Addis Ababa University
School of Graduate Studies

Experimental Verification of Image Perturbation Theory as Applied to the Accuracy of Area Measurement

A Thesis Presented to the School of Graduate Studies
in (part) Fulfillment for the Degree of
Master of Science (M.Sc)
in Electrical Engineering

By

Getachew Hailu

Approval Sheet

Approved by Board Examiners:

Signatures

Dr. Girma Mullisa
Chairman, Department Graduate Committee (DGC) _____

Dr. Wolde-Ghiorgis W.Mariam
Advisor _____

Dr. -Ing. Hailu Ayele
Advisor _____

Dr.
External Examiner _____

Acknowledgements

I would like to place on record my gratitude and deep obligation to my advisers: Dr. Wolde-Ghiorgis W.Mariam, Associate Professor at the Electrical Engineering Department, and General Manager of the Ethiopian Electric Light & Power Authority (EELPA), Dr.-Ing Hailu Ayele, Dean of the Faculty of Technology and Assistant Professor at the Electrical Engineering Department, for their interest, encouragement and close supervision for the successful completion of the thesis.

I am also indebted to my friends: Eng. Shimels Kindie, a Computer Specialist at the DMF Electrocraft Company, Eng. Tefferi G. Meskel a Field Engineer at the NCR corporation, and Eng. Abebayhu Yilma a Field Engineer at IBM corporation for their material support, without which I would not have been able to complete the work in such a short period of time.

Finally, I would like to express my appreciation to W/o Misrak Tefferi, Librarian at the Graduate Record Collection (GRC), for her kind permission to borrow the many publications of the International Electrical and Electronic Engineers (IEEE) transactions on the subject: Image Perturbation Theory.

Getachew Hailu
February, 1992

Table of Contents

Abstract

An area measurement system employing a one dimensional photosensitive array, with inter-element spacing of δ = 3cm is described. A mathematical formulation of perturbation applied to length & area measurement and contour detection is reviewed and presented. A description of the proposed experimental system is given. An area measurement error is defined and applied to the projected object images. It is found that static images (no perturbation) produces considerable error. It is then found that the application of a one dimensional sinusoidal image perturbation greatly reduces the area measurement error, while triangular image perturbation is optimum and theoretically results in zero error. Some practical applications of the system are suggested.

1 Theoretical Effect of Perturbation on Measurement

1.1 Introduction

Photodiodes and phototransistors are currently being applied in many fields to convert optical signals into electrical signals [5]. One application of these devices is in the area of electronic instrumentation. The basic limitation of these devices is the binary nature of the receptor elements which cause the receptor to have non-linear characteristics. Through signal perturbation, a method in which an extra signal of higher frequency is added to the input of the receptor, the non-linearity can be minimized and the accuracy improved.

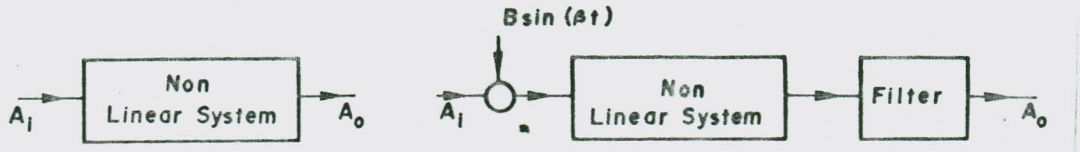
A considerable theoretical work has been done on the improvement of array system behaviour using image perturbation for measurement of position, velocity, center of mass, length, area and also for detecting image boundaries [1],[3],[4],[5]. However, very little experimental work has been reported so far [2].

The purpose of this thesis work is therefore to present experimental results which verify perturbation theory as applied to the accuracy of area measurement. The experimental system employs an x-y recorder as a scanning unit. A phototransistor array is fixed, instead of a pen, in the pen holder of the recorder. During a scanning process, the array sweeps horizontally, and for each horizontal position, the array is perturbed vertically. As the array is perturbed, the output of the phototransistors is recorded and averaged by the microcomputer. This intermediate result is the length of the line image sensed by the array for a given x-position and stored in memory. Because the measured line length ι varies with the shape of the object, strip lengths are registered serially from the time the array sense the object till the scanning operation is completed. Since a two dimensional image can be considered as an ensemble of infinite number of vertical lines, at the end of the scanning operation the microcomputer starts to compute the area using Simpson's algorithm from the length information so far gathered.

An area measurement error will be defined and will be used as index of performance, in measuring object area projected on the array. The defined performance index will be computed and compared for two modes of operations: static image, and perturbed image with different level of perturbation.

1.2 Description of Perturbation Theory

The technique of altering the input-output characteristics of a non linear system through the use of a perturbation signal has been employed for some time. A convenient theory was developed [2],[5] which described the effect of perturbation signal on non-linear systems. This effect is most easily understood in terms of the equivalent input output characteristics in the presence of perturbation signal.



(a) (b)

Figure (1.1) General non linear system and addition of perturbation signal

Figure (1.1a) shows a general non-linear device with input A_i and output A_o.

$$A_o = f(A_i) \qquad (1.1)$$

where $f(A_i)$ is a non linear function of A_i. In figure (1.1b) a perturbation signal $BSin(\beta t)$ is added to the input, and the output of the non-linear system is filtered. The frequency β is large compared to the highest frequency of A_i, and the filter is such as to greatly attenuate all frequencies greater than those contained in A_i. The output of the non-linear system can be expressed as a Fourier series.

With the assumed filter, the following equation can be used to determine A_o:

$$A_o = \frac{\beta}{2\pi} \int_0^{\frac{2\pi}{\beta}} f(A_i + BSin(\beta t))\, dt \qquad (1.2)$$

Here A_i can be assumed constant over the period of $2\pi/\beta$, since β is large compared to frequencies contained in A_i. Equation (1.2) can be used for computing the altered input and output characteristics simply by letting A_i to vary through its range and computing A_o for each values of A_i.

1.3 Perturbation Theory Applied to Measurement of Length and Area

1.3.1 Measurement of Length

Figure (1.2) shows the system used to measure line length using one dimensional receptor array. Only one row of array element is shown for simplicity. The output of each element is quantized by threshold gate to "1" or "0", before going to the summing block. The dimension of the single receptor element is assumed to be m x m. In the analysis that follows, the active area of each element is assumed to be much smaller than the dead space (element separation) δ, and the array is treated as a matrix of light sensitive points. Figure (1.2) shows a line image of length ι coincident with an element row of infinite extent. The left end of the line lies beyond the element designated as the origin by the amount $a\delta$, where $0 \leq a \leq 1$. With no image perturbation the output of the system is shown in figure (1.3).

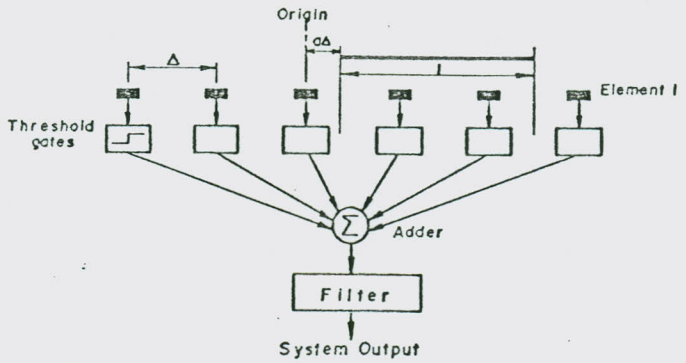


Figure (1.2) Length measurement system

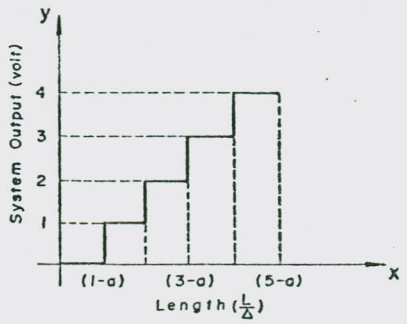


Fig.(1.3) Length measurement system output with no image perturbation

Now let the line be perturbed in a random manner about its steady state position. The position of the left end point is $a\delta + x(t)$, where $x(t)$ is the value of the perturbing signal, and it is a random variable. The system output is the average of the sum of the quantized element output. The average output of an element is directly proportional to the probability that the line covers the element. Hence for the origin element in figure (1.2), we have,

$$p_o = \int_{-a\delta-\iota}^{-a\delta} f(x)\,dx \tag{1.3}$$

where p_o is the probability that the line covers the origin element and f(x) is the probability density function of the perturbing signal. For element i we have,

$$p_i = \int_{(i-a)\delta-\iota}^{(i-a)\delta} f(x)\,dx \tag{1.4}$$

If the output of each threshold gate is 1 v, the system output in volt is,

$$\begin{aligned} S(\iota, a, f, \delta) &= \sum_i p_i \\ &= \sum_{i=-\infty}^{\infty} \left(\int_{(i-a)\delta-\iota}^{(i-a)\delta} f(x)\,dx \right) \end{aligned} \tag{1.5}$$

In general, for a given line length the system output will depend on δ, f(x) and a. However, for $\iota = n\delta$, where n is a positive integer, it will be shown that the system output is independent of a and f(x). For $\iota = n\delta$, we have from equation (1.5),

$$S(n\delta, a, f, \delta) = \sum_{i=-\infty}^{\infty} \left(\int_{(i-a-n)\delta}^{(i-a)\delta} f(x)\,dx \right) \tag{1.6}$$

Expanding equation (1.6) for i = 0, 1, and 2 yields,

$$\begin{aligned} S(n\delta, a, f, \delta) = &\int_{(-a-n)\delta}^{(1-a-n)\delta} f(x)\,dx + \int_{(a-a-n)\delta}^{(2-a-n)\delta} f(x)\,dx + \ldots + \int_{(-1-a)\delta}^{-a\delta} f(x)\,dx + \\ &\int_{(1-a-n)\delta}^{(2-a-n)\delta} f(x)\,dx + \int_{(2-a-n)\delta}^{(3-a-n)\delta} f(x)\,dx + \ldots + \int_{(-a\delta)}^{(1-a)\delta} f(x)\,dx + \\ &\int_{(2-a-n)\delta}^{(3-a-n)\delta} f(x)\,dx + \int_{(3-a-n)\delta}^{(4-a-n)\delta} f(x)\,dx + \ldots + \int_{(1-a)\delta}^{(2-a)\delta} f(x)\,dx \end{aligned} \tag{1.7}$$

In equation (1.7) the integral for each value of i has been written as a sum of n integrals. It can be seen that in taking the sum for all i, equation (1.7) contains n columns of each with an infinite limit of integration. The sum of all the integrals in each column, however is just,

$$\int_{-\infty}^{\infty} f(x)\,dx \tag{1.8}$$

If f(x) is the probability density function then equation (1.8) equals to 1, and we have,

$$S(n\delta) = n \tag{1.9}$$

Equation (1.9) indicates that the curve of S versus ι, for fixed f(x) passes through points $\iota = i\delta$, S = i. To see the behaviour of S between values $\iota = i\delta$ and $\iota = (i+1)\delta$ requires that f(x) be specified. A zero mean gaussian distribution was chosen for illustrative purpose. The mean of the random process has no consequence in this application. The normal density function is,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \tag{1.10}$$

Inserting equation (1.10) into equation (1.5), we get,

$$S(\iota, a, f, \delta) = \sum_{i=-\infty}^{\infty}\left(\int_{(i-a)\delta-\iota}^{(i-a)\delta} \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} dx\right) \tag{1.11}$$

Ignoring computation when the limit of the integral exceeds 4σ, and normalizing line length ι and σ with respect to the connector separation δ,

$$\iota_n = \frac{\iota}{\delta} \quad ; \quad \sigma_n = \frac{\sigma}{\delta} \tag{1.12}$$

we arrive, after some simplification (see appendix A) at,

$$S(\iota_n, a, \sigma_n, \delta) = \sum_{i=i_{min}}^{i_{max}}\left(\int_{\frac{i-a-\iota_n}{\sigma_n}}^{\frac{i-a}{\sigma_n}} \frac{1}{\sqrt{2\pi}}\, e^{-\frac{1}{2}x^2} dx\right) \tag{1.13}$$

where,

$$i_{min} = \lfloor a-4\sigma_n \rfloor \; ; \; i_{max} = \lceil a+4\sigma_n+\iota_n \rceil \qquad (1.14)$$

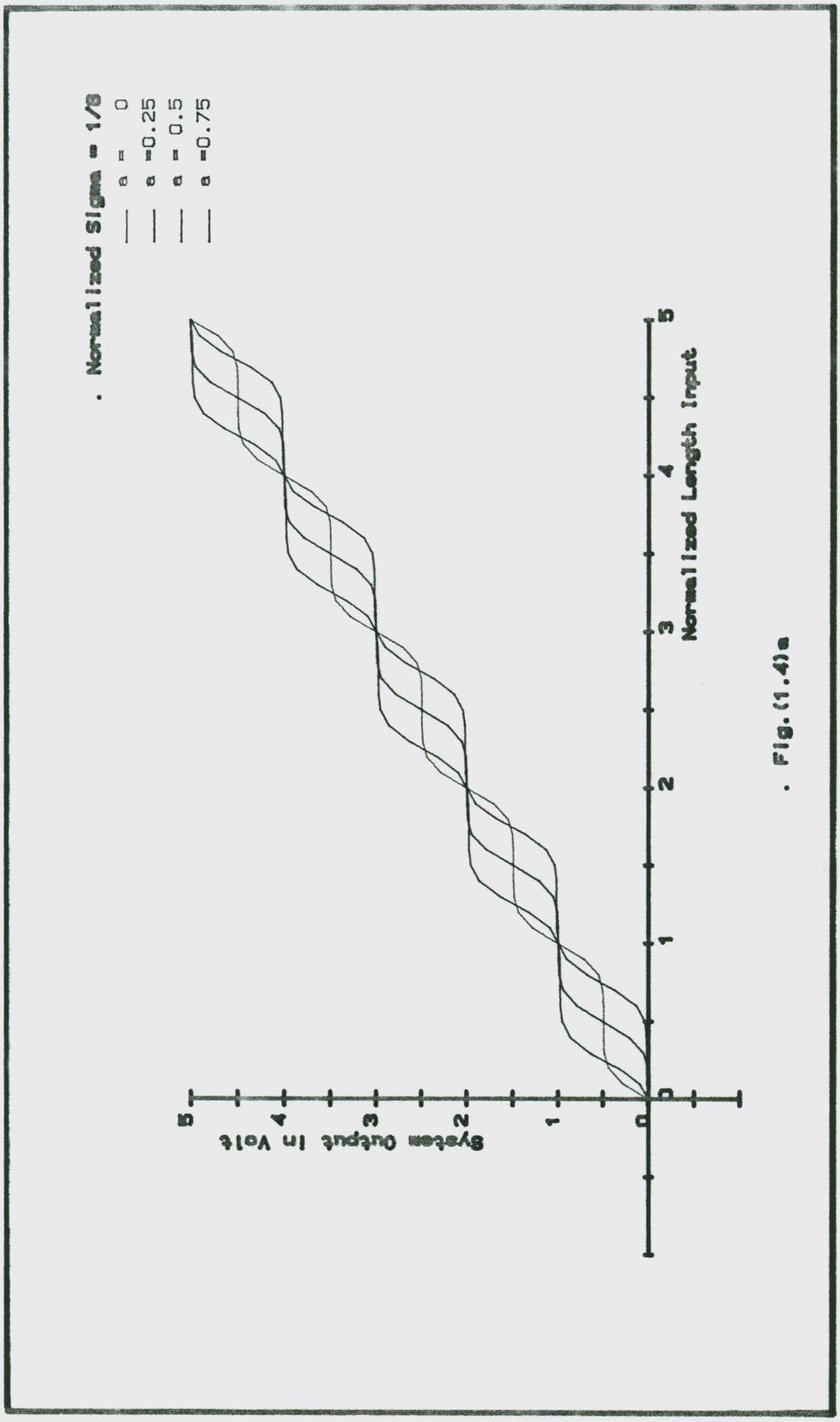
and

$\lfloor x \rfloor$ is the smallest integer greater than or equal to x

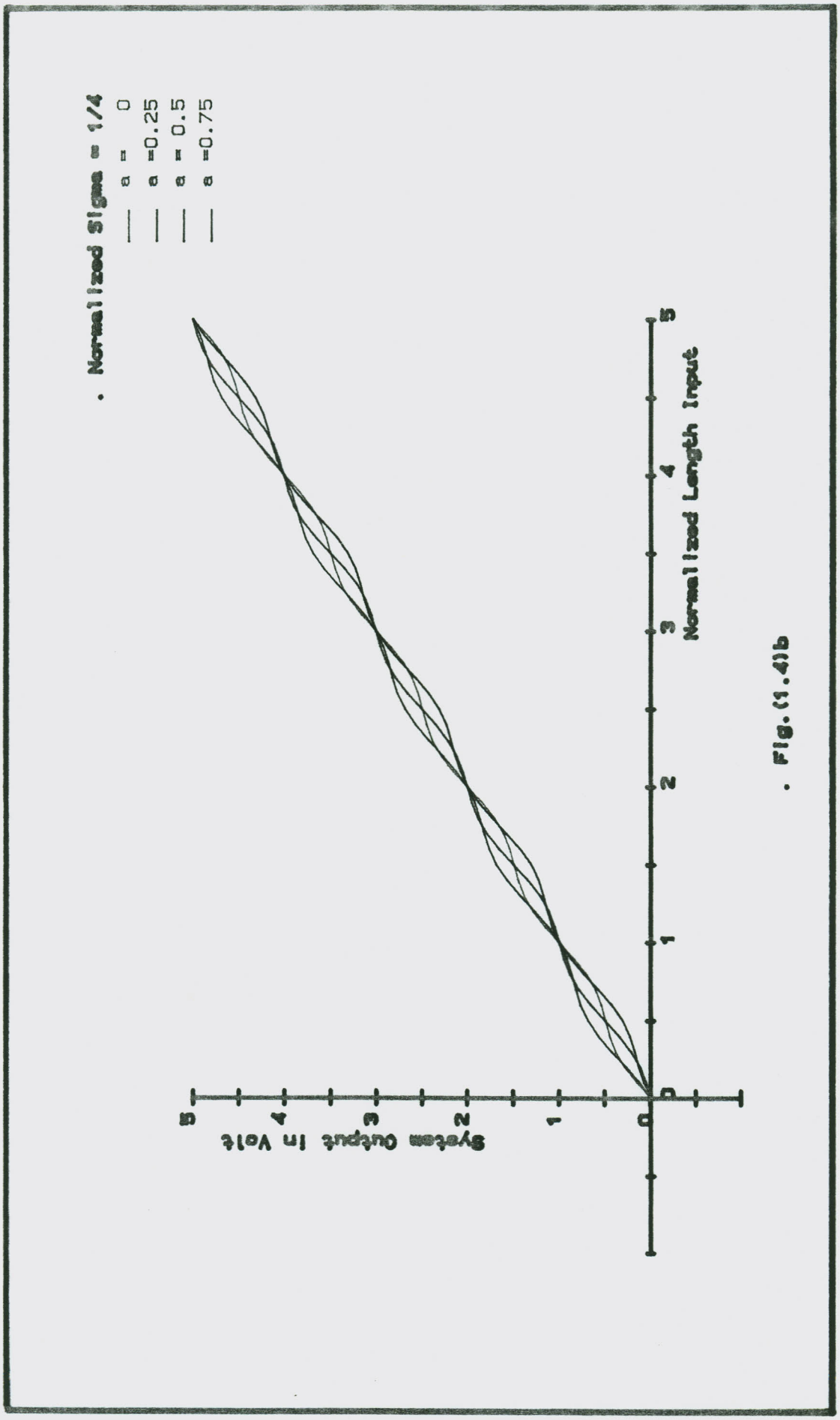
$\lceil x \rceil$ is the largest integer less than or equal to x

Equation (1.13) was used in a computer program (see appendix B) to evaluate system output S versus ι_n with a and σ_n as parameters. Figures (1.4a) - (1.4c) show length measurement system output versus line length for values of a = 0, ¼, ½, ¾ and for three values of σ_n. For the smallest value of σ_n, the curve approximate the stair case function obtained with no perturbation. As σ_n increases, the curve degenerate into a straight line that is achieved for all practical purposes for $\sigma_n > ½$. For σ_n greater than ½, it can be concluded that length measurement is essentially independent of image position on the array and a nearly linear relation exist between length input and system output. In this case, the output of the system can be approximated by,

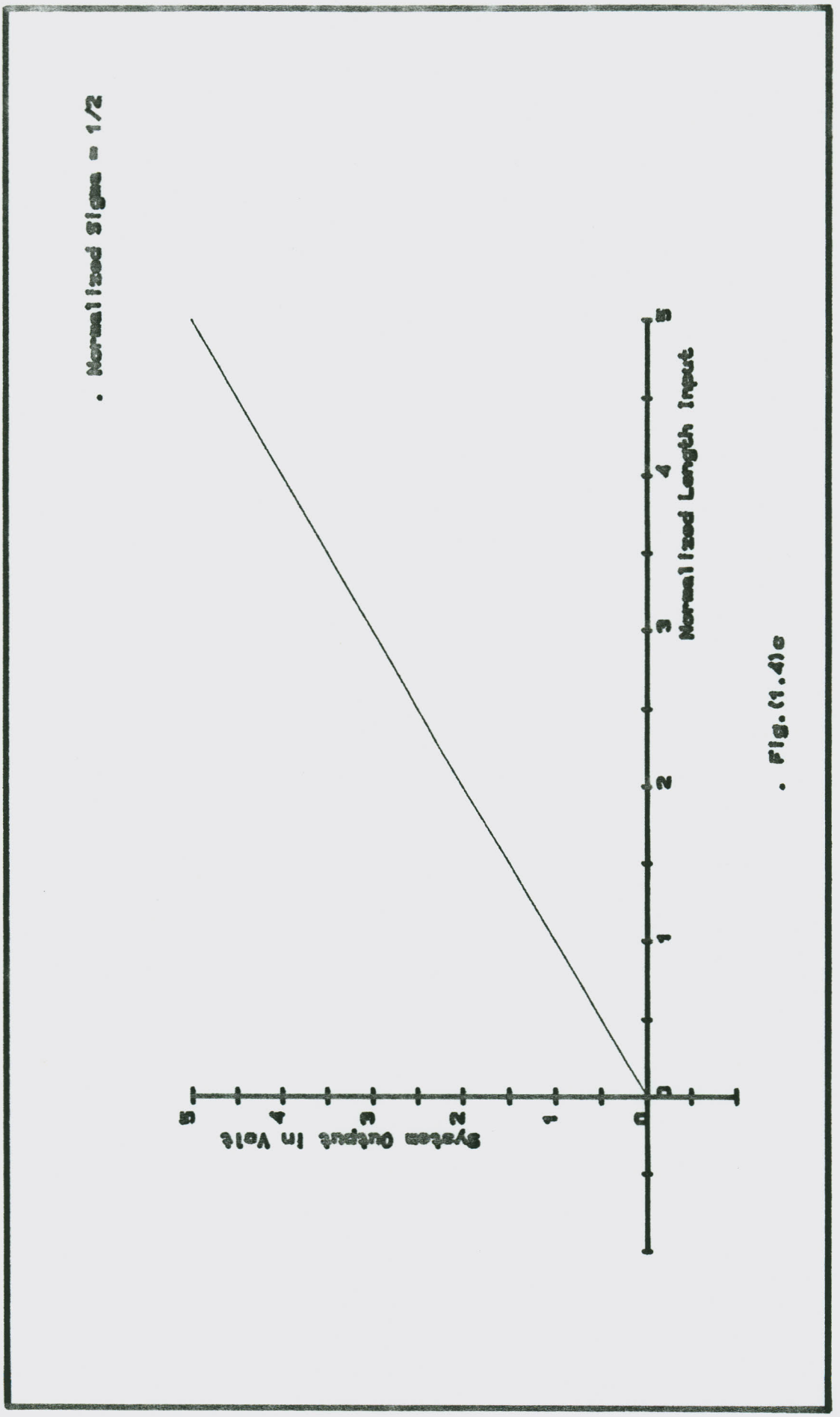
$$S(\iota) = \iota_n = \frac{\iota}{\delta} \qquad (1.15)$$


. Normalized Sigma = 1/8
a = 0
a =0.25
a = 0.5
a =0.75
System Output in Volt
Normalized Length Input
0
1
2
3
4
5

. Fig.(1.4)a


. Normalized Sigma = 1/4
a = 0
a =0.25
a = 0.5
a =0.75
System Output In Volt
Normalized Length Input
0
1
2
3
4
5

. Fig.(1.4)b


. Normalized Sigma = 1/2
System Output in Volt
Normalized Length Input
0
1
2
3
4
5

. Fig.(1.4)e

1.3.2 Measurement of Area

Figure (1.5) shows a rectangular coordinate system superimposed on matrix of elements. An arbitrary area A is shown projected on the matrix. Area measurement is facilitated with the system in figure (1.5) by randomly perturbing the image in the x and y direction simultaneously. In the analysis that follows $f_1(x)$ is the probability density function of the perturbing signal in the x-direction and $f_2(y)$ is the probability density function in the y-direction. It is assumed that the x and y perturbations are independent, so that the probability of the composite perturbation signal lying in the range x, x+dx and y and y+dy is $f_1(x)f_2(y)dxdy$. The analysis is kept general by assuming that the image is bounded by two single valued functions of x, $g_1(x)$ and $g_2(x)$. The extremities of the image in the x-direction are x_1 and x_2 as shown in the figure.

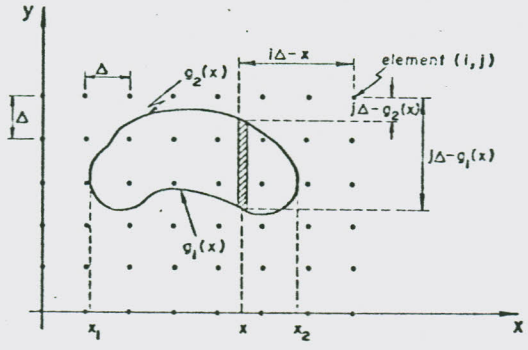


Figure (1.5) Square array of elements

The same approach used in the preceding section for length measurement is used here to determine the output of the area measurement system. The average output of element i,j is directly proportional to the probability $p_{i,j}$ that the image covers the element and the system output is the sum of all the probabilities. The development is started by deriving the probability that a general element is covered. Referring to figure (1.5), the probability is determined by finding that a different strip of image covers the element and then by integrating over the entire area. The probability that the strip covers element i,j is given by,

$$f_1(i\delta - x)\int_{j\delta - g_2(x)}^{j\delta - g_1(x)} f_2(y)\,dy \qquad (1.16)$$

The probability $p_{i,j}$ is found by integrating the above expression between the limit x_1 and x_2 as,

$$P_{i,j} = \int_{x_1}^{x_2} f_1(j\delta - x)\left(\int_{j\delta - g_2(x)}^{j\delta - g_1(x)} f_2(y)\,dy\right)dx \qquad (1.17)$$

The system output is the sum of the individual element output, that is,

$$S(A) = \sum p_{i,j}$$

$$= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \int_{x_1}^{x_2} f_1(i\delta - x) \left(\int_{j\delta - g_2(x)}^{j\delta - g_1(x)} f_2(y)\, dy \right) dx \qquad (1.18)$$

Rearranging equation (1.18) yields,

$$S(A) = \sum_{i=-\infty}^{\infty} \int_{x_1}^{x_2} f_1(i\delta - x) \left(\sum_{j=-\infty}^{\infty} \int_{j\delta - g_2(x)}^{j\delta - g_1(x)} f_2(y)\, dy \right) dx \qquad (1.19)$$

The expression within the bracket in equation (1.19) is equivalent to the right hand side of equation (1.5) with $a\delta = g_1(x)$ and $a\delta + \iota = g_2(x)$. Hence, if $f_2(y)$ is a Gaussian density function with $\sigma_y = \frac{1}{2}\delta$, then the expression within the bracket can be approximated by the line length,

$$\frac{1}{\delta} (g_2(x) - g_1(x)) \qquad (1.20)$$

making the substitution yields,

$$S(A) = \frac{1}{\delta} \sum_{i=-\infty}^{\infty} \left(\int_{x_1}^{x_2} f_1(i\delta - x)\, (g_2(x) - g_1(x))\, dx \right) \qquad (1.21)$$

It remains to show that equation (1.21) is directly proportional to the image area which is,

$$A = \int_{x_1}^{x_2} (g_2(x) - g_1(x))\, dx \qquad (1.22)$$

To test the system response, a circular image is selected. For a circular image, equation (1.21) reduces to (see appendix C),

$$S(A) = \sum_{i_{min}}^{i_{max}} \left(\int_{c-r_n}^{c+r_n} 2\sigma_n \sigma f_1(i\delta - \sigma x) \left(\sqrt{r_n^2 - (x-c)^2} \right) dx \right) \qquad (1.23)$$

where,

$$f_1(i\delta-\sigma x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{i\delta-\sigma x}{\sigma}\right)^2} \qquad (1.24)$$

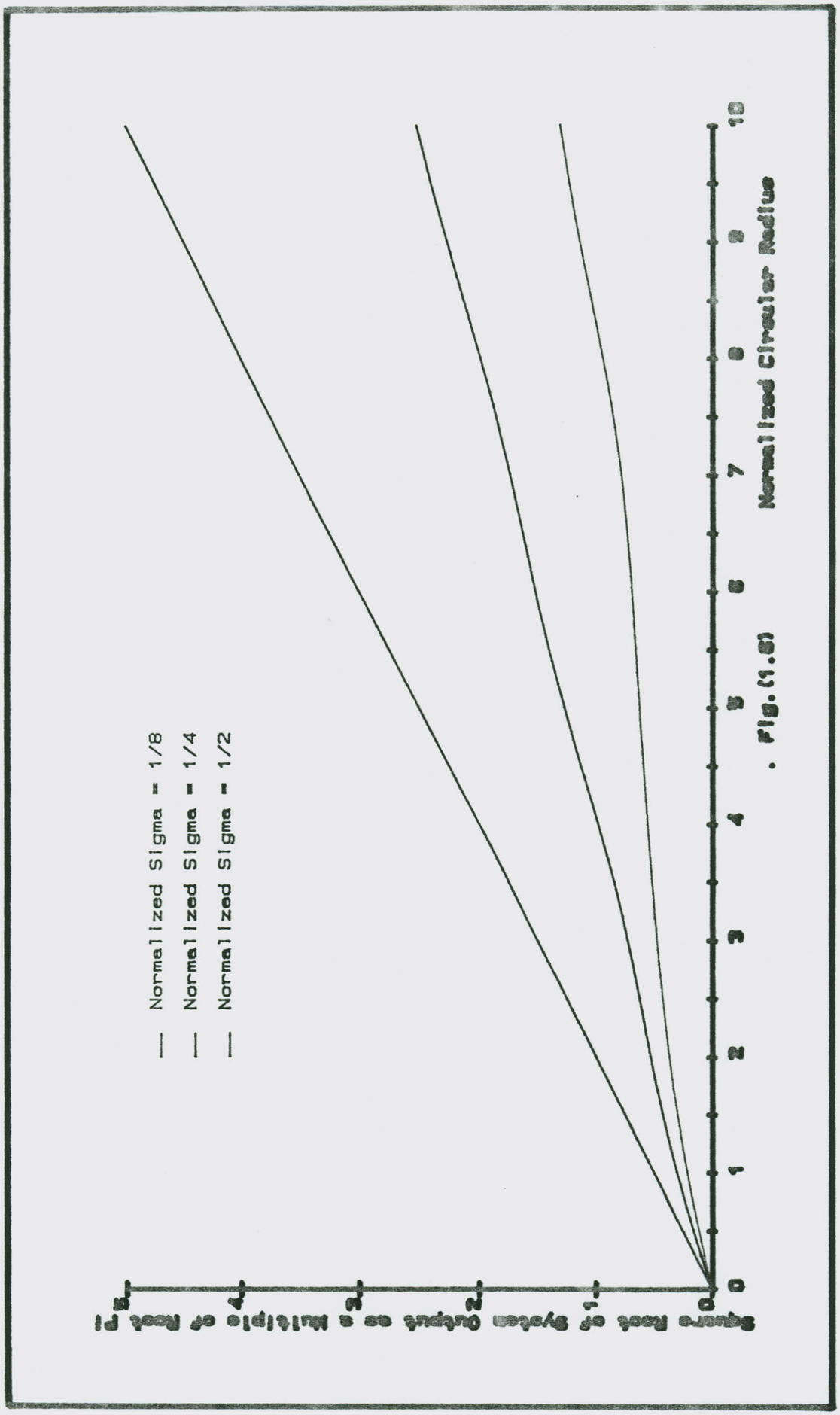
and

$$i_{min} = \lfloor \sigma_n(c-r_n-4) \rfloor \quad ; \quad i_{max} = \lceil \sigma_n(c+r_n+4) \rceil \qquad (1.25)$$

Equation (1.23) was used in a computer program (see appendix D), to evaluate the system response S(A) for three values of σ_n in the x-direction. The circle center is coincident with an element although it was found that for $\sigma_n \geq \frac{1}{2}$ system output was essentially independent of the placement of the circle.

Figure (1.6) is a plot of the square root of the system output versus the radius of the circle r. Again computation was terminated when the limit exceeds 4σ. Just as the length measurement case $\sigma_n \geq \frac{1}{2}$ results in a nearly linear output. For this range of σ_n and with normal perturbation the system output can be closely approximated by,

$$S(A) = \frac{A}{\delta^2} \qquad (1.26)$$


Normalized Sigma = 1/8
Normalized Sigma = 1/4
Normalized Sigma = 1/2
Square Root of System Output as a Multiple of Root PI
0
1
2
3
4
5
6
7
8
9
10
Normalized Circular Radius

. Fig. (1.6)

1.4 Perturbation Theory Applied to Image Boundary Detection

1.4.1 System Description

Image perturbation has also application in the improvement of the accuracy of image shape determination [1],[7]. A linear sensor array system is used to detect contour of various two dimensional objects of uniform intensity. The system consists of a linear array, a signal processor (Microcomputer) and an output indicator (x-y plotter) as shown in figure (1.7). The array system converts the optical energy of the object image into an electrical video signal. The video signal is integrated and then quantized with one bit, before being processed by the microcomputer to yield the contour information of the image. This information is then plotted by x-y plotter.

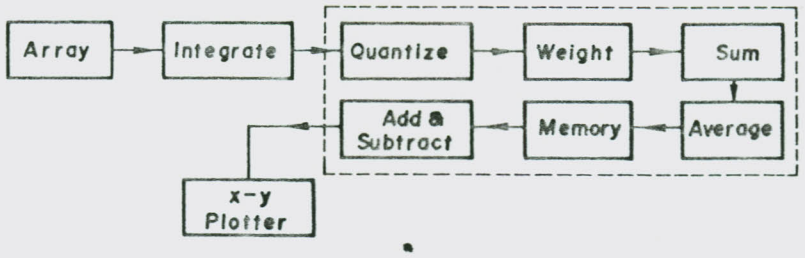


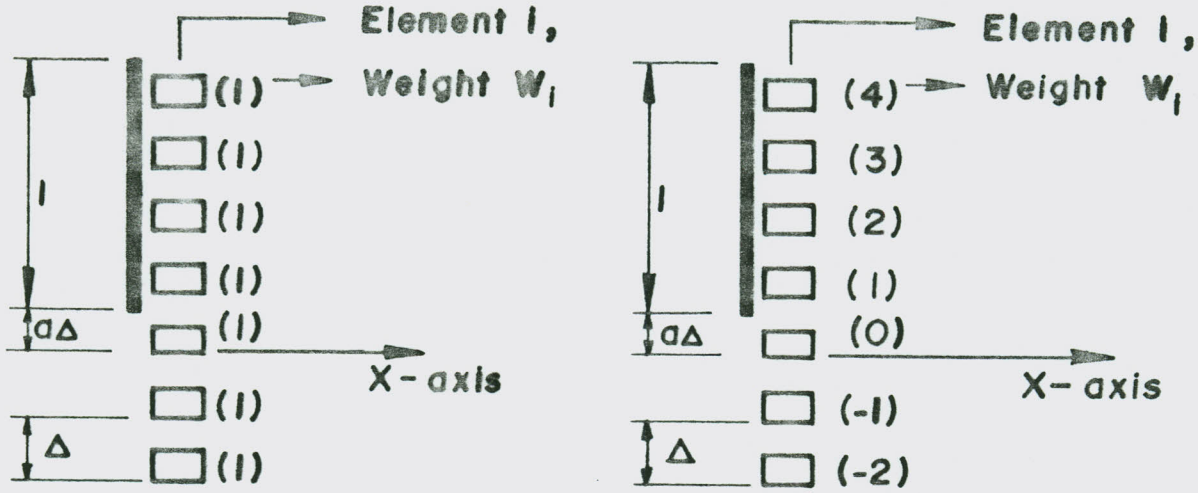
Figure (1.7) System block diagram

The coordinate determination for each point on the contour of a given two dimensional image is accomplished in the following way. The quantized signals are first weighted "1" as shown in figure (1.8a), the signals are then time averaged by the microcomputer. This intermediate result is then the length of the line image sensed by the array and is saved. The weight of each element is then set according to its y-coordinate as shown in figure (1.8b). The change of weight from figure (1.8a) to figure (1.8b) can be done automatically through software. The signals are again averaged by the microcomputer. The result is the moment of the line about the x-axis. This is divided by the stored line length to obtain the y-coordinate of the centroid. Knowing the centroid and the length of the line image, the y-coordinate for both ends of the line can be calculated and plotted against the given x-position. The entire contour is plotted when the linear array is swept in x-direction completely across the image plane.

1.4.2 Contour Generation with a Linear Array

Assuming that the active area of each element in the linear array is much smaller than the element separation δ, and the array is treated as light sensitive points, the quantization of the object image by the array introduce error. The error is a function of image shape, complexity, and location of image pattern with respect to the array element matrix. In order to reduce this quantization error, random image perturbation is used.

Figure (1.8) shows a line of image to be detected coincident with the array elements which are assumed to be infinitely long. The length of the line is assumed to be ι. The bottom end of the line lies above the element designated as the origin by the amount $a\delta$ where $0 \leq a \leq 1$. Now let the line be perturbed in a random manner in the y direction about its steady state position. The position of the bottom end is $a\delta$ + y(t) where y(t) is the value of the perturbing signal and is a random variable with respect to time t. The average output of an array element is directly proportional to the probability that the line covers the element, the output of each array element is purely weighted; hence, the probability that the line covers the array element i multiplied by W_i is,



(a) (b)

Figure (1.8) Weight arrangement for
(a) Length measurement
(b) Moment measurement

$$M_i = \int_{(i-a)\delta-\iota}^{(i-a)\delta} W_i \, f(y)\, dy \qquad (1.27)$$

Where i and W_i are integers or zeros, and the random perturbation function is assumed to be gaussian with zero mean as,

$$f(y) = \frac{1}{\sigma\sqrt{2\pi}} \, e^{-\frac{1}{2}\left(\frac{y}{\sigma}\right)^2} \qquad (1.28)$$

The system output M, which is the moment of the line about the x-axis is the summation of the response from all array elements.

$$M = \sum_{i=-\infty}^{\infty} M_i$$
$$= \sum_{i=-\infty}^{\infty} \left(\int_{(i-a)\delta-\iota}^{(i-a)\delta} W_i \, f(y)\, dy \right) \qquad (1.29)$$

If the output of each element is weighted "1" as shown in figure (1.8a), the system output indicates the length of the line ι and reduces to,

$$\iota = \sum_{i=-\infty}^{\infty} \left(\int_{(i-a)\delta-\iota}^{(i-a)\delta} f(y)\, dy \right) \qquad (1.30)$$

Knowing ι and M, the centroid of the line can be obtained as,

$$\bar{y} = \frac{M}{\iota} \qquad (1.31)$$

The y-coordinate of the top end of the line is,

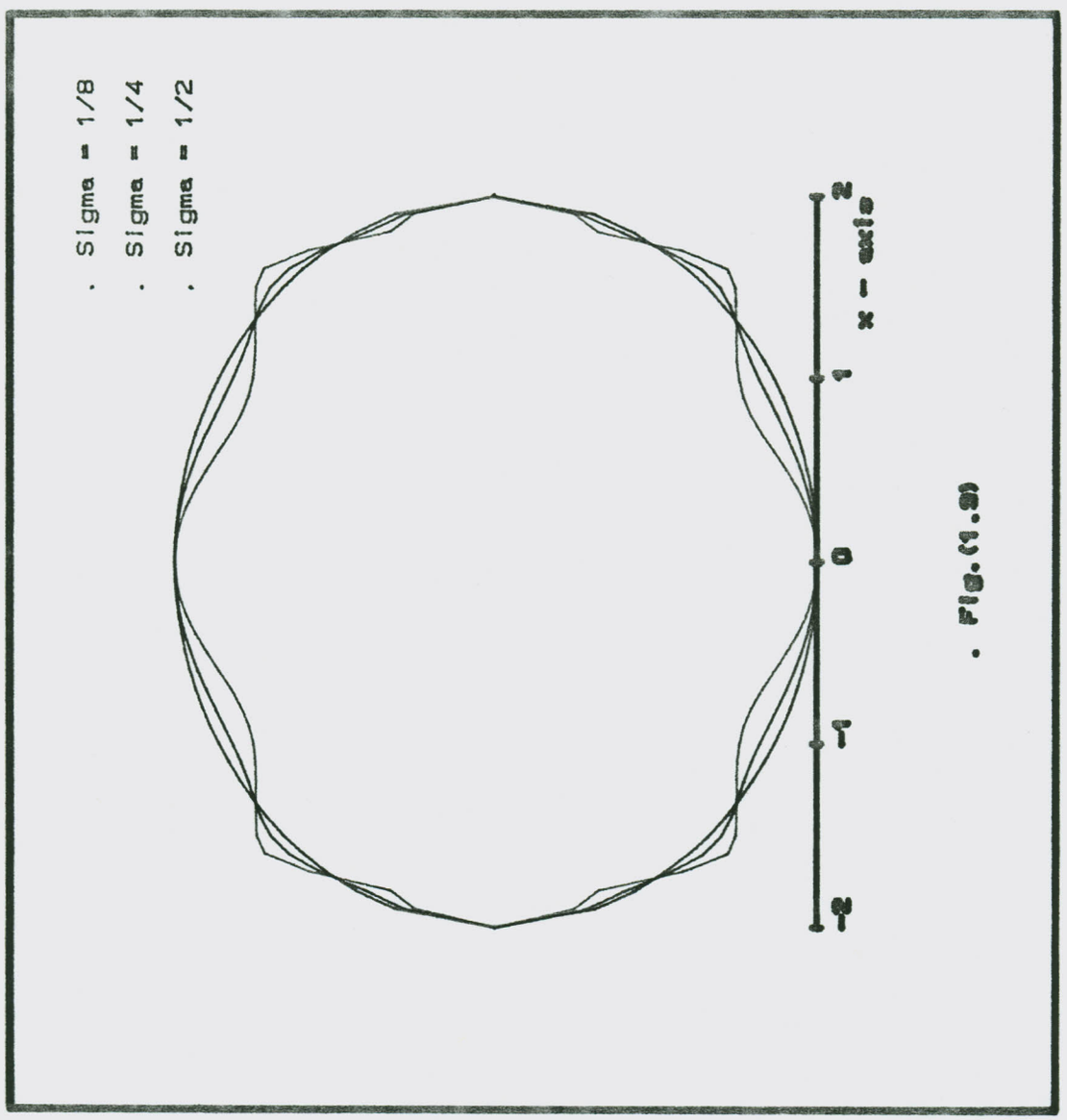
$$y_1 = \bar{y} + \frac{\iota}{2} \qquad (1.32)$$

and that of the bottom,

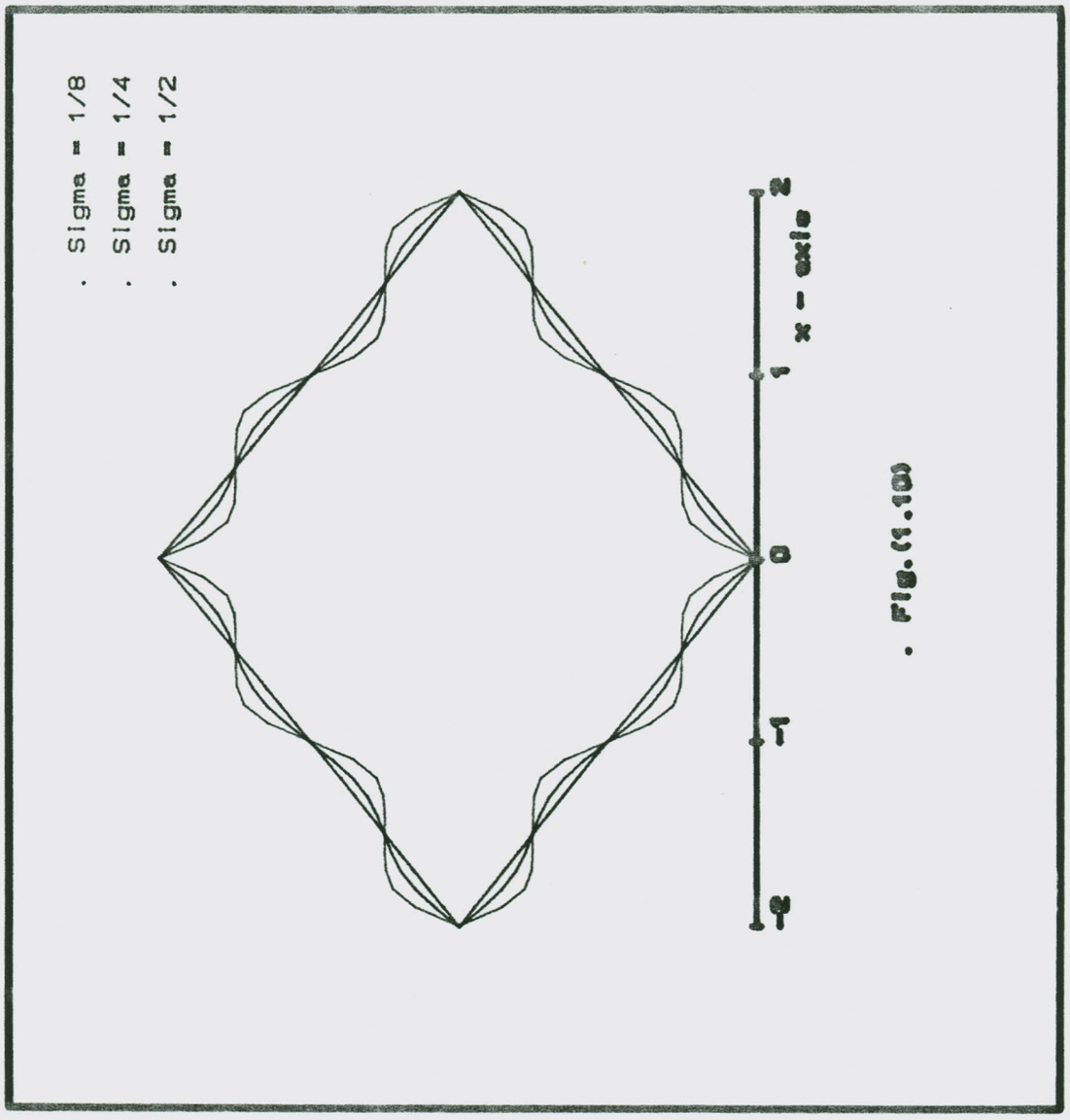
$$y_2 = \bar{y} - \frac{\iota}{2} \qquad (1.33)$$

As the array element sweeps across in the x-direction, a set of y_1 and y_2 are generated with respect to the given x - coordinate. These values are then plotted on an x-y plotter to constitute the contour of the two dimensional image under detection.

In order to test the accuracy of the linear array system, some simple image shapes were selected as examples. They were a circle and a diamond. Contour for each image shape were generated using equations (1.27) through (1.33) and plotted in figures (1.9) and (1.10) for three values of σ, where σ is the standard deviation of the random signal (see appendix E for program listing.) It is seen from all the examples shown that increasing σ results in better detection accuracy. Almost perfect contour detection was obtained for the two geometric bodies considered when σ was set equal to one half of the array element separation δ.



. Fig.(1.9)



. Fig.(1.10)

2 System Description

2.1 Hardware Design and Construction

The complete area measurement using a microcomputer as a controller is shown schematically, in term of functional modules, in figure (2.1). The functional modules can be categorized into three major sections. These are:

- signal processor (Microcomputer) section
- transducer and interface section, &
- perturbation section.

Since an almost continuous scanning of the object by an x-y recorder is adopted in the design system, instead of a matrix of phototransistor array, the system designed is particularly suitable for the measurement of small area. The principle of operation and detailed system description, along with the interfacing circuit are discussed below.

2.1.1 Signal Processor (Microcomputer) Section

The main function of the signal processor (Microcomputer) system is to receive the electrical signal from each receptor element and to sum and pass it through a digital low pass filter so that only the average value remains. The microcomputer also selects the type of wave form, triangular or sinusoidal, that is used as a perturbing signal. The signal processor system consists of three circuit boards. Figure (2.2a) - (2.2c) show the schematic representation of the microcomputer system. The Central Processing Unit (CPU) is Z80B microprocessor. Although other types of microprocessor can accomplish the same task, the Z80B microprocessor was selected mainly because of the familiarities with the chip and its programming language. Also, the cross assembler for the Z80 was available. The CPU board which has a capacity of addressing up to 64k words of memory and 256 input/output ports was constructed with EPROM and RAM board and a few supporting components. It is operated at 5.0MHz clock frequency using a 10.0MHz crystal oscillator. The EPROM provides 16K bytes erasable and programmable read only memory and the RAM also provides 16K bytes random access memory (RAM). The printer adapter board contains two output and one input ports. The allocation of the input/output port for the various functions are as follows,

a) Port address 00 is used to input the four bits status lines of the printer. These are: BUSY, ACKNOWLEDGE, PAPER OUT, and ERROR.
b) Port address 01 is used to output the data to the printer.
c) Port address 02 is used to send the strobe signal to the printer.

The microcomputer system was assembled according to the above description after it is designed and tested using the RS 488-618 laboratory prototype board.

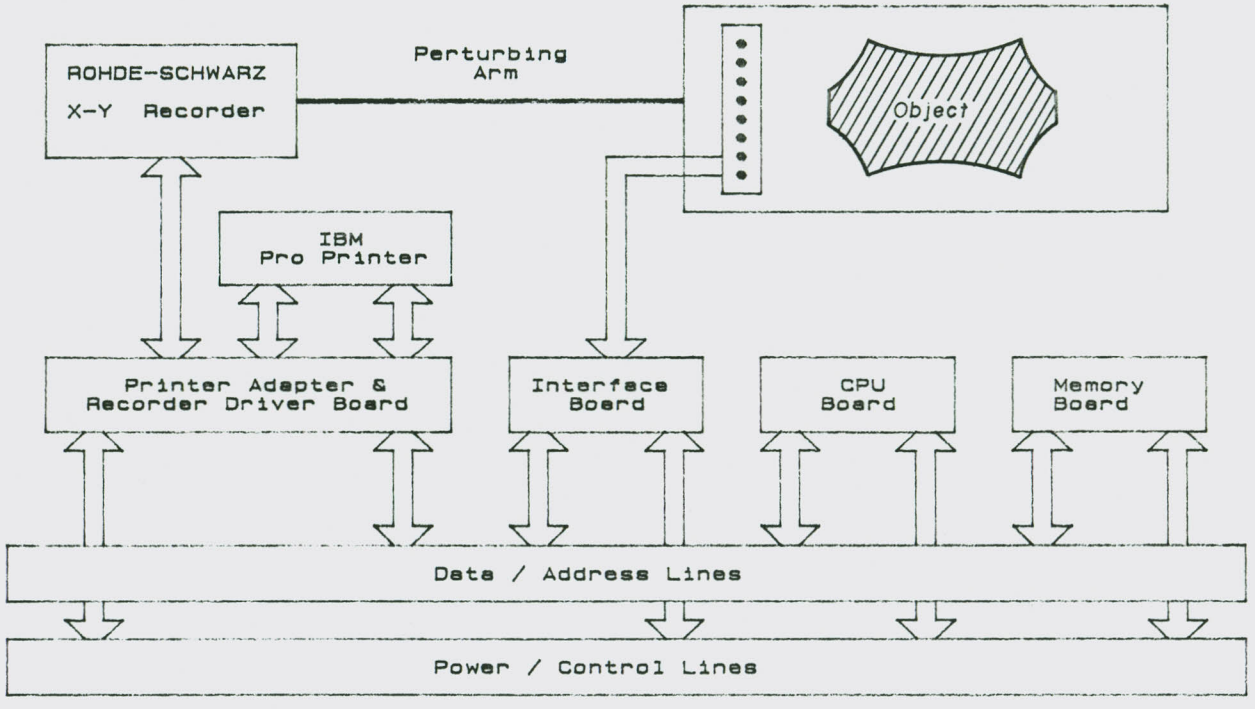


Fig. (2.1) Schematic Representation of Area Measurement System with Microcomputer as a Controller

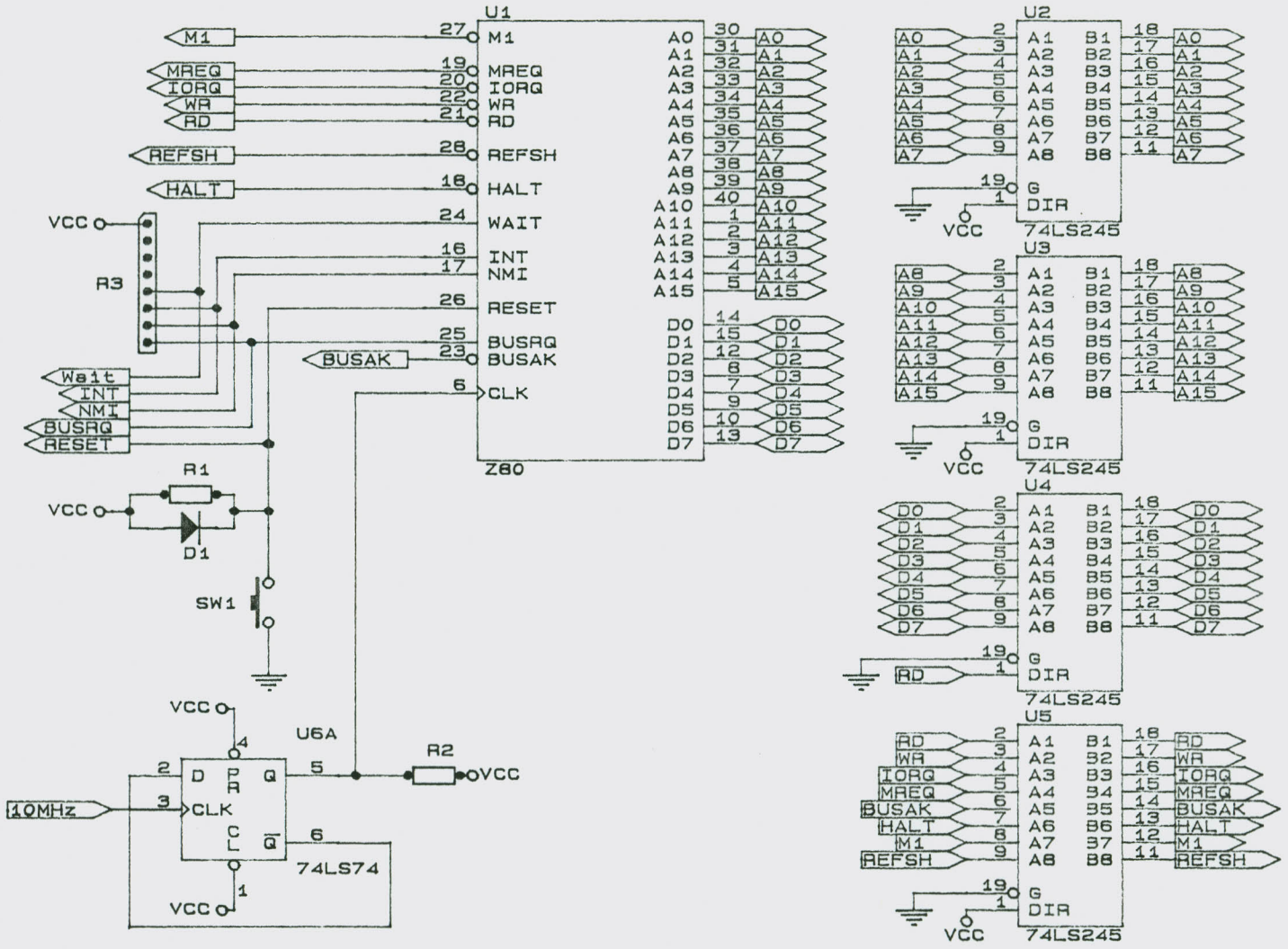


Fig. (2.2a) The Central Processing Unit (CPU) Board

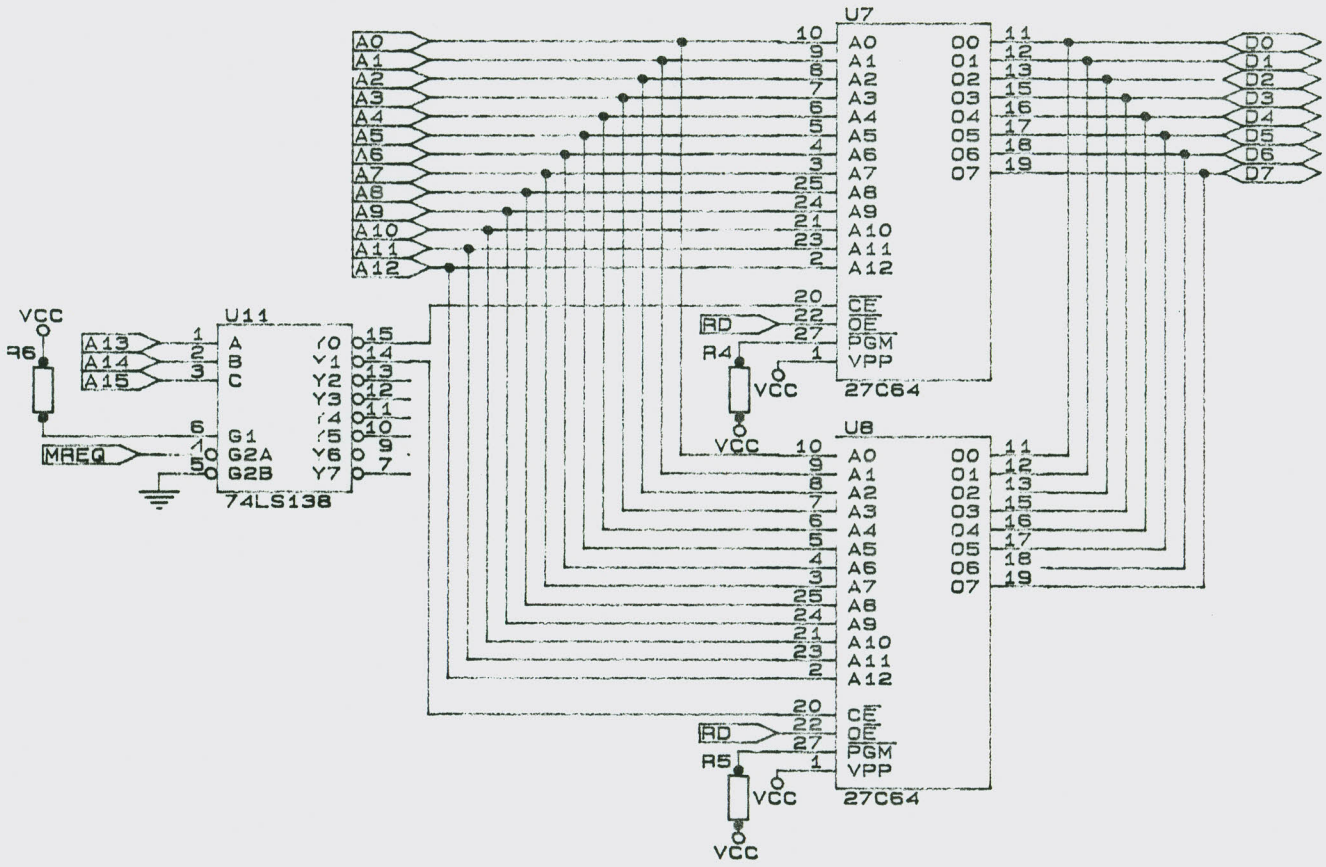


Fig. (2.2b) Schematic Diagram of the Memory Board (EPROM) only

... cont'd

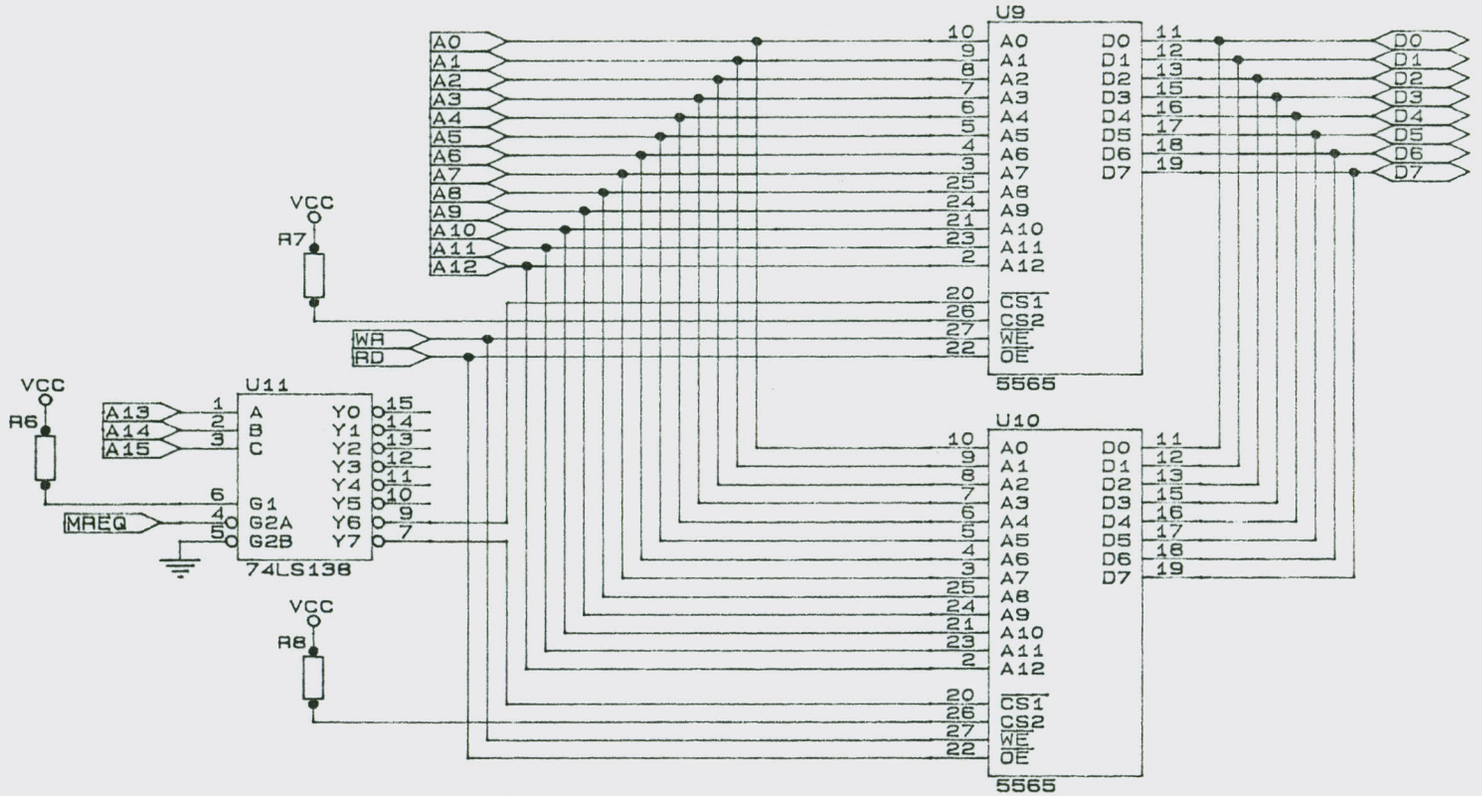


Fig. (2.2b) Schematic Diagram of the Memory Board (RAM only)

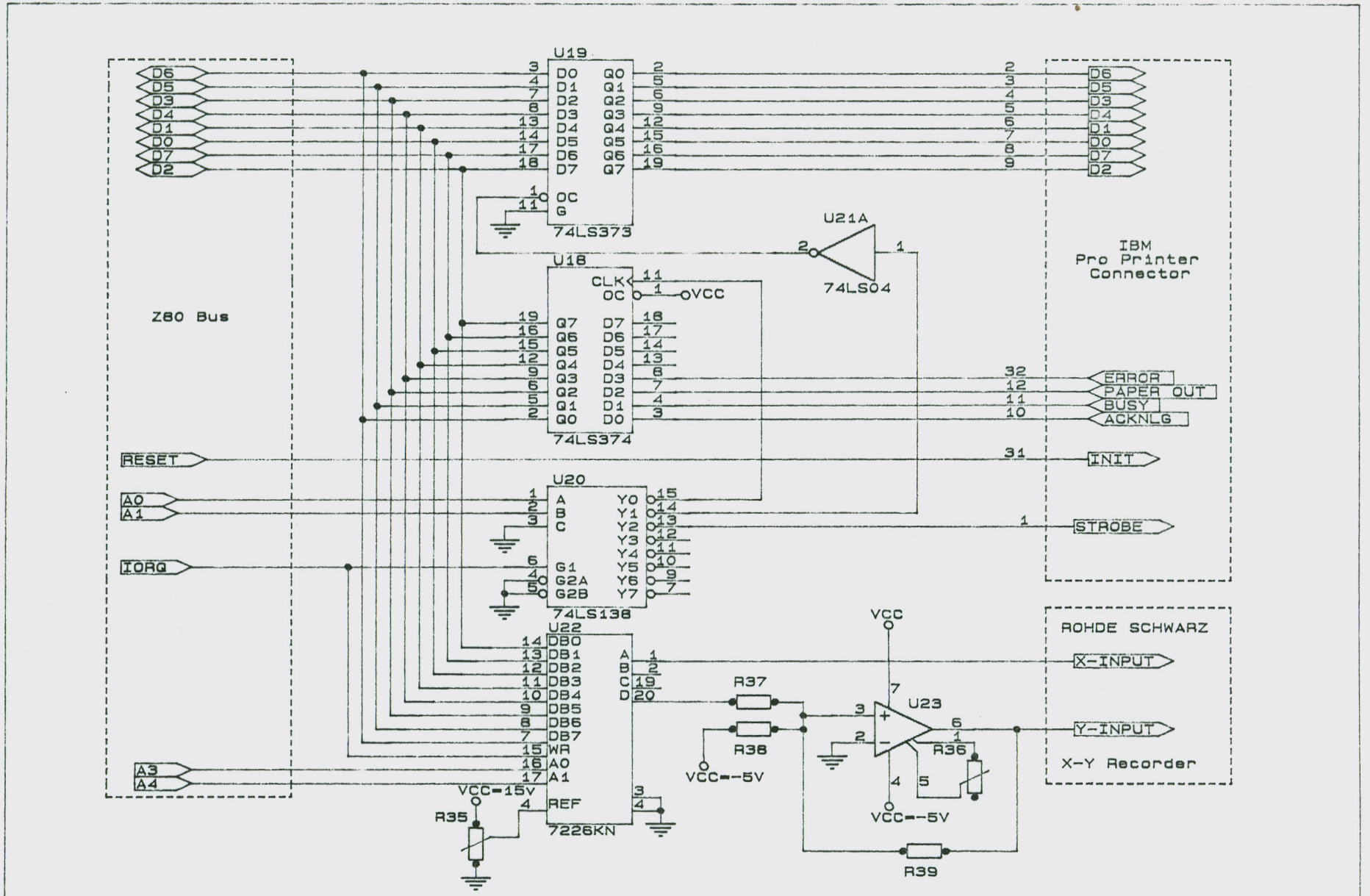


Fig. (2.2c) Schematic Diagram of Recorder & Printer Driver Board

2.1.2 Transducer and Interface Section

The purpose of the transducer and interface section is to convert an optical signal that exceeds the threshold value of the sensor into an electrical signal. In figure (2.3) the schematic diagram of an eight element receptor system is shown. The system will be described with the aid of this diagram.

2.1.2.1 Sensor Circuit

The photosensitive element used in this circuit is the SFH 309, 100mA silicon phototransistor. Eight such sensors were mounted on a special printed circuit board with a separation distance δ of 3cm. The output of the sensors were picked and fed to the interface circuit by a data cable as the sensor array sweeps across the object. In the circuit of figure (2.3) the sensors are part of the bias network for the BSX-19 switching transistor, so that with no light on the phototransistor, it is essentially an open circuit. In this condition the output which is taken across the emitter resistor is almost zero. Increasing the light on the sensor decreases its resistance, and the sensor turns on giving an output greater than zero. Because the phototransistor has a low current limitation (4mA) and may have to be matched to a low input impedance of the second stage, the transistor is used in an emitter follower configuration.

2.1.2.2 Threshold Circuit

The output from the sensor circuit is applied to one of the input of the voltage comparator (TDB 193) in the next stage. Connected to the other input of the comparator is a variable dc voltage. This voltage serves as a threshold voltage. The threshold voltage is determined by assuming that the output of the sensor circuit is linearly related to the percent of the sensor element's surface covered by light, that is for threshold of 0.5, the variable voltage is set equal to 0.5 times the maximum of sensor circuit output. The voltage comparator has a very high gain so that any input drives the amplifier into saturation. As soon as the sensor output exceeds the threshold value, the voltage comparator will be saturated. The output of each voltage comparator has an absolute value of approximately 5V, which is at TTL level and can be interfaced directly with TTL circuit.

2.1.3 Perturbation Section

The perturbation section of the system is concerned with moving the phototransistor array back and forth across the object surface. It is necessary that the amplitude and intensity of the signal should be controlled closely. For a perturbing signal, a sinusoidal and a triangular waves were chosen both for their convenience and ease of reproduction in the laboratory. The microcomputer is programmed to generate one of these signal at a given time.

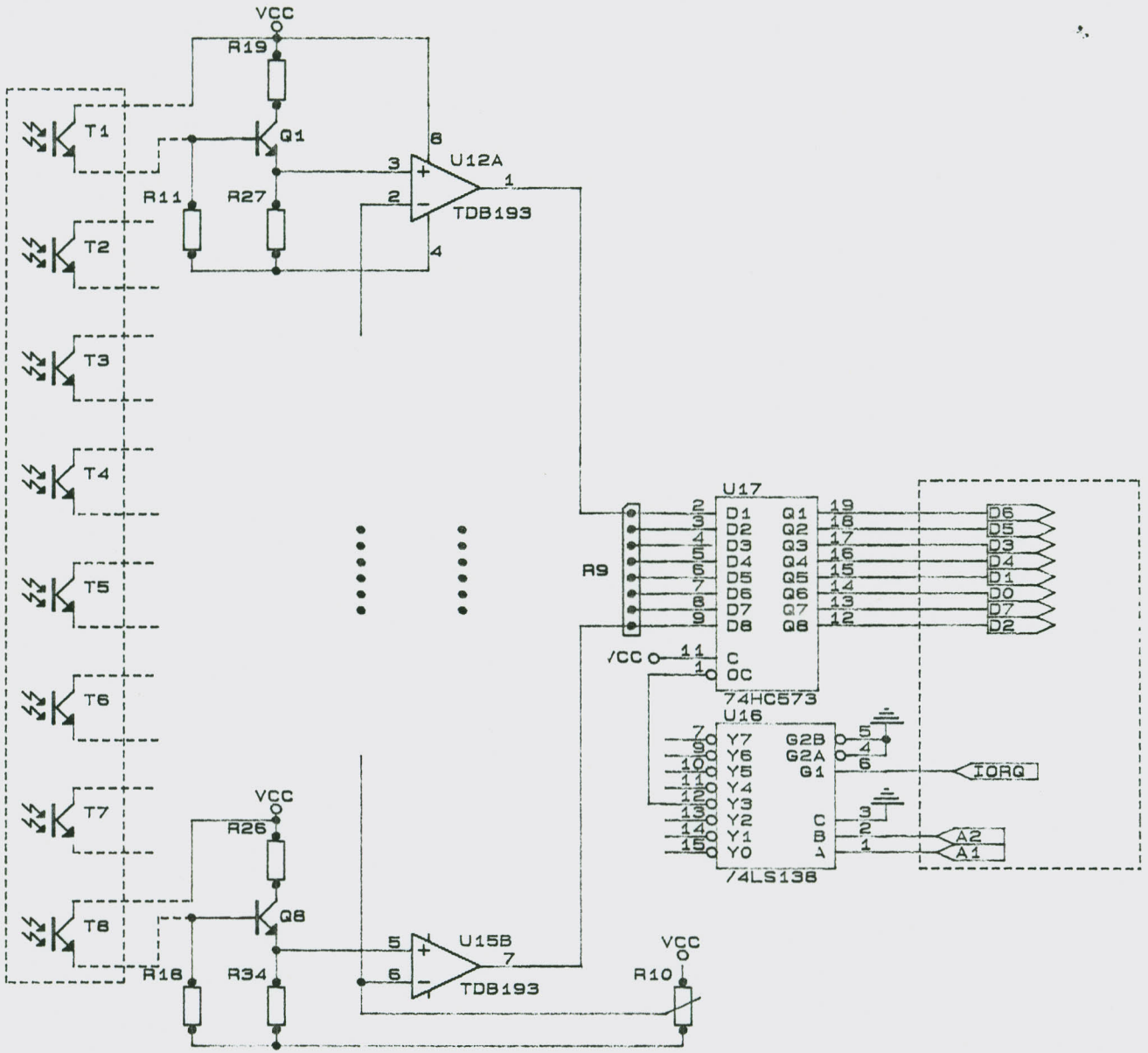


Fig. (2.3) Transducer & Interface Board

The system employs the ROHDE & SCHWARZ model 290-2016-06 x-y recorder as a scanning unit. A phototransistor array instead of a pen is fixed in the pen holder of the recorder. The recorder is driven by a special interface circuit, whose schematic diagram is shown in figure (2.2c). The circuit consists of a quad 8 bit digital to analog converter 7226KN (RS 300-388) and an LM 351 operational amplifier. Their main purpose is to provide a bipolar symmetrical perturbing signal through one of the channel and a staircase (ramp) wave form which is synchronized with the perturbing signal, through another channel. The image pattern is perturbed sinusoidally or triangularly in the vertical direction with high frequency and simultaneously move in the horizontal direction with low frequency. The frequency of the vertical motion is assumed to be very high as compared to that of the horizontal motion. This ensures that the x - position of the image is essentially constant during one vertical oscillation. The ramp and the perturbing signals are then sent to the x and y inputs of the recorder respectively to produce the scanning pattern shown in figure (2.4). The back and forth excursion of the sensor array is closely controlled by the V/cm amplifier, which is built in the recorder.

A light source is fitted beneath a transparent glass sheet and over it the eight phototransistors mounted on a PCB board can be moved freely. The glass sheet serves as a platform for placing the object under measurement. During the scanning process the phototransistor array is moved on the x-y plane as shown in figure (2.4). The phototransistor doesn't conduct as long as it is covered by the object and starts to conduct when it is uncovered or exposed to the light source. It is clear from the scanning diagram that the object is effectively divided in to many strip of equal width. Each strip can be represented as a rectangle of Δ units of width, while the length depends on the object image. The area of the object may be computed by knowing the length of each strip. The strip length is measured by perturbing the sensor array in the y-direction. In the design system, strip length are registered serially from the time the phototransistor array senses the object till the scanning operation of the object is completed. At the end of the scanning operation the array is returned to its initial position and the microcomputer system starts computing the area employing Simpson's algorithm from the length information gathered. The area of the object scanned will then be printed on the printer.

2.1.4 Testing the Functionality of the Boards

The complete hardware setup consisting of four circuit boards, viz, the CPU board, the memory board, printer adapter & recorder driver board and transducer & interface board, were installed in an aluminum chassis. Before installing, these four microprocessor controlled system components, they were tested individually for continuity, shorts and their functionality using Tektaonix 465B Oscilloscope and HP 1650A Logic analyzer. The transducer & interface board was tested by exposing the individual phototransistors to a torch light and measuring the voltage at the emitter leg of the

corresponding switching transistor. The threshold voltage was set first by measuring the emitter voltage of the transistor while the phototransistor was fully exposed to a light source. The voltage recorded at this time was 1.5V. Taking a threshold value of 0.85, the potentiometer voltage was set to a value 0.85 times the maximum sensor output, i.e. 1.5 x 0.85 = 1.30V.

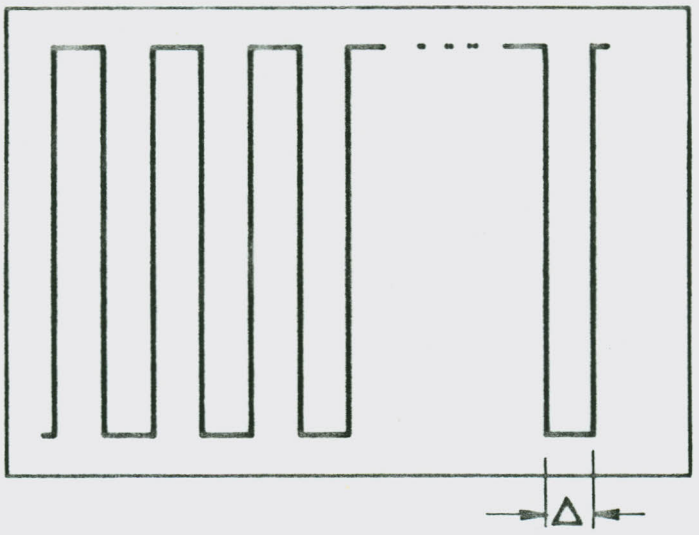


Figure (2.4) Scanning pattern of the recorder

2.2 System Software Design Detail

The software development proceeded concurrently with the hardware design and construction. The problems encountered during the hardware development may well call for substantial software modification like the inclusion of a delay subroutine rather than a timer circuit in the hardware. The software is divided into three distinct phases which are described in the following three sub-sections.

2.2.1 The Production of Flow Chart

In this stage of the software development, a logical sequential flow of the program is drawn bearing in mind that the area is computed from the length information and the length is measured by averaging the sensors output while they are perturbed back and forth. The program starts, see figure (2.5), with the initialization of the:

. stack pointer (SP) to 0000 Hex, end of RAM.
. position of the recorder pen to (1,14), which is the x & y position in cm as read from the scale of the recorder.
. register pair (de), to point to the base address of buffer #2, which holds strip length.

Once this is done, the microcomputer sweeps the array horizontally in step, with step width $\Delta = 0.1$ cm, until it detects the object. When the object is detected, the microprocessor records its x coordinate. At this x position, the array is perturbed vertically, and for each vertical position the array assumes, the microcomputer takes the sensors output and stores it in a contingent buffer, buffer #1. The signal processor then operates on buffer #1 to compute the average length of the image at that x position. This intermediate result is the length of the line image sensed by the perturbed array and saved in a separate buffer, buffer #2, and the length counter, count_2, is incremented. Next the pen holder of the recorder moves a step to the right and the average length of the object at this new x position is measured. This new average length is stored consecutively in buffer #2, and length counter is also incremented. The above procedure is repeated until the scanning operation is completed. The microprocessor then calculates the area from the length information which is stored in buffer #2 using Simpson's algorithm and the result is sent to the printer.

2.2.2 The Translation of the Flow Chart into Z80 Machine Code

The assembly language program was keyed into IBM AT microcomputer and compiled. This compiled program is in object code which is then converted into hexadecimal code and then finally into binary code for programming the EPROM. The program in hexadecimal code is in fact a machine code acceptable by the microcomputer. The hexadecimal program is listed in appendix F and is stored in a three and half inch diskette to facilitate documentation and further program modification.

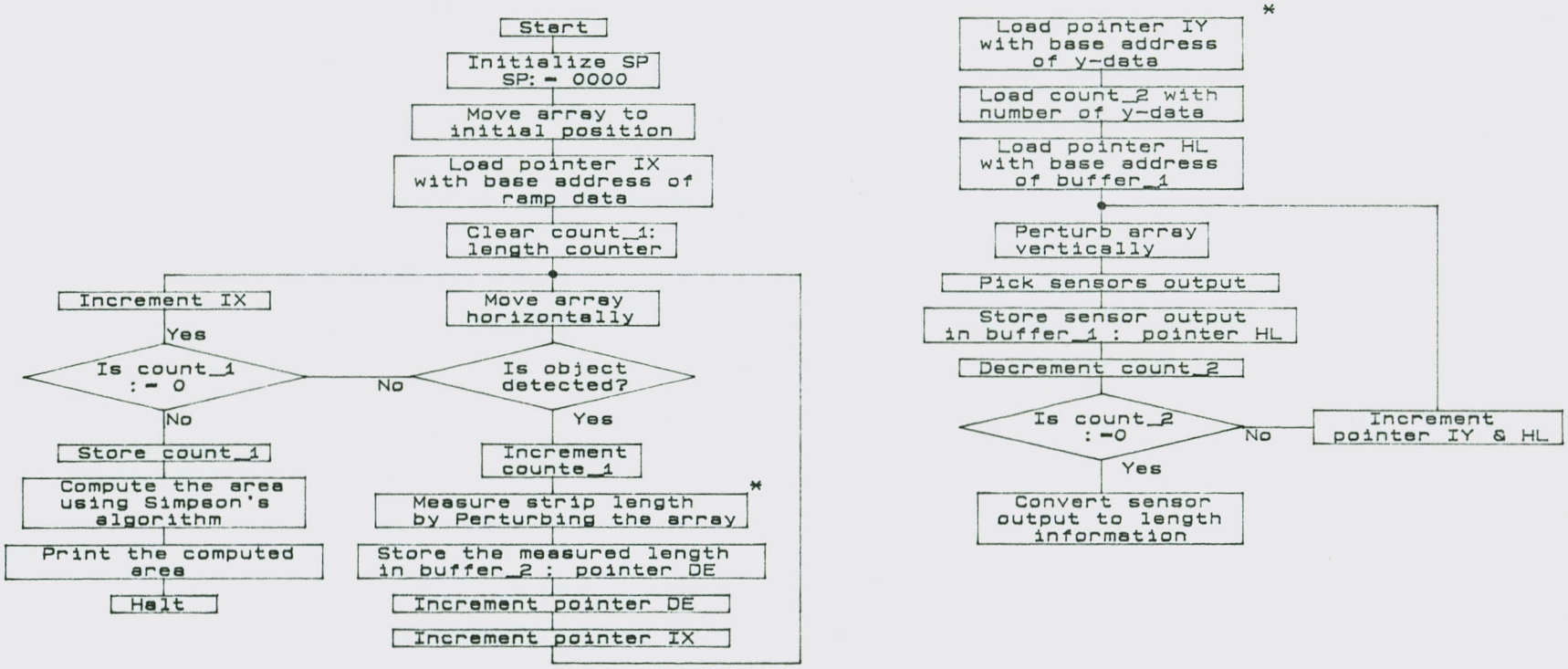


Fig. (2.5) Flow Chart of Area Measurement Using Microcomputer as a Controller

2.2.3 Testing the Program

The testing of the software is in fact the testing of the whole microcomputer system including the hardware. The testing was done piece wise, i.e. individual hardware and software modules were tested to determine whether the input output behaviour conformed to design specifications. The perturbing software together with the recorder and the DAC was tested using the special test routine. The test program for the sinusoidal perturbation is shown below:

```
dac_2 :          equ  13h              ; channel #2 of DAC
data_length :    equ  179
                 org  0
                 ld   sp, 0h
                 jp   100h
                 org  100h
test_perturb :   ld   iy, sinusoidal  ; load pointer with
                 ld   b, data_length  ; base address
repeat :         ld   a, 10           ; set recorder speed
                 call delay
                 call m_vertical      ; perturb vertically
                 inc  iy              ; increment pointer &
                 djnz repeat          ; repeat until all
                 jr    test_perturb   ; data are out
m_vertical :     ld   a, (iy+0)       ; load a with data
                 out  (dac_2), a      ; send it to channel
                 ret                  ; # 2 of the DAC
```

The same program was executed using triangular signal and the motion of the recorder was observed. The delay routine is included in the program to synchronize the high execution speed of the microcomputer and the slow motion of the recorder. A test program was also written to test the individual sensor output together with the corresponding threshold circuit. Next, the printer adapter and the printer were tested by sending ASCII code to the printer port #01. The routine which test the printer adapter is shown below.

```
sta_port :       equ  07h              ; status port
str_port :       equ  02h              ; strobe port
data_port :      equ  02h              ; data port
                 org  0
                 ld   sp, 0h           ; SP:= 0
test_printer :   jp   100h
                 org  100h
wait :           in   a, (sta_port)    ; get printer status
                 bit  1, a             ; is printer BUSY ?
                 jr   nz, wait         ; if yes, wait
                 ld   a, (hl)          ; get data to print
                 out  a, (data_port)   ; send it out
                 out  (str_port), a    ; strobe the data
                 ret
```

After these testing, a stage is reached where all the components modules were linked together into a whole system which again was tested for functionality to verify its input/ output characteristics.

2.3 Performance Testing of the Whole Microcomputer Control System

The final phase of testing placed the system with complete and wholly integrated hardware and software components in its working environments connected to the transducer, the recorder and the printer as shown in figure (2.6) and figure (2.7). In this final test, the program was burned into the 27C64Q EPROM. The burning process was done in INTEL Technology using Sunshine EPROM Writer card V-5.7 Model EW-901B, EW-904B. To compute the area of the object the following operations were carried out by the system.

a) The object is scanned in a manner shown in figure (2.4).
b) The first & the last line lengths and the odd & even line lengths should be identified to facilitate the application of Simpson's rule for computing the area.
c) The system should wait for a reset signal before it start of the next scanning operation.
d) Area should be computed and displayed in the desired unit.

Two important points should be mentioned here about the scanning operation,

1) A two dimensional image can be considered as an ensemble of infinite number of vertical lines. And as the array sweeps in the x direction, the measured line length ι_i, i = 1, ... , n varies with the shape of the image.
2) The return path is arbitrary and the array may be covered during this time, and alter the final result of the array measurement. Therefore, this extra unwanted signal should be eliminated.

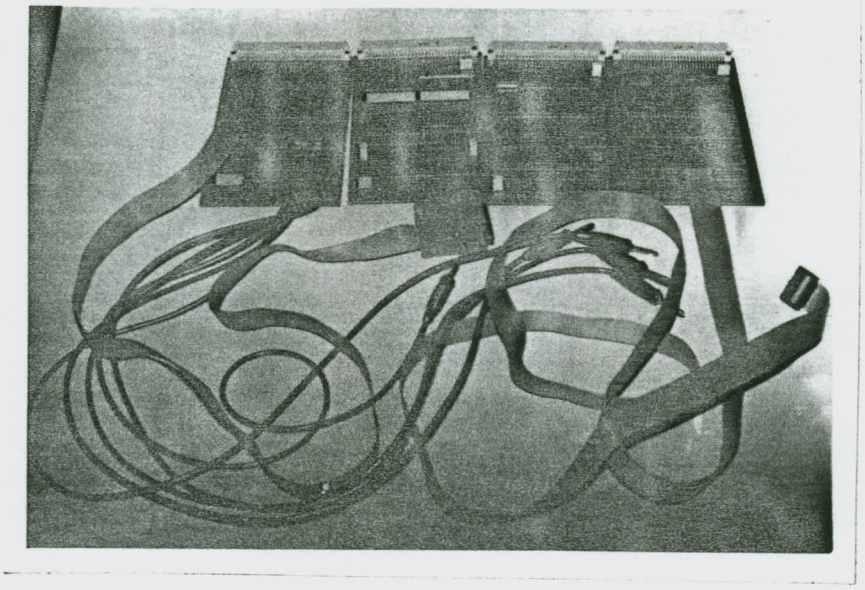


Figure (2.6) The System Boards

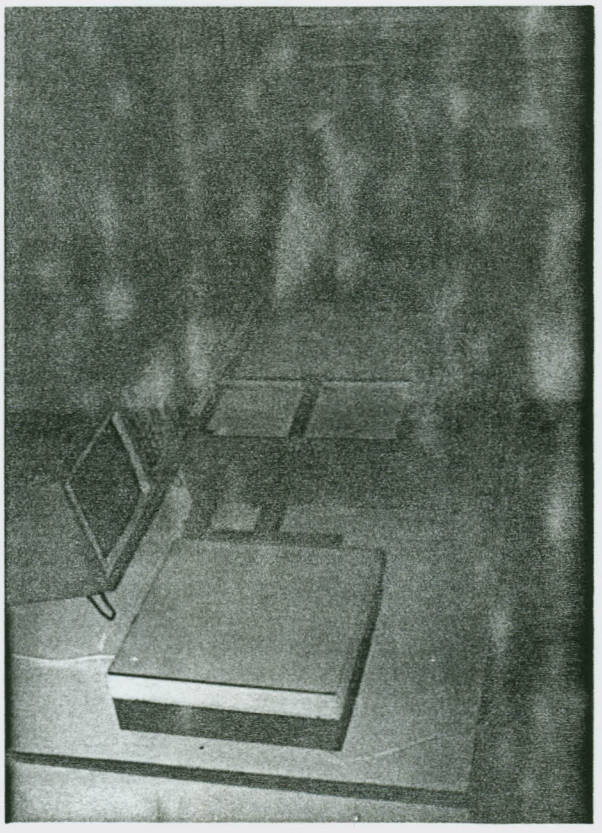


Figure (2.7) The Experimental Setup

3 Results and Conclusion

3.1 Theoretical Results

In this section data will be presented to demonstrate how well the ideal model predicts the system real performance. To verify perturbation theory four (three regular & one irregular) plane geometric bodies were considered. Table (3.1) shows types of image shapes considered and there areas as calculated theoretically.

Image	Calculated area interms of δ^2	Actual numerical value in cm^2
Square	$36\delta^2$	324.000
Circle	$9\pi\delta^2$	254.469
Ellipse	$6\pi\delta^2$	124.093
Irregular	-	177.606

δ = 3cm, dead space

Table (3.1) Theoretical area for selected image shapes

3.2 Experimental Results

In the experimental procedure three modes of operations were analyzed: 1) static image (no perturbation), 2) sinusoidal image perturbation, and 3) triangular image perturbation. The experimental data were taken for the same geometric bodies as the theoretical images. These results are tabulated in table (3.2a) - (3.2c) for the three modes of operations.

Image shape	Measured area in cm^2
Square	235.967
Circle	177.867
Ellipse	82.300
Irregular	100.000

Table (3.2a) Static (no perturbation)

Image Shape	Measured area in cm^2		
	Level of perturbation (B)		
	0.5	1.0	1.5
Square	249.331	282.465	304.828
Circle	212.352	225.654	252.987
Ellipse	93.115	105.279	122.635
Irregular	115.668	150.640	167.666

Table (3.2b) Sinusoidal perturbation

Image Shape	Measured area in cm^2		
	Level of perturbation (B)		
	0.5	1.0	1.5
Square	270.020	283.010	316.084
Circle	224.988	234.462	253.532
Ellipse	105.101	110.101	123.472
Irregular	126.341	159.410	175.640

Table (3.2c) Triangular perturbation

3.3 Error Analysis

The results presented here are from the theoretical calculation and the experimental system. The quantization of the object image gives rise to an area measurement error. This error is a function of the shape, complexity, and location of the image with respect to the array elements. To provide, a quantitative measure of the effect of image perturbation, a performance index (PI) was developed in [4] as,

$$J = \frac{|A_a - A_m|}{A_a} \qquad (3.1)$$

Where A_a is the actual area of the image projected on the array and A_m is the area indicated by the system output indicator. PI measures the percentage deviation of the measured area from the theoretical value. The analysis of the system area measurement error was made for the selected image shapes under the three modes of operations. The calculated PI for each mode is tabulated in table (3.3).

Image shape	Performance index (J)						
	Mode of operations						
	Static	Sinusoidal			Triangular		
		Level of perturbation			Level of perturbation		
		0.5	1.0	1.5	0.5	1.0	1.5
Square	0.37	0.30	0.15	0.06	0.20	0.12	0.03
Circle	0.43	0.20	0.13	0.005	0.13	0.09	0.003
Ellipse	0.51	0.33	0.19	0.01	0.18	0.13	0.005
Irregular	0.78	0.54	0.18	0.06	0.41	0.11	0.01

Table (3.3) Area measurement error for selected image shapes

3.4 Conclusion

In the past several years many theoretical papers, have been published on the feasibility of using image perturbation as a method of improving the measurement accuracy of discrete receptor elements.

Although an experiment with two photosensors have been previously reported [2], this paper describes the first complete implementation and detailed experimentation of image perturbation for the accuracy of area measurement. The microcomputer based system developed in this paper for measurement of area is capable of handling object of 20x20cm size. The limitation on the size is mainly due to the scanning unit employed.

In order to asses the performance of the system, samples of different sizes and shapes were selected and their areas measured for both static (no perturbation) and perturbed images. The measurement of area with no image perturbation results in relatively large error. The use of sinusoidal image perturbation causes a significant improvement in accuracy. Even more accuracy was obtained when triangular image perturbation is used [9]. Error in measurement for square object was found to be maximum. This is expected as Simpson's rule adopted for calculation is more accurate for curved boundaries.

The area computed by Simpson rule is the area of the object enclosed between the first and last line scanned on the object. This simple calculation result in negative error as small piece of object on either end are left unaccounted for. The area left unaccounted for depends on the object position and strip width selected. It is irregular in shape. It is obvious that the length of the strip left out are equal to the first and last length of the object, the width of these strip may vary from 0 to Δ depending on the position of the

object. The object boundary on both sides is treated as an area of a parabola to estimate the area of the portion left out. The area of the these portions may be estimated from the relation,

$$A_p = \frac{2}{3}(\iota_1\Delta_1+\iota_n\Delta_n) \qquad (3.2)$$

Where ι_1, ι_n are length of first and last line, respectively and Δ_1, Δ_n are the width of the left out portions. For $\Delta_1 = \Delta_n = \Delta/2$, the left out area is equal to,

$$A_p = \frac{\Delta}{3}(\iota_1+\iota_n) \qquad (3.3)$$

The total area is then given by the relation

$$A_T = A_m+A_p = A_m+\frac{\Delta}{3}(\iota_1+\iota_n) \qquad (3.4)$$

Possible applications of area measurement system are in the area of process control or in process where sorting according to shape and size is desired. Since the area measurement system measures length in two directions, it may be feasible to extend the system, so that actual object shape could be determined.

Appendix A

Derivation of Equation (1.13)

The equation and plot of the normal distribution curve are,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \qquad \textbf{(a.1)}$$

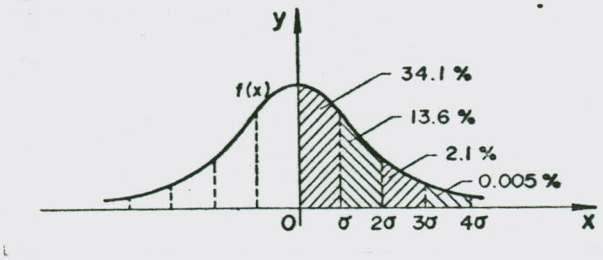


Figure (a.1) Normal distribution curve

Assuming that f(x) ≅ 0 ∀ x such that, |x| > 4σ, we get the approximated normal curve of figure (a.2).

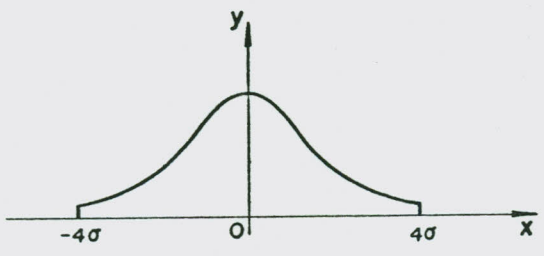


Figure (a.2) Approximated normal curve

Rewriting equation (1.5),

$$S(\iota, a, f, \delta) = \sum_{i=-\infty}^{\infty} \left(\int_{(i-a)\delta-\iota}^{(i-a)\delta} f(x)\, dx \right) \qquad \textbf{(a.2)}$$

and terminating computation when the limit of the integral exceeds 4σ, we get the lower and upper bounds of i as,

$$\textit{Upper limit } (i_{max}) : (i-a)\,\delta - \iota \leq 4\sigma$$

$$i \leq a + \frac{4\sigma+\iota}{\delta} \qquad \text{(a.3)}$$

$$i_{max} = \lceil a + \frac{4\sigma+\iota}{\delta} \rceil$$

$$\textit{Lower limit } (i_{min}) : (i-a)\,\delta \geq -4\sigma$$

$$i \geq a - \frac{4\sigma}{\delta} \qquad \text{(a.4)}$$

$$i_{min} = \lfloor a - \frac{4\sigma}{\delta} \rfloor$$

where

$\lceil x \rceil$ means the largest integer less than or equal to x.

$\lfloor x \rfloor$ means the smallest integer greater than or equal to x.

If we define normalized variance σ_n and normalized length ι_n as,

$$\sigma_n = \frac{\sigma}{\delta} \quad ; \quad \iota_n = \frac{\iota}{\delta} \qquad \text{(a.5)}$$

then, equations (a.3) and (a.4) can be written as,

$$i_{min} = \lfloor a - 4\sigma_n \rfloor \quad ; \quad i_{max} = \lfloor a + 4\sigma_n + \iota_n \rfloor \qquad \text{(a.6)}$$

and the system output, equation (a.2), will be,

$$S(\iota, a, \sigma_n, \delta) = \sum_{i=i_{min}}^{i_{max}} \left(\int_{(i-a)\delta-\iota}^{(i-a)\delta} \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} dx \right) \qquad \text{(a.7)}$$

Making a change of variable x = x / σ, we get,

$$S(\iota_n, a, \sigma_n, \delta) = \sum_{i=i_{min}}^{i_{max}} \left(\int_{\frac{i-a-\iota_n}{\sigma_n}}^{\frac{i-a}{\sigma_n}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx \right) \qquad \text{(a.8)}$$

(Q.E.D)

Appendix B

Program Listing for Length Measurement

```
Program Length_measurement ( Input, Output );

Uses printer;

Const
              delta = 0.1;
           position = 0;
          intervals = 50;
          dimension = 50;
normalized_variance = 0.125;

Type
            vector = array [ 0..dimension ] of real;

Var
                                counter : integer;
          length_input, system_output : vector;
        partial_sum, normalized_length : real;

{$I plotter.pas}
{$I include.pas}

Procedure simpson (  a, b : real;
                        n : integer;
                     var
                        i : real );
Var
          j : integer;
  h, x, sum : real;

Begin
  x := a;
  h := (b-a)/n;
  sum := gaussian ( x );
  j := 1;
  Repeat
    x := x + h;
    Case j mod 2 of
      0 : sum := sum + 2 * gaussian ( x );
      1 : sum := sum + 4 * gaussian ( x );
    End;
    j := j + 1;
  Until j = n;
  sum := sum + gaussian ( b );
  i := h/3 * sum;
End;

Procedure compute_system_output;

Var
  i, i_min, i_max : Integer;
  term,  x_1, x_2 : Real;

Begin
  i_min := largest_integer ( position - 4 *
           normalized_variance );
  i_max := smallest_integer ( position + 4 *
           normalized_variance + normalized_length );
  partial_sum := 0;
  For i := i_min to i_max do
    Begin
      x_1 := ( i - position - normalized_length ) /
             normalized_variance;
      x_2 := ( i - position ) / normalized_variance;
      simpson ( x_1, x_2, intervals, term );
      partial_sum := partial_sum + term;
    End;
  system_output [ counter ] := partial_sum;
End;

Procedure prepare_plotter;
Begin
  initialize_plotter ( 1, -2, 7, -2, 7 );
  draw_axis ( 2, 12, 12, 0, 0, -1, -1, 0.5, 0.5 );
  write_label ( 2 ,0.5, 1, 25, 25, 0, 0, 1, 1, 0, 0, 5, 5,
                'Normalized Length Input ',
                'System Output in Volt ' );
End;

Begin
  prepare_plotter;
  normalized_length := 0;
  For counter := 0 to dimension do
    Begin
      compute_system_output;
      normalized_length := normalized_length + delta;
    End;
  draw ( 5, dimension, 0, delta, 1, system_output );
  write_key ( 2, 0, 4.5, 6, 'Normalized Sigma = 1/8' );
  write_key ( 5, 0.2, 5.5, 5.7, 'a =   0 ' );
  write_key ( 2, 0, 2, -1.5, 'Fig.(1.4)a' );
End.
```

Appendix C

Derivation of Equation (1.23)

Rewriting equation (1.21)

$$S(A) = \frac{1}{\delta}\sum_{i=-\infty}^{\infty}\left(\int_{x_1}^{x_2} f_1(i\delta - x)\,(g_2(x) - g_1(x))\,dx\right) \qquad \text{(c.1)}$$

where,

$$f_1(i\delta - x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{i\delta - x}{\sigma}\right)^2} \qquad \text{(c.2)}$$

Since computation is terminated when the limit of the integral exceeds 4σ, this will limit the lower and upper bound of i as,

$$\textit{Upper limit } (i_{max}) : i\delta - x_2 \leq 4\sigma$$

$$i \leq \frac{x_2 + 4\sigma}{\delta} \qquad \text{(c.3)}$$

$$i_{max} = \lceil \frac{x_2 + 4\sigma}{\delta} \rceil$$

$$\textit{Lower limit } (i_{min}) : i\delta - x_1 \geq -4\sigma$$

$$i \geq \frac{x_1 - 4\sigma}{\delta} \qquad \text{(c.4)}$$

$$i_{min} = \lfloor \frac{x_1 - 4\sigma}{\delta} \rfloor$$

Define normalized variance as,

$$\sigma_n = \frac{\sigma}{\delta} \qquad \text{(c.5)}$$

then S(A) becomes,

$$S(A) = \frac{1}{\delta} \sum_{i=i_{min}}^{i_{max}} \left(\int_{x_1}^{x_2} f_1(i\delta - x)\,(g_2(x) - g_1(x))\,dx \right) \qquad \text{(c.6)}$$

for a particular geometry chosen, see figure (c-1), $g_1(x)$ and $g_2(x)$ can be computed as,

$$g_1(x) = y_o - \sqrt{r^2 - (x-x_o)^2}$$

$$g_2(x) = y_o + \sqrt{r^2 - (x-x_o)^2} \qquad \text{(c.7)}$$

$$g_2(x) - g_1(x) = 2\sqrt{r^2 - (x-x_o)^2}$$

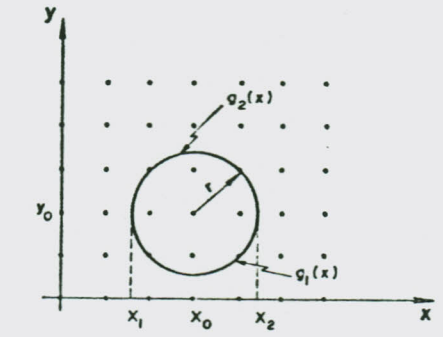


Figure (c.1) Projection of circular image on matrix of elements

and for a given circular radius, x_1 and x_2 can be computed as,

$$x_1 = x_o - r \quad ; \quad x_2 = x_o + r \qquad \text{(c.8)}$$

so, S(A) becomes,

$$S(A) = \frac{1}{\delta} \sum_{i=i_{min}}^{i_{max}} \left(\int_{x_o-r}^{x_o+r} 2 f_1(i\delta - x) \sqrt{r^2 - (x-x_o)^2}\,dx \right) \qquad \text{(c.9)}$$

making a change of variable $y = x/\sigma$, we get,

$$S(A) = \frac{1}{\delta} \sum_{i=i_{min}}^{i_{max}} \left(\int_{\frac{x_o-r}{\sigma}}^{\frac{x_o+r}{\sigma}} 2\sigma\, f_1(i\delta - \sigma y) \sqrt{r^2 - (\sigma y - x_o)^2}\,dy \right) \qquad \text{(c.10)}$$

defining,

$$c = \frac{x_o}{\sigma} \quad ; \quad r_n = \frac{r}{\sigma} \qquad \text{(c.11)}$$

and using equations (c.5), (c.8) and (c.11), equations (c.3) and (c.4) can be written as,

$$i_{max} = \lceil (c+r_n+4)\,\sigma_n \rceil \quad ; \quad i_{min} = \lfloor (c-r_n-4)\,\sigma_n \rfloor \qquad \text{(c.12)}$$

making a simple change of variable y = x in equation (c.10) we get,

$$S(A) = \sum_{i=i_{min}}^{i_{max}} \left(\int_{c-r_n}^{c+r_n} 2\sigma_n \sigma f_1(i\delta - \sigma x) \sqrt{r_n^2 - (x-c)^2}\, dx \right) \qquad \text{(c.13)}$$

(Q.E.D)

Appendix D

Program Listing for Area Measurement

```
Program Area_measurement ( Input, Output );

Uses printer;

Const
                    delta = 0.1;
                intervals = 50;
                dimension = 100;
      normalized_center = 0;
    normalized_variance = 0.125;

Type
                  vector = array [ 0..dimension ] of real;

Var
                           counter : integer;
     radius_input, system_output : vector;
  partial_sum, normalized_radius : real;

{$I plotter.pas}
{$I include.pas}

Function f ( x : real; y : integer ) : real;

Begin
  f := 2 * normalized_variance / sqrt(2*pi) *
       sqrt (sqr ( normalized_radius ) - sqr ( x - normalized_center ) ) *
       exp ( -0.5 * sqr (x - y / normalized_variance));
End;

Procedure simpson ( a, b : real; n, k : integer; Var i : real );
Var
          j : integer;
  h, x, sum : real;

Begin
  x := a;
  h := (b-a)/n;
  sum := f ( x, k );
  j := 1;
  Repeat
    x := x + h;
    Case j mod 2 of
      0 : sum := sum + 2 * f ( x, k );
      1 : sum := sum + 4 * f ( x, k );
    End;
    j := j + 1;
  Until j = n;
  sum := sum + f ( b, k );
  i := h/3 * sum;
End;

Procedure compute_system_output;
Var
  i, i_min, i_max : integer;
   term, x_1, x_2 : real;

Begin
  i_min := largest_integer ( normalized_variance *
           ( normalized_center - normalized_radius - 4 ) );
  i_max := smallest_integer ( normalized_variance *
           normalized_center + normalized_radius + 4 ) );
  partial_sum := 0;
  For i := i_min to i_max do
    Begin
      x_1 := normalized_center - normalized_radius;
      x_2 := normalized_center + normalized_radius;
      simpson ( x_1, x_2, intervals, i, term );
      partial_sum := partial_sum + term;
    End;
  system_output [ counter ] := sqrt ( partial_sum / pi );
End;

Procedure prepare_plotter;
Begin
  initialize_plotter ( 1, -1, 11, -1, 6 );
  draw_axis ( 2, 20, 5, 0, 0, 0, 0, 0.5, 1 );
  write_label ( 2, 0.5, 1, 30, 50, 0, 0, 1, 1, 0, 0, 10, 5,
                'Normalized Circular Radius ',
                'Square Root of System Output as a Multiple of Root Pi' );
End;

Begin
  prepare_plotter;
  normalized_radius := 0;
  For counter := 0 to dimension do
    Begin
      compute_system_output;
      normalized_radius := normalized_radius + delta;
    End;
  draw ( 5, dimension, 0, delta, 1, system_output );
  write_key ( 5, 0.2, 2, 5, 'Normalized Sigma = 1/8' );
  write_key ( 2, 0, 4.5, -0.5, 'Fig.(1.6)' );
End.
```

Appendix E

Program Listing for Contour Detection

```
Program Contour_detection ( Input, Output );

Uses printer;

Const
                delta = 0.1;
               radius = 2.0;
             variance = 0.125;
            intervals = 50;
            dimension = 40;

Type
                vector = array [ 0..dimension ] of extended;

Var
              counter : integer;
        length_input,
      system_output_1,
      system_output_2 : vector;
            x, moment,
     length, c_length,
    normalized_length,
   centroid, position : extended;

{$I plotter.pas}
{$I include.pas}

Procedure compute_length_and_position;

Var
  temp_1,
  temp_2, temp_3 : real;

Begin
  temp_1 := sqr ( radius );
  temp_2 := sqr ( x );
  temp_3 := 4 * ( temp_1 - temp_2 );
  c_length := sqrt ( temp_3 );
  position := radius - c_length / 2;
End;

Procedure compute_system_output;

Var
                i,
     i_min, i_max : integer;
    term, y_1, y_2 : real;

Begin
  i_min := largest_integer ( position - 4 * variance );
  i_max := smallest_integer ( position + 4 * variance +
           c_length );
  length := 0;
  moment := 0;
  For i := i_min to i_max do
    Begin
      y_1 := ( i - position - c_length ) / variance;
      y_2 := ( i - position ) / variance;
      simpson_1 ( y_1, y_2, intervals, term );
      length := length + term;
      moment := moment + i * term;
    End;
  If length = 0 then
    centroid := position
  Else
    centroid := moment / length;
  system_output_1 [ counter ] := centroid + 0.5 * length;
  system_output_2 [ counter ] := centroid - 0.5 * length;
End;

Procedure prepare_plotter;

  Begin
    initialize_plotter ( 1, -3, 3, -1, 5 );
    draw_axis ( 2, 4, 0, 0, 0, -2, 0, 1, 0 );
    write_label ( 2, 0.5, 1.0, 10, 0, 0, 0, 1, 1,
                  -2, 0, 2, 0, ' x - axis ', 11 );
  End;

Begin
  prepare_plotter;
  x := - radius;
  counter := 0;
  Repeat
    compute_length_and_position;
    compute_system_output;
    x := x + delta;
    counter := counter + 1;
  Until counter > dimension;
  draw ( 5, dimension, -radius, delta,1, system_output_1 );
  draw ( 5, dimension, -radius, delta,1, system_output_2 );
  write_key ( 5, 0, 1.5, 4.5, 'Sigma = 1/8' );
  write_key ( 2, 0, -0.5, -1.0, 'Fig.(1.9)' );
End.
```

Appendix F

System Software Listing

```
                ; data size listing
x_data_length   equ     256
y_data_length   equ     100
                ; port address listing
        dac_1   equ     00bh
        dac_2   equ     013h
     sen_port   equ     007h
     sta_port   equ     000h
     str_port   equ     002h
    data_port   equ     001h
                org     0
                nop
                nop                     ; allow processor to
                nop                     ; execute nop instruction
                nop                     ; for 20 clock cycles
                nop
                jp      0100h           ; skip vector interrupt area
                org     0100h
                ;
                ; routine : main
                ; purpose : scan the image by perturbing the
                ;           photosensitive array and print
                ;           its area
                ;   entry : object is placed on the transparent glass
                ;    exit : the area of the image will be printed and
                ;           the processor is halted
                ;
          main :
                ld      sp, 0h          ; initialize stack pointer
                ld      b, 0            ; initialize length counter
                ld      ix, ramp        ; ix pointer to ramp data
                ld      (ix+0), 0       ; initial ramp value 0
                ld      de, len_buffer  ; de pointer to length buffer
          scan :
                call    horizontal      ; move horizontally
                ld      a, 5            ; set forward speed
                call    delay
                in      a, (sen_port)
                cp      0ffh            ; is object beneath array?
                jr      z, no_object    ; no
                call    perturb         ; yes, perturb the array
                call    _convert        ; convert array output to
                                        ; length information
                call    length          ; compute the average length
                inc     b               ; increment length counter
                inc     de              ; increment pointer
        not_yet :
                inc     (ix+0)          ; increment ramp value
                jr      scan            ; continue scanning
       no_object :
                ld      a, b            ; is object not yet detected
                or      a               ; or has been scanned
                jr      z, not_yet      ; and passed?
                call    simpson         ; has been scanned and passed
                                        ; compute the image area from
                                        ; length information using
                                        ; simpson algorithm
                call    print           ; print the area
                call    retrace         ; send recorder to the origin
                halt                    ; halt processor (job is complete)

                ;
                ; routine : horizontal
                ; purpose : move the array from the current
                ;           x-position one step to the right.
                ;   entry : register ix = pointer to x data
                ;    exit : array is moved a step to the right
                ;
      horizontal :
                ld      a, (ix+0)       ; load data pointed by ix to a
                out     (dac_1), a      ; send it to the DAC port #1
                ret

                ;
                ; routine : vertical
                ; purpose : move the array from the current
                ;           y-position one step up or down.
                ;   entry : register iy = pointer to y data
                ;    exit : array is moved either up or down
                ;
        vertical :
                ld      a, (iy+0)       ; load data pointed by iy to a
                out     (dac_2), a      ; send it to the DAC port #2
                ret
```

```
                ;
                ; routine : perturb
                ; purpose : perturb the array vertically and at
                ;           the same time pick sample snapshots
                ;    exit : array is perturbed and sensor
                ;           buffer is filled by snapshots data
                ;
        perturb :
                push    bc
                ld      iy, sinusoidal ; choose sinusoidal signal
                ld      hl, sen_buffer ; hl, pointer to sensor buffer
                ld      b, y_data_length
                                       ; register b holds # of
                                       ; perturbation points
           more :
                call    vertical       ; perturb
                ld      a, 10          ; set speed of perturbation
                call    delay
                in      a, (sen_port)  ; pick snapshot
                ld      (hl), a        ; store the snapshot pointed by hl
                inc     hl             ; increment pointer
                inc     iy             ; get another perturbing data
                djnz    more           ; repeat until # of
                                       ; perturbation point is 0
                pop     bc
                ret                    ; end perturbation

                ;
                ; routine :  convert
                ; purpose : this routine convert the array
                ;           output to length information.
                ;    exit : the content of the buffer is
                ;           modified to reflect length
                ;
        _convert :
                push    bc
                ld      hl, sen_buffer
                ld      b, y_data_length
          loop :
                ld      a, (hl)        ; get snapshot
                ld      c, 0           ; clear register c
         bit_0 :
                bit     0, a           ; is bit 0 low?
                jr      nz, bit_1      ; no
                inc     c              ; yes
         bit_1 :
                bit     1, a           ; is bit 1 low?
                jr      nz, bit_2      ; no
                inc     c              ; yes
         bit_2 :
                bit     2, a           ; is bit 2 low?
                jr      nz, bit_3      ; no
                inc     c              ; yes
         bit_3 :
                bit     3, a           ; is bit 3 low?
                jr      nz, bit_4      ; no
                inc     c              ; yes
         bit_4 :
                bit     4, a           ; is bit 4 low?
                jr      nz, bit_5      ; no
                inc     c              ; yes
         bit_5 :
                bit     5, a           ; is bit 5 low?
                jr      nz, bit_6      ; no
                inc     c              ; yes
         bit_6 :
                bit     6, a           ; is bit 6 low?
                jr      nz, bit_7      ; no
                inc     c              ; yes
         bit_7 :
                bit     7, a           ; is bit 7 low?
                jr      nz, no_bit     ; no
                inc     c              ; yes
        no_bit :
                dec     c              ; decrement c to reflect length
                ld      (hl), c        ; length := (c) * delta, store c
                inc     hl             ; get another snapshot and repeat
                djnz    loop           ; until end of sensor buffer size
                pop     bc             ; is reached
                ret                    ; end conversion

                ;
                ; routine : print_buffer
                ; purpose : print the content of a buffer.
                ;           the length of the buffer is specified
                ;           in the first buffer location
                ;   entry : hl = buffer pointer
                ;
  print_buffer :
                ld      b, (hl)
                ld      a, b
                or      a
                jr      z, end_print
    print_more :
                inc     hl
                call    char_print
                djnz    print_more
     end_print :
                ret
```

```
                ;
                ; routine : length
                ; purpose : sum each sample of 8-bit length
                ;           information, yielding a 16 bit result.
                ;    exit : the average length is stored in a
                ;           buffer pointed by register pair de.
                ;
      length :
                push    bc
                ld      hl, sen_buffer
                ld      b, y_data_length
                sub     a               ; register pair ca
                ld      c, a            ; is cleared to start summation
       _sum :
                add     a, (hl)         ; a := a + (hl)
                jr      nc, jump        ; if carry increment
                inc     c               ; the msb byte
       jump :
                inc     hl              ; increment source address
                djnz    _sum            ; decrement size of array
                                        ; and jump to sum until
                                        ; size equals to zero
                ld      (de), a         ; store low byte of length in
                inc     de              ; a location pointed by de,
                ld      a, c            ; increment de, store high byte
                ld      (de), a         ; of length in a location
                pop     bc              ; pointed by de
                ret                     ; end average length

                ;
                ; routine : simpson
                ; purpose : computes the area of the perturbed
                ;           image from the average length of
                ;           information using simpson algorithm.
                ;    exit : register hl = integral part of the area
                ;           register de = fractional part of the area
                ;
     simpson :
                ld      hl, len_buffer ; hl points to length buffer
                call    adjust          ; call adjust
                call    sum             ; sum up the lengths of each x
                ld      ix , location_1
                ld      (ix+0), l       ; store register ahl in dividend
                ld      (ix+1), h       ; array
                ld      (ix+2), a
                ld      de, 1000        ; de := 1000
                ld      (ix+0), e       ; store register de in divisor
                ld      (ix+1), d       ; array
                ld      (ix+2), 0
                ld      hl, location_1
                ld      de, location_2
                ld      b, 3
                call    divide          ; call multiple precision
                ld      e, (hl)         ; division
                inc     hl              ; de = remainder
                ld      d, (hl)         ; (hl) := quotient
                ld      hl, (location_1)
                ret                     ; end simpson

                ;
                ; routine : retrace
                ; purpose : send the pen holder of the
                ;           recorder to original position
                ;
     retrace :
                ld      a, 5            ; set retrace speed
                call    delay
                dec     (ix+0)
                call    horizontal      ; retrace one step
                or      a               ; is recorder at origin?
                jr      nz, retrace     ; no, keep on retracing
                ret                     ; yes

                ;
                ; routine : char_print
                ; purpose : send a character to the printer
                ;   entry : hl = pointer to the character
                ;                to be printed
                ;
  char_print :
                in      a, (sta_port)   ; is printer BUSY?
                bit     1, a
                jr      nz, char_print  ; yes
                ld      a, (hl)         ; no, get data addressed by hl
                out     (data_port), a  ; data out
                out     (str_port ), a  ; strobe out
                ret
```

```
;
; routine : adjust
; purpose : adjust the array of length, so that simpson
;           algorithm can be applied very easily to
;           compute the area of the object. it multiply
;           odd length by 4, even length by 2 and leave
;           the two end point lengths.
;   entry : register hl = pointer to start of buffer
;           register b  = contains number of data points
;    exit : content of the buffer is adjusted
;
adjust :
        push    hl              ; save pointer
        push    bc              ; save length counter b
        dec     b
        ld      c, 0            ; clear register c
        inc     hl              ; skip the first length
 loop_ :
        inc     hl              ; hl points to next length
        inc     c               ; increment length counter
        ld      a, c
        cp      b
        jr      z, skip         ; skip the last length, too
        srl     a               ; is line odd or even line?
                                ; divide by 2, remainder
                                ; in carry bit.
        jr      nc, by_2        ; if carry = 0 line is even
        or      a               ; clear carry
        rl      (hl)
        inc     hl
        rl      (hl)            ; multiply length by 4
        dec     hl
  by_2 :
        or      a               ; clear carry
        rl      (hl)
        inc     hl
        rl      (hl)            ; multiply by 2
        jr      loop_           ; go through the buffer
  skip :
        pop     bc              ; get length counter
        pop     hl              ; get pointer
        ret                     ; end adjust

;
; routine : sum
; purpose : sum the elements of an array, yielding
;           a 24 bit result. maximum size is 255
;           16-bit elements.
;   entry : register pair hl = base address of array
;           register b = size of array in words
;    exit : register a = high byte of sum
;           register h = middle byte of sum
;           register l = low byte of sum
;
   sum :
; test array length
; exit with sum = 0, if nothing in array
        ex      de, hl          ; save base address of array
        ld      hl, 0           ; initialize sum to zero
; check for array length of zero
        ld      a, b            ; test array length
        or      a               ; exit with sum = 0
        ret     z               ; if length = 0
; initialize array pointer, sum
        ex      de, hl          ; move base address to hl
        ld      c, e            ; low, middle byte of sum = 0
                                ; high byte of sum = 0
                                ; c = high byte of sum
                                ; d = middle byte of sum
                                ; e = low byte of sum
; add word length elements to sum one at a time
; increment high byte of a sum whenever a carry
; occurs.
sum_word :
        ld      a, e            ; add low bytes of element
        add     a, (hl)         ; and sum
        ld      e, a
        inc     hl
        ld      a, d            ; add high byte of element to
        adc     a, (hl)         ; to middle byte of sum
        ld      d, a
        jr      nc, next_word   ; jump if no carry
        inc     c               ; else increment high byte
next_word :
        inc     hl
        djnz    sum_word
        ex      de, hl          ; middle and low byte of sum
        ld      a, c            ; high byte of sum
        ret
```

```
                ;
                ; routine : print
                ; purpose : print the measured area
                ;   entry : hl = integral part of the area
                ;           de = fractional part of the area
                ;
        print :
                push    hl
                ld      hl, message
                call    print_buffer
                pop     hl
                ; convert the integral and fractional
                ; part to ascii decimal
                push    de              ; save the binary fraction
                push    hl              ; save the binary integer
                ld      hl, quo_buffer
                pop     de              ; convert the binary integer
                call    binary_ascii    ; to ascii integer
                ld      hl, rem_buffer
                pop     de              ; convert the binary fraction
                call    binary_ascii    ; to ascii fraction
                ; print the integral part
                ld      hl, quo_buffer
                call    print_buffer
                ; put a dot between the integral & fractional part
                ld      hl, dot
                call    char_print
                ; print the fractional part
                ld      hl, rem_buffer
                call    print_buffer
                ; send carriage return & line feed
                ; control characters
    cr_line_feed :
                ld      hl, return
                call    char_print
                ld      hl, line_feed
                call    char_print
                ret

                ;
                ; routine : binary_ascii
                ; purpose : convert a 16 bit signed binary value
                ;           to ascii decimal
                ;   entry : register l = low byte of buffer address
                ;           register h = high byte of buffer address
                ;           register e = low byte of value to convert
                ;           register d = high byte of value to convert
                ;    exit : the first byte of the buffer is the
                ;           length, followed by the characters
                ;
    binary_ascii :
                ; save parameters
                ld      (b_pointer), hl; save buffer pointer
                ex      de, hl          ; hl = value to convert
                ld      a, 0
                ld      (c_length), a   ; current buffer length = 0
                ld      a, h
                ld      (sign), a       ; save sign of value
                or      a               ; set flags
                jp      p, convert      ; jump if value is positive
                sub     a               ; else take absolute value
                sub     l               ; ( 0 - value )
                ld      l, a
                sbc     a, a            ; propagate borrow
                sub     h
                ld      h, a
                ; convert value to a string
        convert :
                ; hl := hl div 10        ; dividend, quotient
                ; de := de mod 10        ; remainder
                ld      de, 10          ; load divisor
                call    divide
                ld      a, e
                add     a, "0"          ; convert 0..9 to ascii '0'..'9'
                call    insert
                ; if quotient is not zero keep dividing
                ld      a, h            ; test quotient
                or      l
                jr      nz, convert
                ld      a, (sign)
                or      a
                jp      p, positive     ; branch if original value
                ld      a, "-"          ; positive
                call    insert          ; put a minus sign infront
        positive :
                ret
```

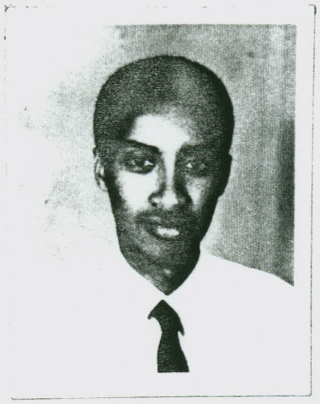
```
                ;
                ; routine : insert
                ; purpose : insert character at the front the buffer
                ;   entry : c_length = length of the buffer
                ;           b_pointer = current address of
                ;                       last character in buffer
                ;    exit : content of register a is inserted
                ;           immediately after length byte
                ;
        insert :
                push    hl              ; save hl
                ld      c, a            ; save character in c
                ; move the buffer right one character
                ld      hl, (b_pointer); get buffer pointer
                ld      d, h            ; de = source
                ld      e  l
                inc     hl              ; hl = destination
                ld      (b_pointer), hl; store new buffer pointer
                ld      a, (c_length)  ; test for length
                or      a
                jr      z, store        ; just store a character
                ld      b, a            ; b = loop counter
        shift_ :
                ex      de, hl          ; hl = source
                ld      a, (hl)         ; get next character
                ex      de, hl
                ld      (hl), a         ; store it
                dec     hl              ; decrement destination
                dec     de              ; decrement source
                djnz    shift_          ; decrement counter
                                        ; continue until all bytes moved
         store :
                ld      a, c            ; get character to insert
                ld      (hl), a         ; insert character at the front
                ld      a, (c_length)  ; increment current length
                inc     a
                ld      (c_length), a
                dec     hl              ; point to length byte of buffer
                ld      (hl), a         ; update it
                pop     hl              ; restore hl
                ret

                include "delay.asm"
                include "divide.asm"
                include "divide_.asm"
                include "multiply.asm"
                ; data area
                org     0500h
     sinusoidal :
                db      100, 106, 113, 119, 125, 131, 137, 143, 148, 154
                db      159, 164, 168, 173, 177, 181, 184, 188, 190, 193
                db      195, 197, 198, 199, 200, 200, 200, 199, 198, 197
                db      195, 193, 190, 188, 184, 181, 177, 173, 168, 164
                db      159, 154, 148, 143, 137, 131, 125, 119, 113, 106
                db      100, 094, 087, 081, 075, 069, 063, 057, 052, 046
                db      041, 036, 032, 027, 023, 019, 016, 012, 010, 007
                db      005, 003, 002, 001, 000, 000, 000, 001, 002, 003
                db      005, 007, 010, 012, 016, 019, 023, 027, 032, 036
                db      041, 046, 052, 057, 063, 069, 075, 081, 087, 094
                org     01000h
          dot : db      2eh
       return : db      0dh
    line_feed : db      0ah
      message : db      40, "the area of the object (in sqr cm) is : "
                org     0c000h
    sen_buffer : ds     y_data_length
                org     $ + 100
    len_buffer : ds     512
                org     $ + 100
    quo_buffer : ds     50
                org     $ + 100
    rem_buffer : ds     50
                org     $ + 100
          sign : ds     1
          ramp : ds     1
         count : ds     1
         dummy : ds     2
       divisor : ds     2
      dividend : ds     2
      c_length : ds     1
      buffer_1 : ds     255
      buffer_2 : ds     255
     pointer_1 : ds     3
     pointer_2 : ds     3
     b_pointer : ds     2
    location_1 : ds     2
    location_2 : ds     2
                end
```

References

1. EDWARD A.PARRISH & A.BARKANA, "Signal perturbation theory applied to boundary detection," *IEEE Trans. Ind. Electron Contr. Instrum.*, VOL IECI-22, pp 197-201, May 1975.
2. JHON J.STANAWAY & GERALD COOK, "Experimental verification of signal perturbation applied to the accuracy of position detection," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-17, pp 369-374, Aug.1970.
3. EDWARD A.PARRISH & JOHN W.STOUGHTON, " Signal perturbation techniques for achievement of linear position transfer characteristics," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-19, pp 1-3, Feb. 1972.
4. PI-FUAY CHEN & WILLIAM W.SEEMULLER, "Area measurement using a chemical array with image perturbation," *IEEE Trans. Ind. Electron. Control. Instrum.*, VOL IECI-19, pp 81-85, Aug. 1972.
5. WILLIAM W. SEEMULLER & PI-FUAY CHEN, "Area measurement using random image perturbation with discrete array," *IEEE Trans. Instrum. Meas.*, VOL IM-21, pp 140-144, May. 1972.
6. PI-FUAY CHEN & WILLIAM W.SEEMULLER, "Further development on area measurement using sinusoidal image perturbation," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-20, pp 90-92, May 1973.
7. PI-FUAY CHEN, "Image boundary detection using array system," *IEEE Trans. Instrum. Meas.*, VOL IM-24, pp 79-83, Feb. 1975.
8. EDWARD A.PARRISH,JR., & JAMES H.AYLOR, "Hardware implementation of signal perturbation theory," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-20, pp 258-264, Nov. 1973.
9. PI-FUAY CHEN, "Optimum perturbation signal wave form for sensing arrays," *IEEE Trans. Instrum. Meas.*, VOL IM-19, pp 136-139, May 1970.

Bibliography

1. EUGENE S.McVEY & PI-FUAY CHEN, "Improvement of position and velocity detecting accuracy by signal perturbation," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-16, pp 94-98, July 1969.
2. E.A PARRISH & A.BARKANA, "The effect of signal perturbation parameters on images produced on film by light emitting diodes," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-23, pp 153-161, May 1976. 3. PINGALI S.SARMA, "A microprocessor based system for area measurement," *IEEE Trans. Instrum. Meas.*, VOL IM-23, Sep. 1984, pp 168-171, Sep. 1984.
4. EDWARD A.PARRISH, "On improvement of position detection accuracy using signal perturbation theory," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-17, pp 7-9, Feb. 1970.
5. J.H AYLOR & EDWARD A.PARRISH, "Optimum design of a position detection system with sinusoidal perturbation signal," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-19, pp 114-119, Nov. 1972.
6. PI-FUAY CHEN & JAMES W.GDADDEN, "Chemical array as a position detector," *IEEE Trans. Instrum. Meas.*, VOL IM-21, pp 64-66, Feb. 1972.
7. E.A.PARRISH,JR., & J.W STOUGHTON, "Achieving improved position detection using a modified triangular perturbation signal," *IEEE Trans. Instrum. Meas.*, VOL IM-21, pp 78-80, Feb. 1972.
8. GERALD COOK, "Some practical limitation on the use of perturbation signal for resolution enhancement," *IEEE Trans. Ind. Electron. Contr. Instrum.*, VOL IECI-20, pp 114-117, Aug.1973.
9. PI-FUAY CHEN & WILLIAM W.SEEMULLER, "Center of mass measurement using an array system," *IEEE Trans. Instrum. Meas.*, VOL IM-20, pp 78-83, March 1973.



Signed Declaration

The basic materials of this thesis work, which deals with the effect of perturbation on measurement, has been gathered from *IEEE transaction on Instrumentation & Measurement and Industrial Electronics & Control Instrumentation,* and no originality is claimed.

However, the specific design and experimental setup, together with the results, are my own original work, which has never been presented for a Degree in any other University.

All source of material used for this thesis work has been listed in the reference and bibliography sections and dully acknowledged.

Getachew Hailu

Getachew Hailu

Addis Ababa University
School of Graduate Studies
Department of Electrical Engineering