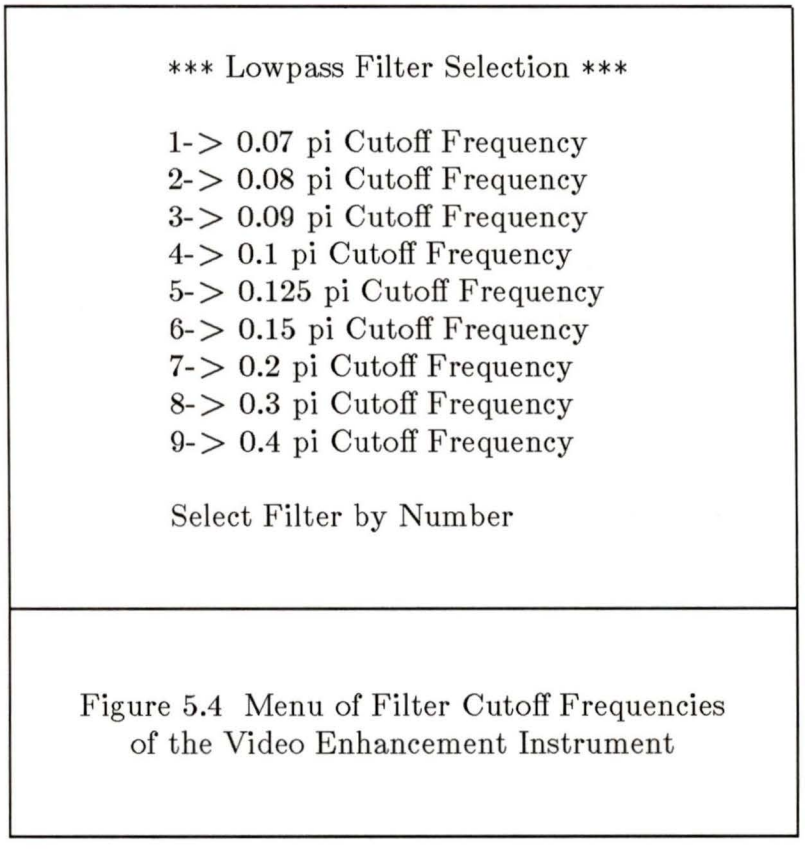


Figure 5.4 Menu of Filter Cutoff Frequencies
of the Video Enhancement Instrument

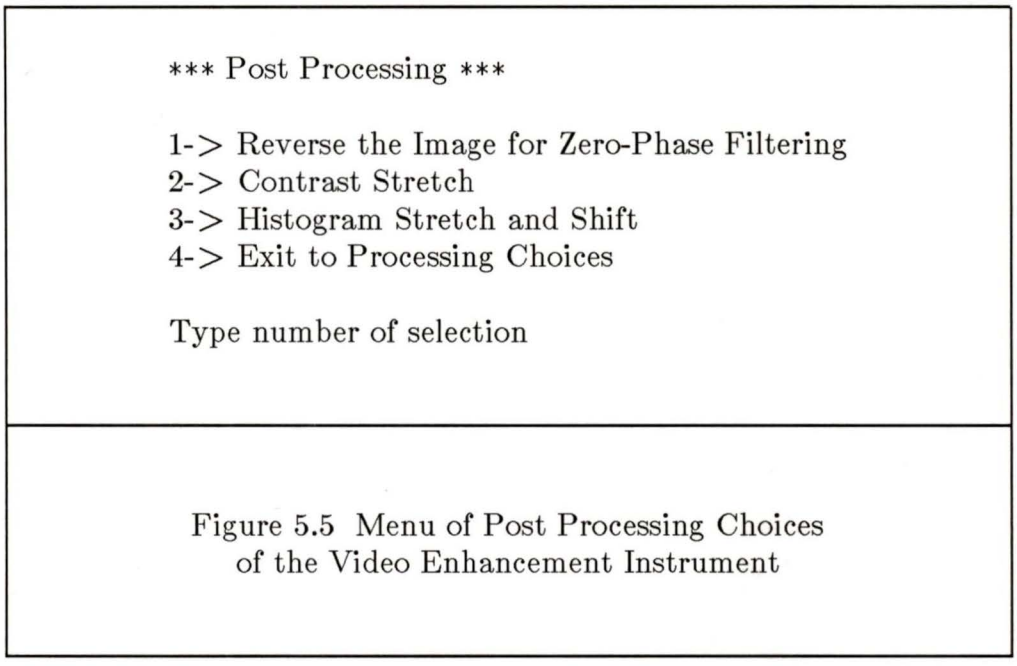


Figure 5.5 Menu of Post Processing Choices
of the Video Enhancement Instrument

6. CONCLUSIONS AND RECOMMENDATIONS

6.1. CONCLUSIONS

A Video Enhancement Instrument for enhancing television images using 2D LSI recursive filters is presented. An outstanding feature of the Instrument is its speed. The Instrument can digitize an image in 1/30 second, process a (512 x 512) by 8-bit image in approximately 4.0 seconds and then instantaneously display the results on the screen of a television monitor. The Instrument permits much faster processing than can be achieved on a typical dedicated supermini computer. Examples of applications include biomedical images such as CAT scan, ultra-sound, and X-ray images, industrial inspection in low illumination conditions, inspection of the ocean floor, radar images, sonar images and computer vision.

The Instrument uses 2D recursive filters because they are computationally efficient when compared to other methods such as 2D nonrecursive filters and 2D filtering techniques using the FFT. Recursive filters may be implemented with a minimum of scratch pad storage and they may start processing an image immediately after the first pixel has been digitized.

For many years, a major disadvantage of 2D recursive filtering was ensuring the stability of the difference equation. However, stable 2D recur-

sive filters for any transfer function can be obtained using the Ramamoorthy/Bruton method [3,4]. The multiplier coefficients of the filters used in the design of the Instrument were obtained using a program developed by Bruton/Bartley that is an implementation of the Ramamoorthy/Bruton method called 2DFil [8]. If other filter classes or cutoff frequencies are required, 2DFil can generate the required multiplier coefficients.

Preliminary studies on the Spatial Integrator SFG structure suggested that it would be capable of implementing 2D filters with much shorter multiplier coefficient word lengths [5,13]. This motivated this investigation of the Spatial Integrator structure for high-speed 2D filtering because shorter word length arithmetic is easier to implement.

In this context, the main contributions of this thesis are:

(1) A detailed experimental investigation of the Spatial Integrator SFG structure and the Direct Form SFG structure. This investigation shows the improved performance of the Spatial Integrator structure over the Direct Form structure under actual image processing conditions. Spatial Integrator recursive filters of order (3 x 3) may be implemented using 16-bit multiplications and 32-bit additions. The widely used Direct Form structure requires 20-

bit multiplications and 34-bit additions. The results show that the low sensitivity of the Spatial Integrator structure to quantization of multiplier coefficients makes possible a 16-bit single precision arithmetic implementation.

(2) An algorithm has been developed for a multi-processor implementation. It is shown by examination of row-recursive 2D filtering, that the $(n+1)$ th row of an image may be processed concurrently with the n th row, providing the recursion in the $(n+1)$ th row follows by one pixel the recursion in the n th row. For a highly parallel system which contains M row processors, it is shown that square images may be processed at $T \cdot (2M-1)$ seconds per image, where M is the maximum number of pixels per row or column and T is the time taken to process 1 pixel. If one processor is used the total time to process an image is M^2T seconds. Typically for $T = 26.4 \cdot 10^{-6}$ and $M = 512$, multi-processing reduces the processing time from 6.92 seconds for one processor to $27.0 \cdot 10^{-3}$ seconds.

(3) An Instrument for 2D LSI filtering has been *designed and constructed.* The Instrument uses the Spatial Integrator SFG structure and the multi-processor algorithm to

enhance digital television images in approximately 4 seconds. It uses four high performance 16-bit microprocessors operating in parallel to perform the image enhancement.

6.2. RECOMMENDATIONS

The primary limitation of the system is its inability to archive images. To remedy this situation would involve incorporating a fixed disk to store images for further processing and/or for image comparison. A multi-megabyte disk is desirable since each (512 x 512) by 8-bit image occupies 256 Kbytes. The system would also require a tape drive or floppy disk for system back-up and long term storage. A disk operating system is necessary for the storage, retrieval and removal of image files from the disk.

It would also be advantageous to have several RAM (512 x 512) image buffers so that images may be easily added or subtracted. This amount of memory (256 Kbytes per image) would be most easily handled by incorporating a central processor with a larger address space such as the MC68000 or the 8086.

The second RS232C port on the Controller of the Instrument could be used to transmit an image serially to a remote computer. The incorporation of a parallel communications interface is also desirable to increase the rate of image transmission.

For transmission or archiving of large numbers of images, data compression techniques should also be investigated and incorporated into the image processing system.

The Instrument could also easily incorporate an operation called density slicing, where only pixels in a certain intensity range are displayed. The displayed pixels could be color coded through the use of the Look-up Table. Color coding aids in the interpretation of the results.

Most of the abovementioned improvements are easily incorporated using commercially available software and such hardware as disk controller boards, megabyte memory boards and single board computers.

REFERENCES

[1] Dan E. Dudgeon, Russell M. Mersereau, "Multidimensional Digital Signal Processing", Englewood Cliffs, N. J., Prentice-Hall, Inc., 1984.

[2] Alan V. Oppenheim, Ronald W. Schafer, "Digital Signal Processing", Englewood Cliffs, N. J., Prentice-Hall, Inc., 1975.

[3] P.A. Ramamoorthy, L.T. Bruton, "Design of Stable Two-dimensional Analogue and Digital Filters with Applications in Image Processing" *Int. Jour. on Circuit Theory and Applications,* Vol 7, pp. 229-245, 1979.

[4] P.A. Ramamoorthy, L.T. Bruton, "Design of Two-dimensional Recursive Filters," *Topics in Applied Physics,* Vol. 42, Two-dimensional Digital Signal Processing I: Linear Filters, T.S. Huang, ed., Springer-Verlag, New York, N. Y., pp. 41-83, 1981.

[5] L.T. Bruton, T.C. Strecker, "Two-dimensional Descrete Filters Using Spatial Integrators", *IEE Proceedings* Vol. 130, Pt. G, no. 6 (Dec. 1983), pp. 271-275.

[6] Alfred Fettweis, "Digital Circuits and Systems", *IEEE Trans. Circuits and Systems,* CAS-31, no. 1 (Jan. 84), pp. 31-48.

[7] Andreas Antoniou, "Digital Filters: Analysis and Design", New York, N. Y., McGraw-Hill, Inc., 1979.

[8] L.T. Bruton, N.R. Bartley, "A General-Purpose Computer Program for the Design of Two-dimensional Recursive Filters - 2DFil", *Circuits, Systems and Signal Processing,* Vol. 3, no. 2, pp. 243-264, 1984.

[9] "VAF-512 Graphics Support Board", Manual no. 176-A50-02/0, Matrox Electronic Systems Ltd., Montreal, Quebec, Canada.

[10] "RGB-GRAPH Color Graphics Controller", Manual no. 167-A50-05A/2 Matrox Electronic Systems Ltd., Montreal, Quebec, Canada.

[11] "TMS32010 User's Guide", Part no. SRU001A, Texas Instruments Inc., 1983.

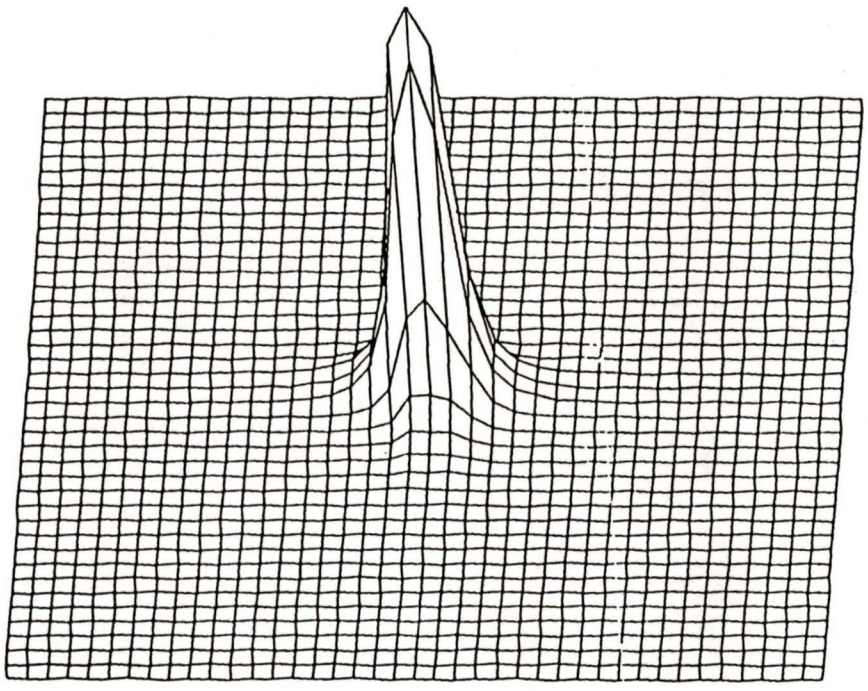
[12] K. McDonough, E. Caudel, S. Magar, A. Leigh, "Microcomputer with 32-bit Arithmetic does High-precision Number Crunching", *Electronics,* Vol. 55, no. 4 (Feb. 24, 82), pp. 105-110.

[13] T.C. Strecker, "2D Discrete Filters Using Integrators", MSc. Thesis, Dept. of Electrical Engineering, University of Calgary, Calgary Alberta, April 1983.

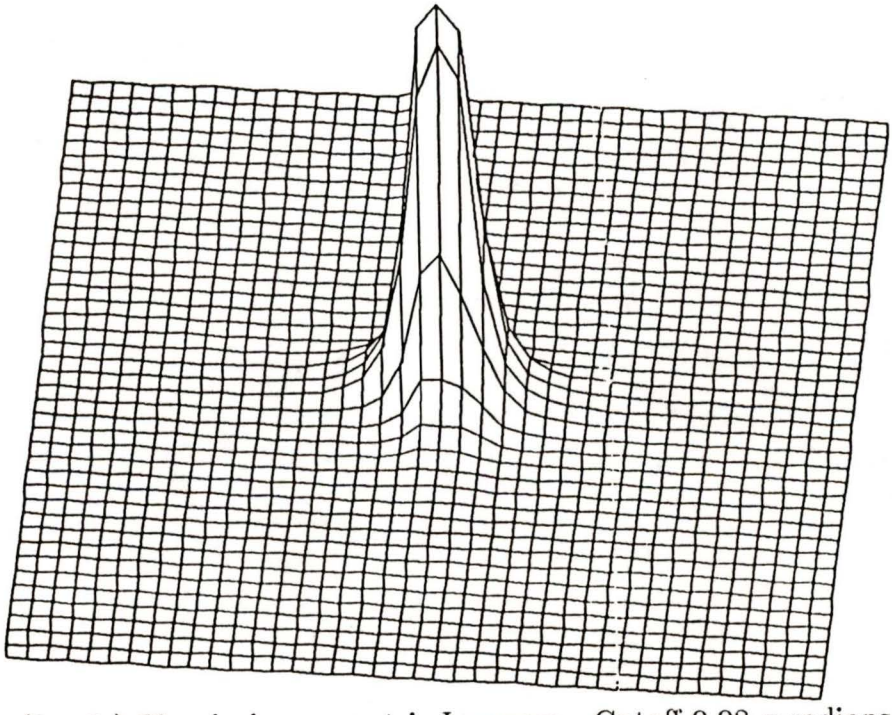
[14] Tran Thong, "Digital Image Processing Test Patterns", *IEEE Trans. Acoustics, Speech, and Signal Processing,* ASSP-31, no. 3 (June 83), pp. 763-766.

[15] Azriel Rosenfeld, Avinask C. Kak, "Digital Picture Processing", Vol. 1, New York, N. Y., Academic Press, Inc., 1982.

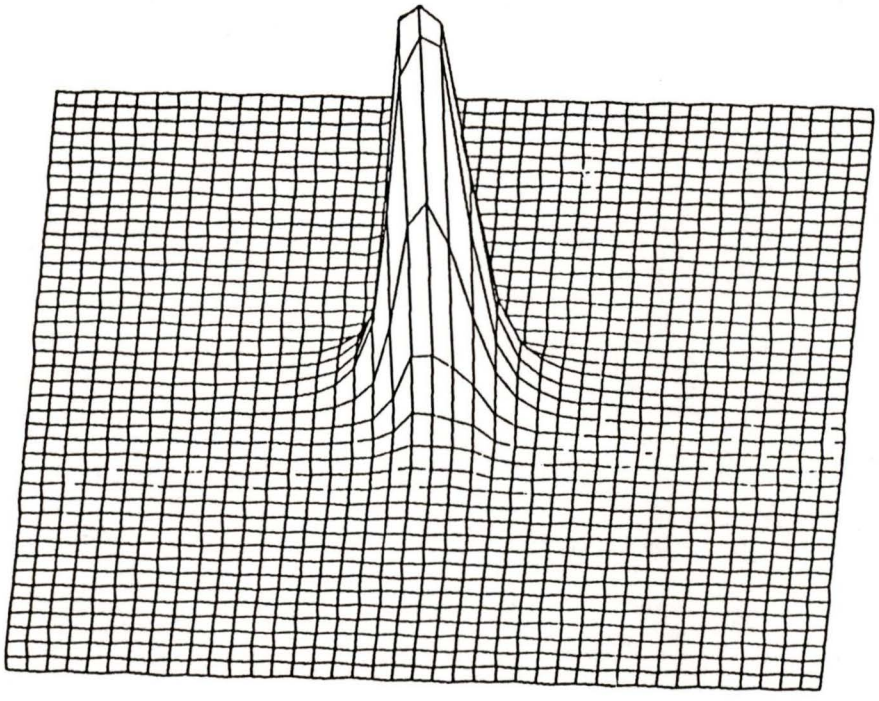
APPENDIX



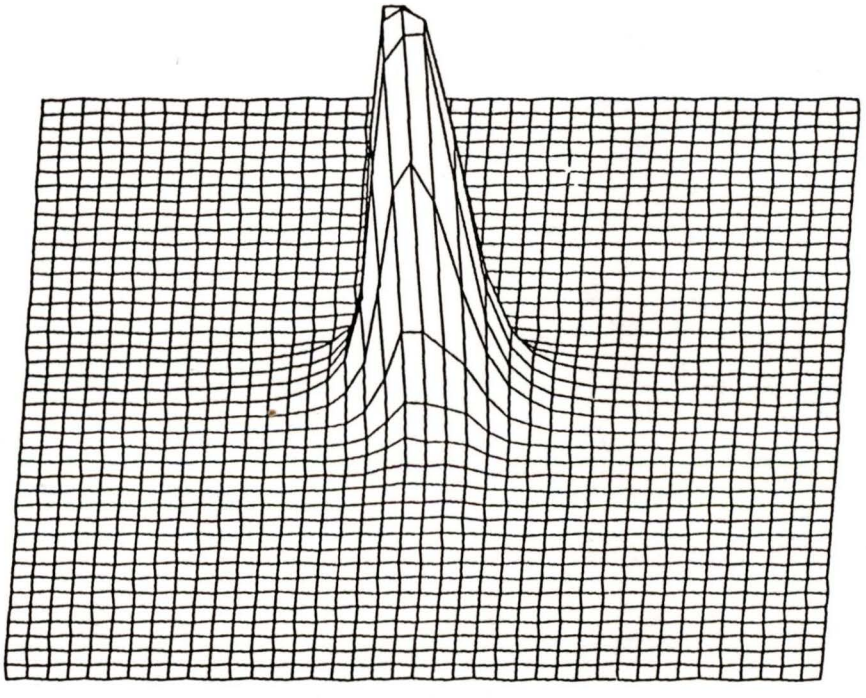
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.07 π radians



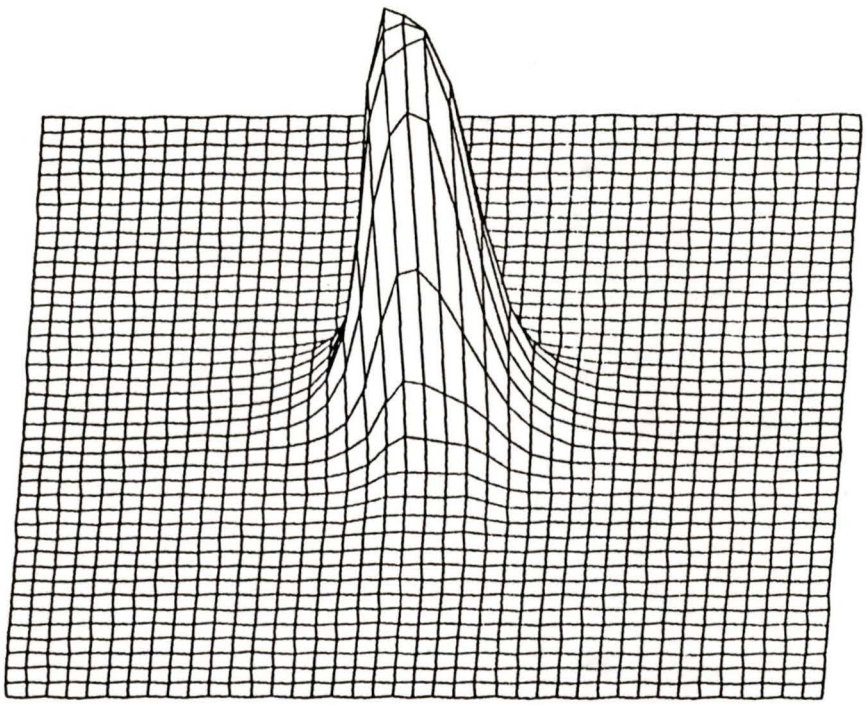
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.08 π radians



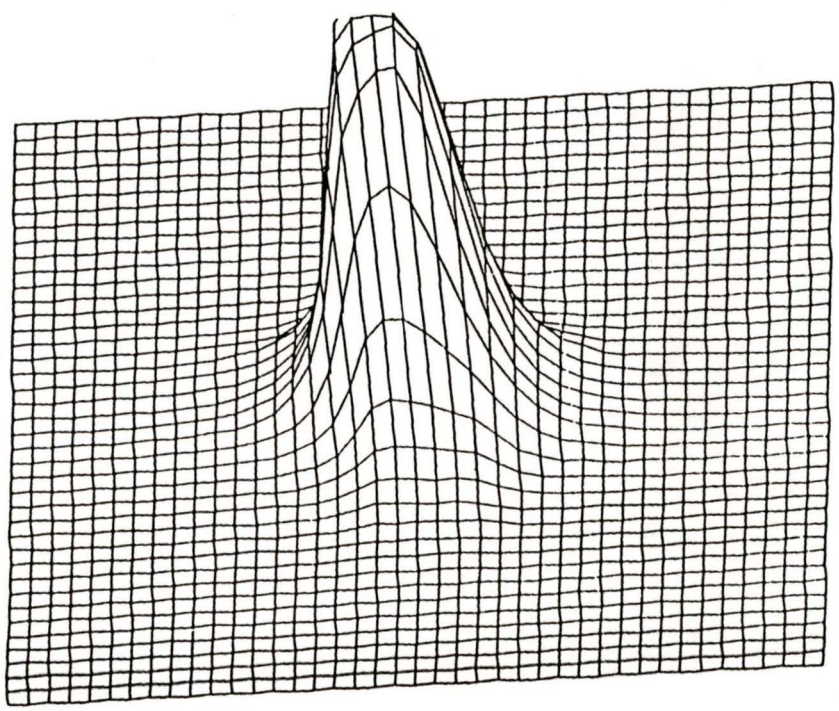
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.09 π radians



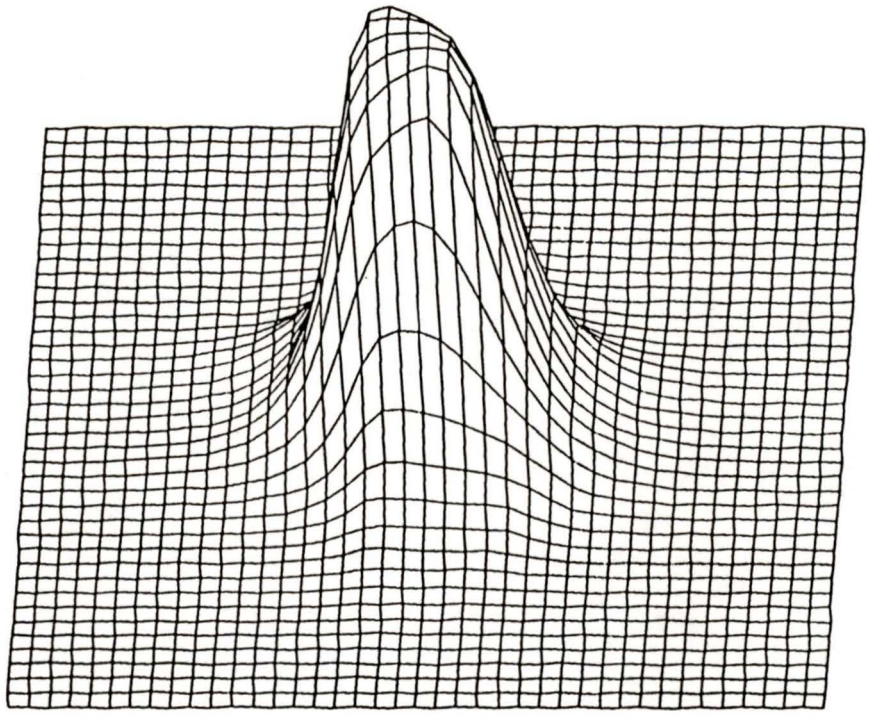
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.1 π radians



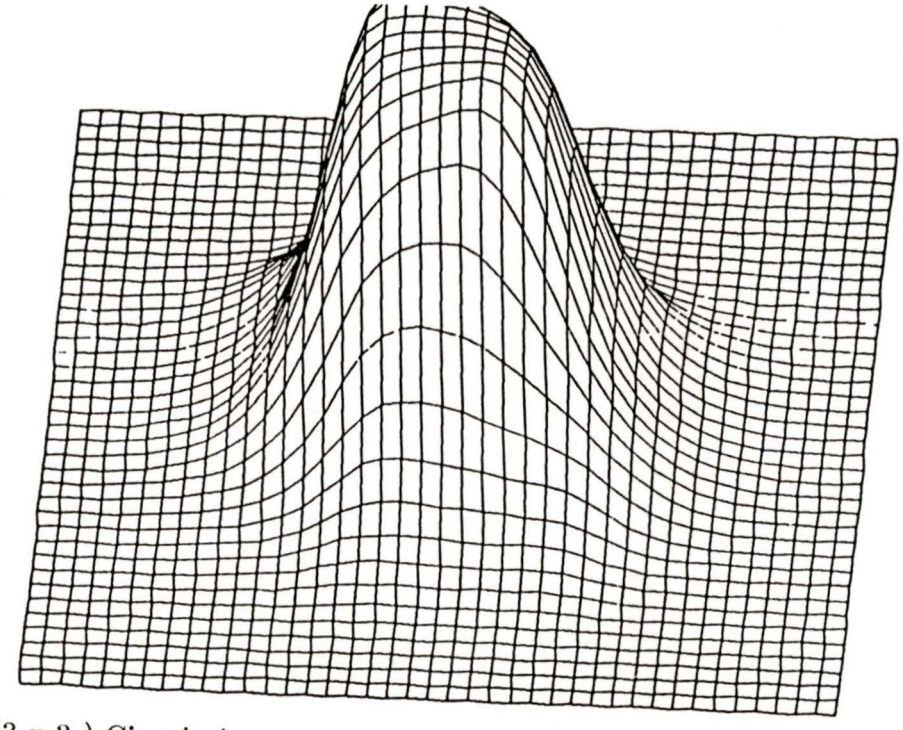
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.125 π radians



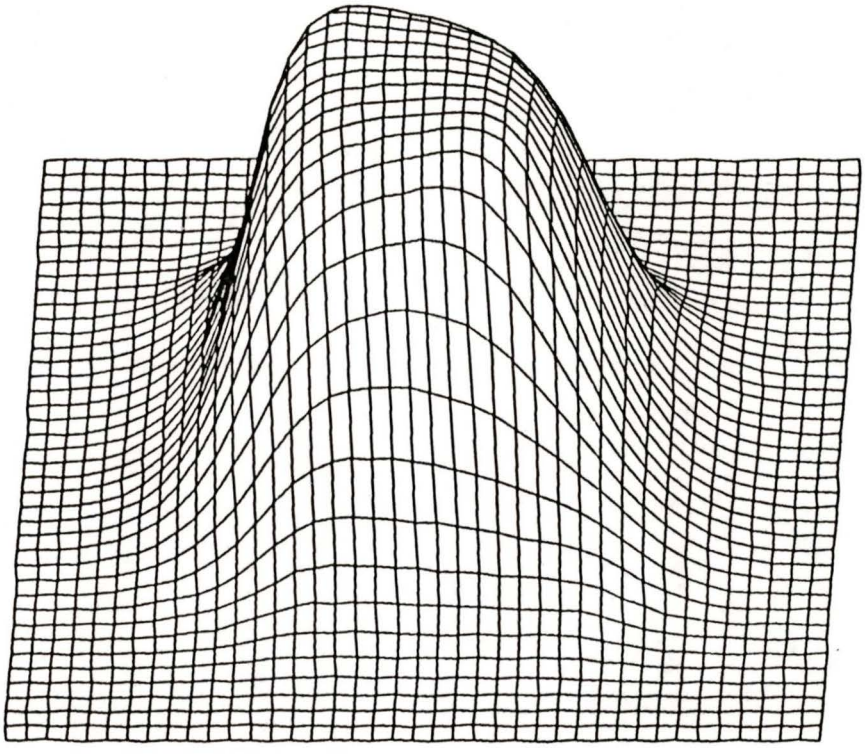
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.15 π radians



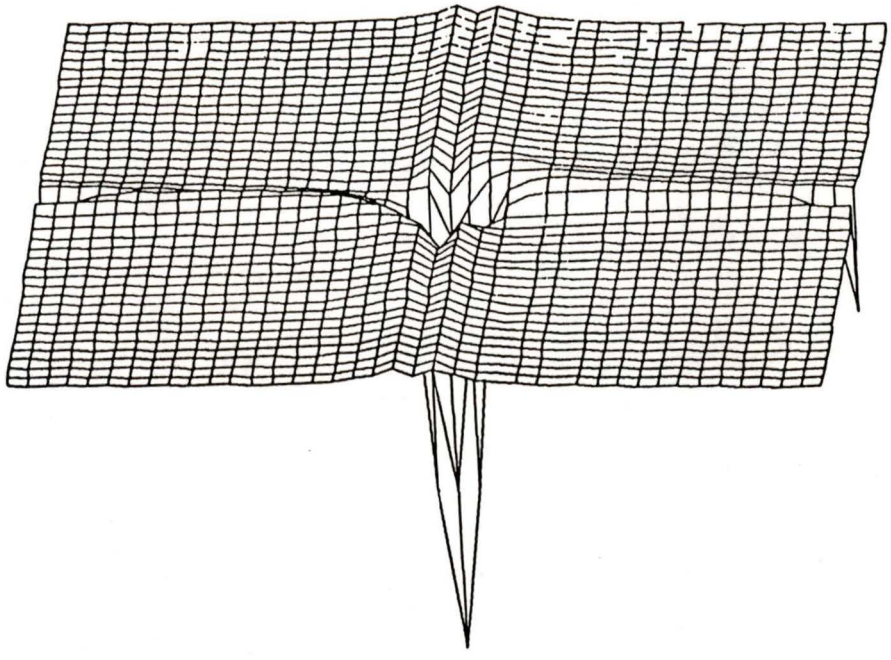
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.2 π radians



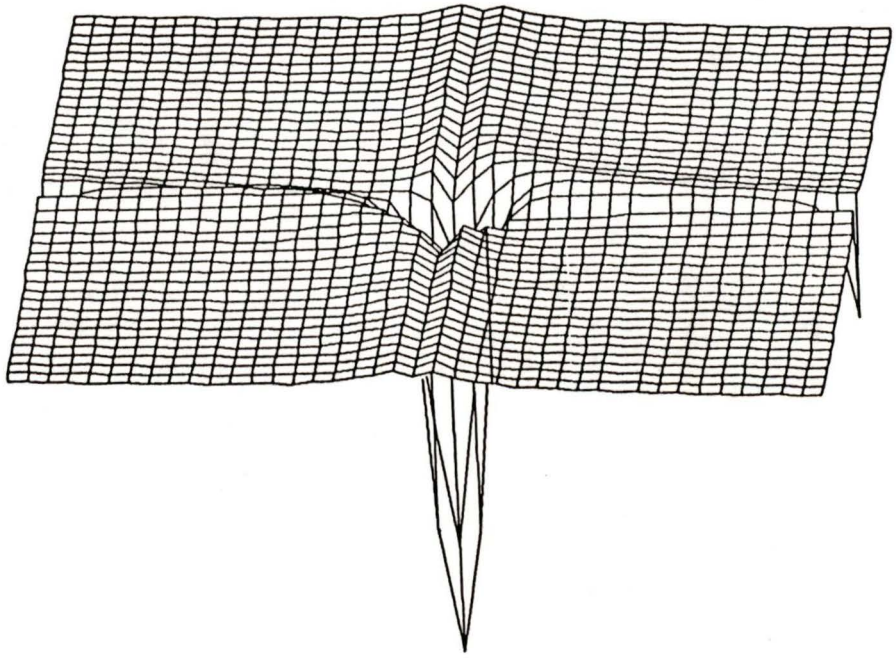
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.3 π radians



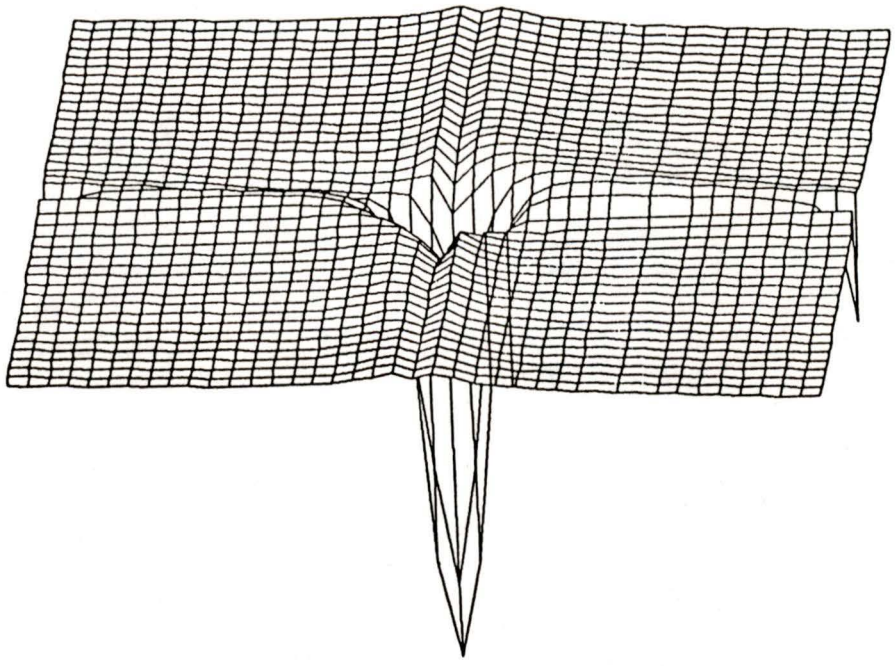
(3 x 3) Circularly symmetric Lowpass - Cutoff 0.4 π radians



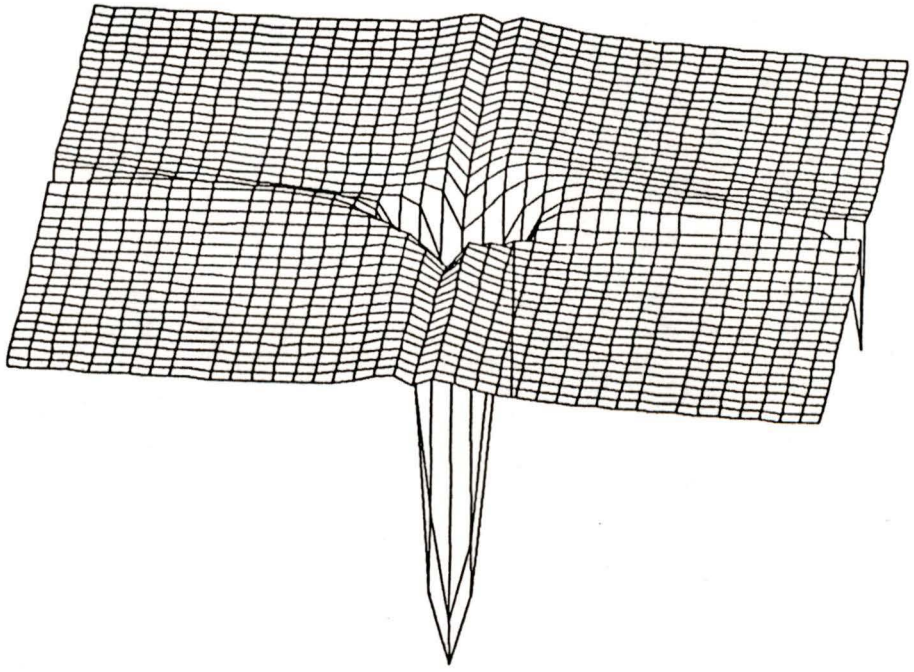
(3 x 3) Circularly symmetric Highpass - Cutoff 0.07 π radians



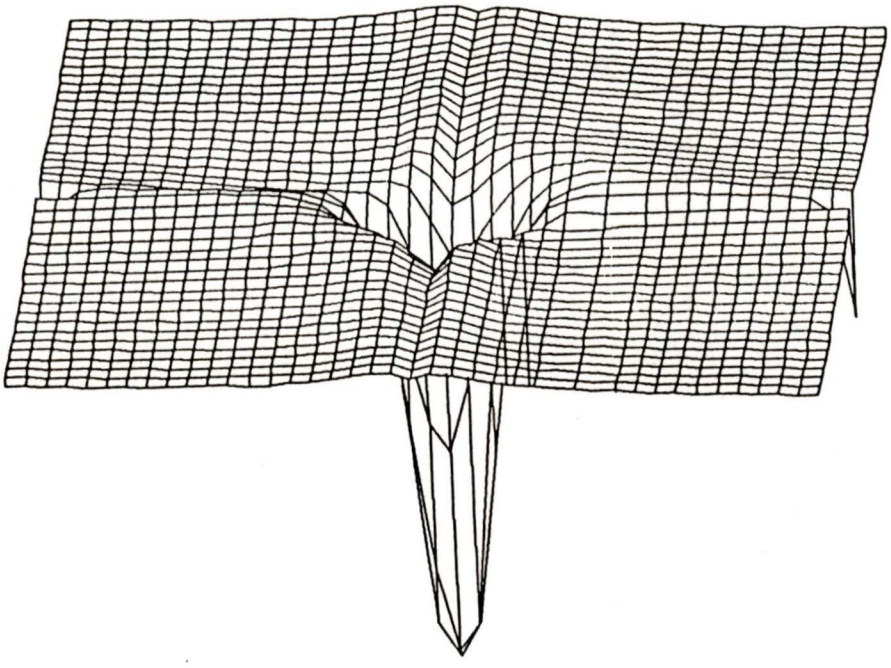
(3 x 3) Circularly symmetric Highpass - Cutoff 0.08 π radians



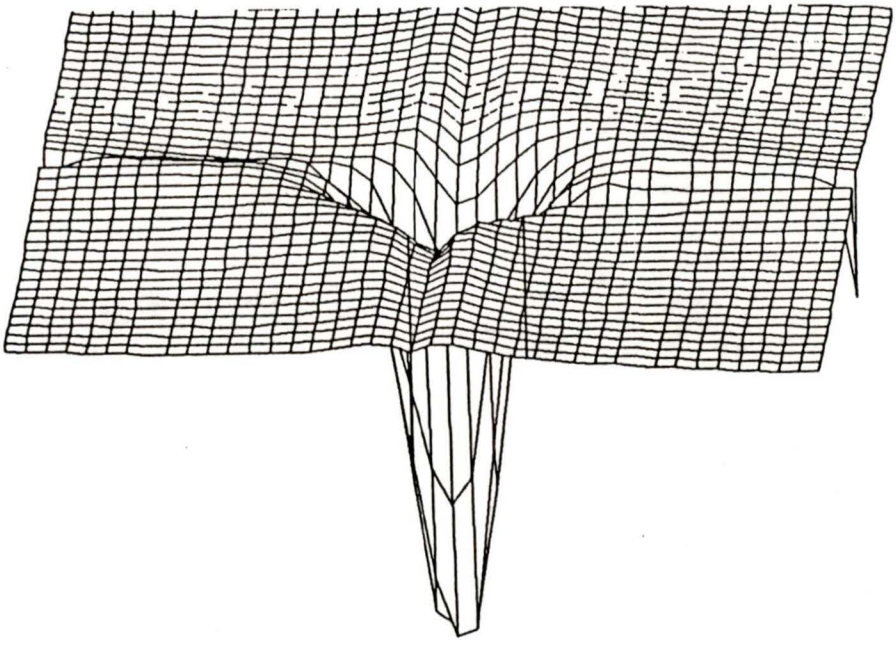
(3 x 3) Circularly symmetric Highpass - Cutoff 0.09 π radians



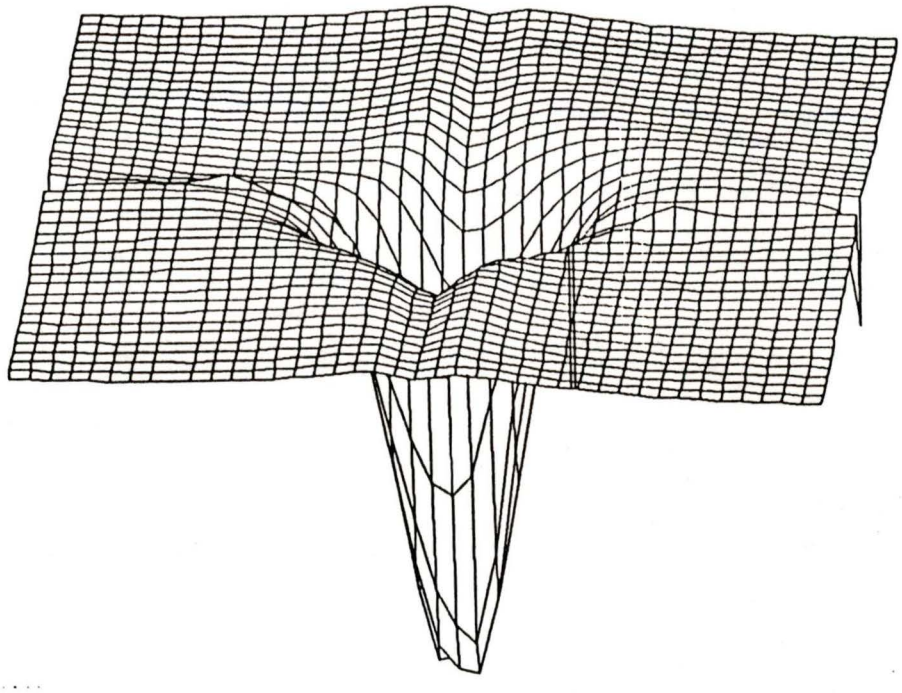
(3 x 3) Circularly symmetric Highpass - Cutoff 0.1 π radians



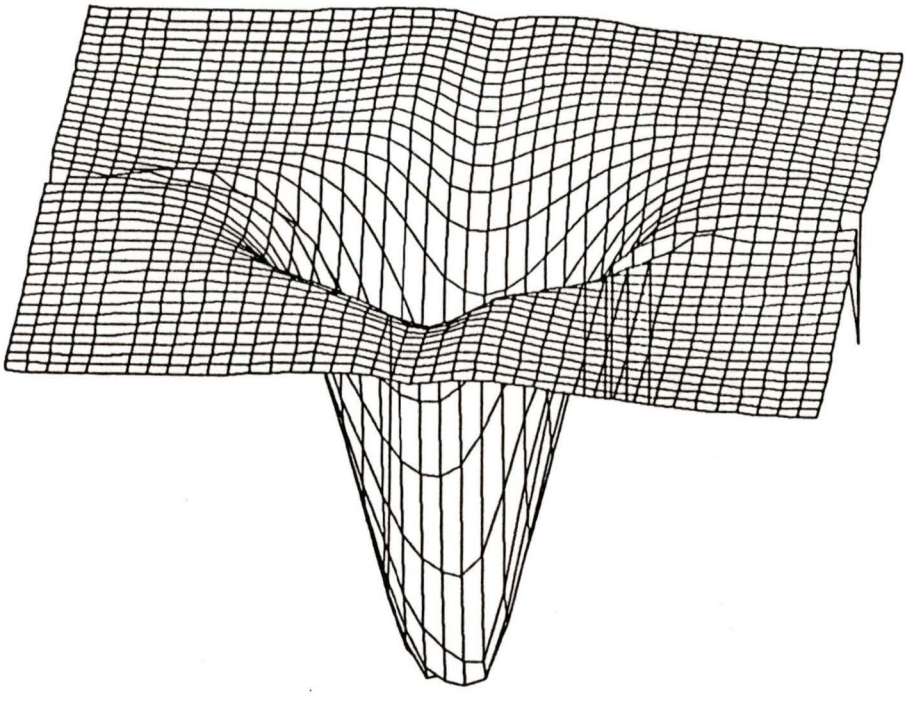
(3 x 3) Circularly symmetric Highpass - Cutoff 0.125 π radians



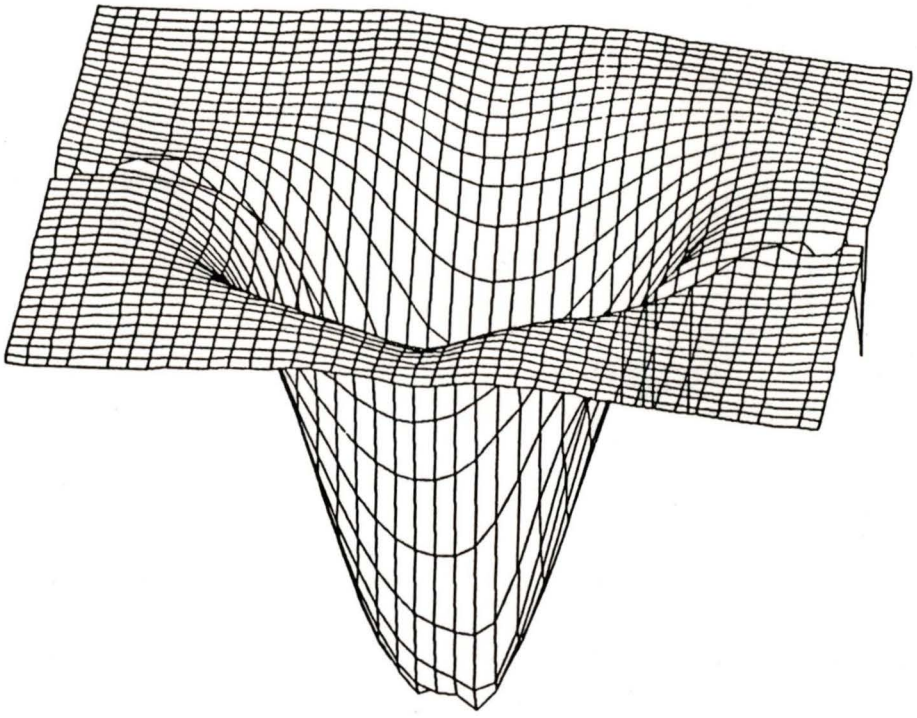
(3 x 3) Circularly symmetric Highpass - Cutoff 0.15 π radians



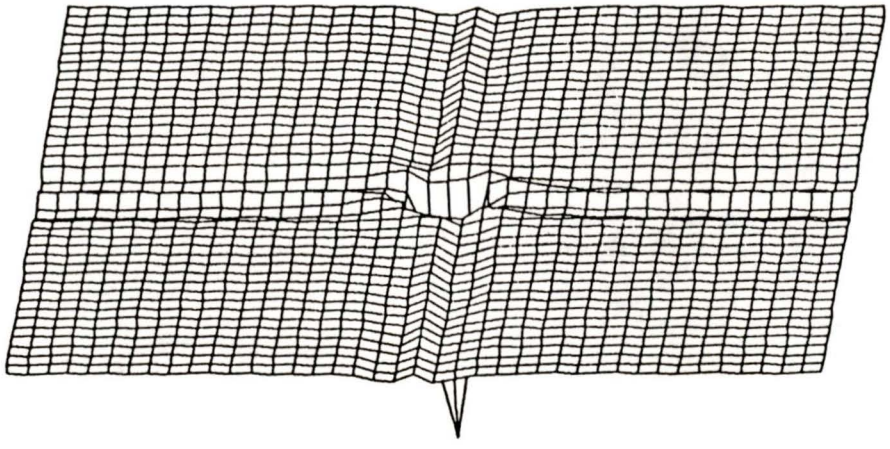
(3 x 3) Circularly symmetric Highpass - Cutoff 0.2 π radians



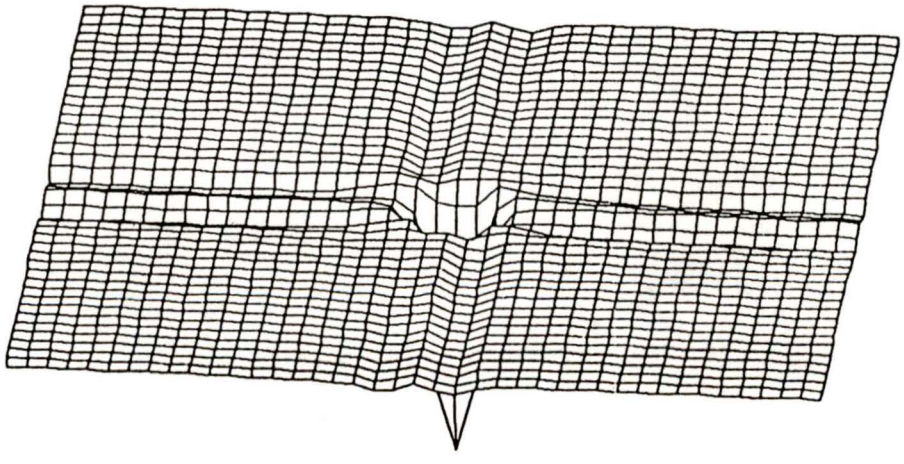
(3 x 3) Circularly symmetric Highpass - Cutoff 0.3 π radians



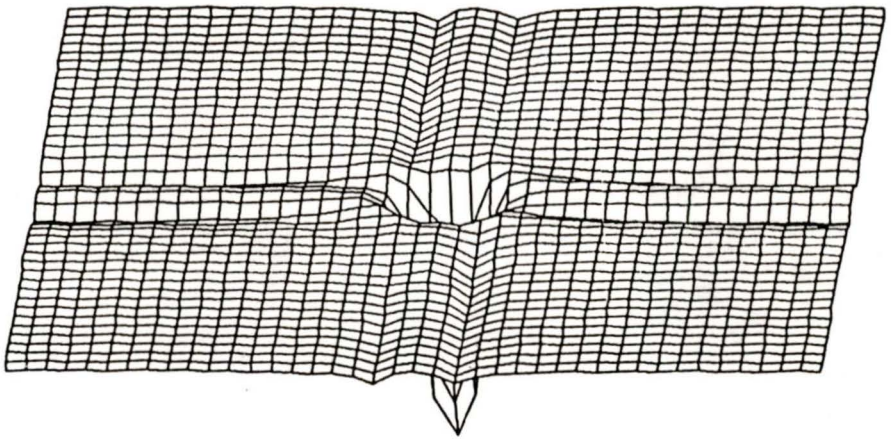
(3 x 3) Circularly symmetric Highpass - Cutoff 0.4 π radians



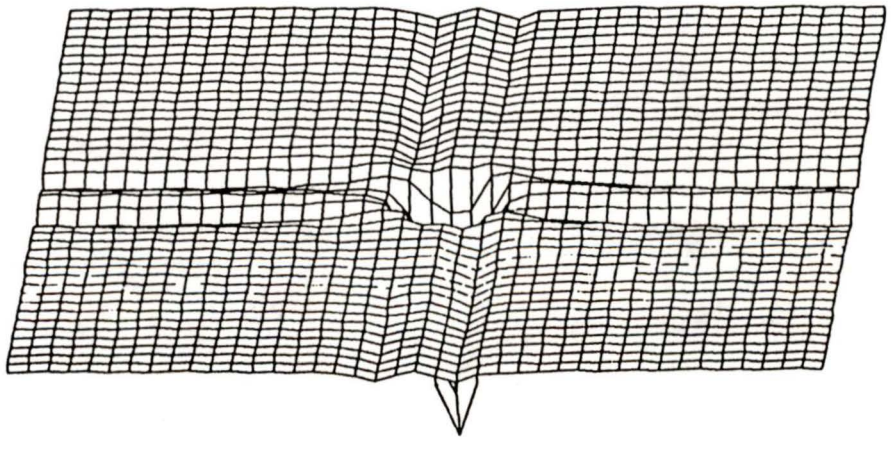
(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.07 π radians



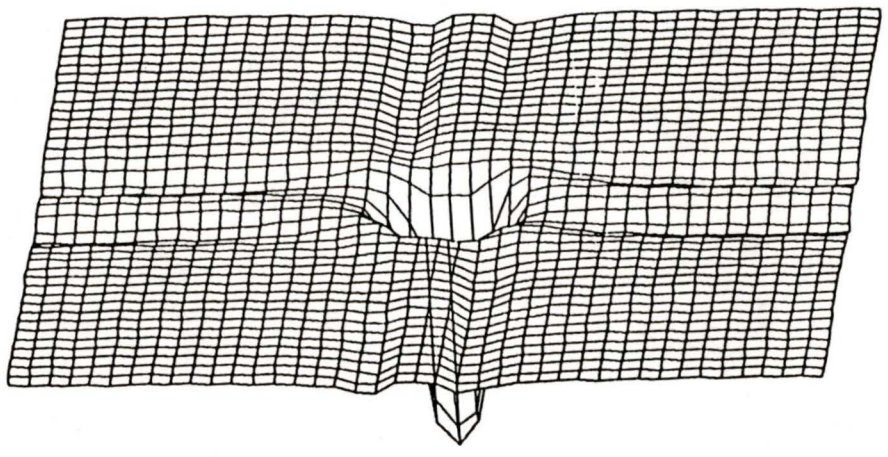
(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.08 π radians



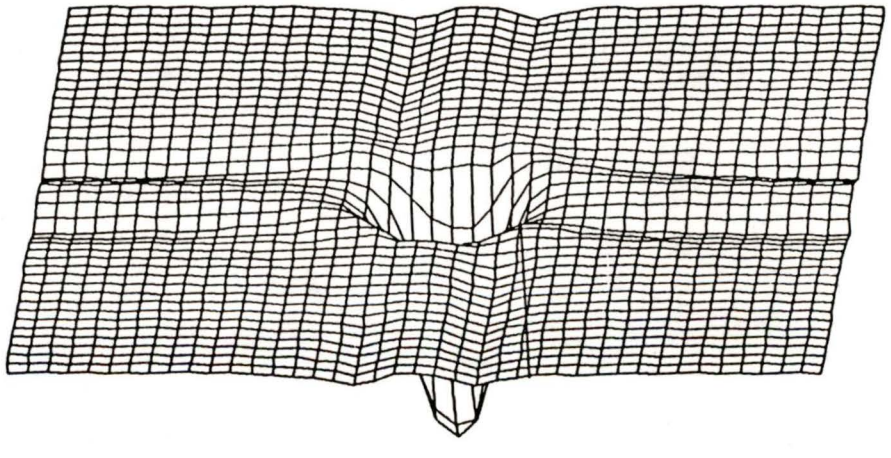
(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.09 π radians



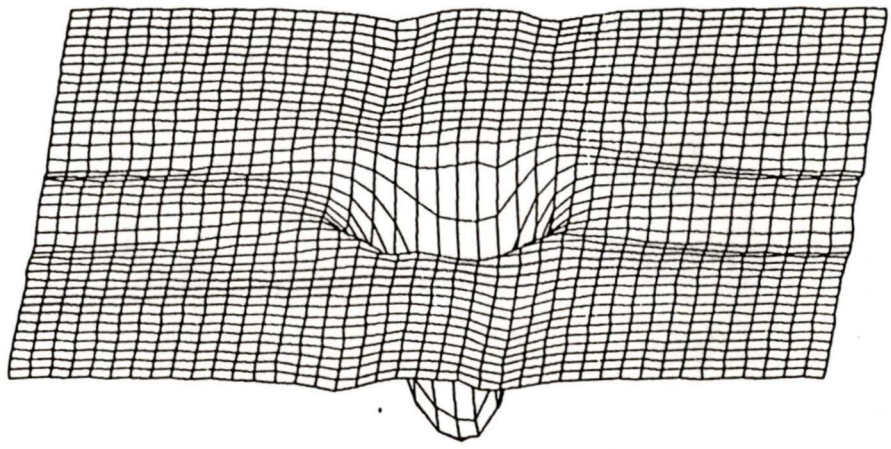
(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.1 π radians



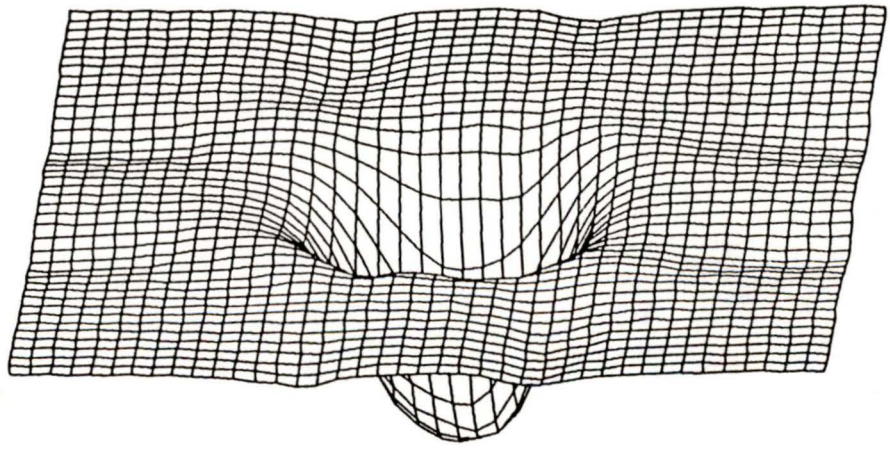
(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.125 π radians



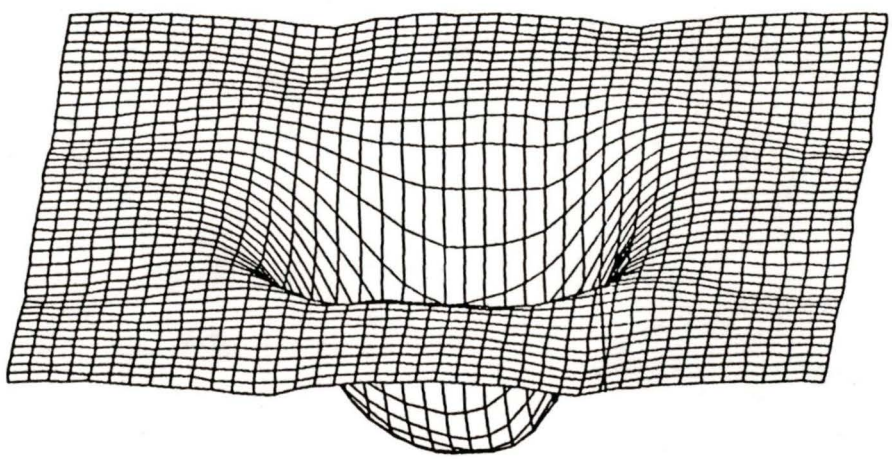
(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.15 π radians



(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.2 π radians



(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.3 π radians



(3 x 3) Circularly symmetric Image Enhancement - Cutoff 0.4 π radians

VITA

Surname: HEWITT Given Names: LAWRENCE BRUCE

Place of Birth: CALGARY, ALBERTA Date of Birth: Feb. 6, 1958

Educational Institutions Attended, with Dates of Entering and Leaving:

UNIVERSITY OF CALGARY	1978	to	1983
		to	
		to	

Degrees, Diplomas, Etc., Awarded, with Dates and Names of Institutions:

B.Sc.	1982	University of Calgary

Honors and Awards:

NSERC Scholarship, 1982/1983 and 1983/1984

Publications:

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis (the title of which is shown below) to users of the University of Victoria Library, and to make *single copies only* for such users or in the response to a request from the library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication ot this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis

THE DESIGN AND IMPLEMENTATION OF A

HIGH PERFORMANCE VIDEO ENHANCEMENT INSTRUMENT

Author [redacted]

Signature

LAWRENCE BRUCE HEWITT

Name

March 22, 1985.

Date